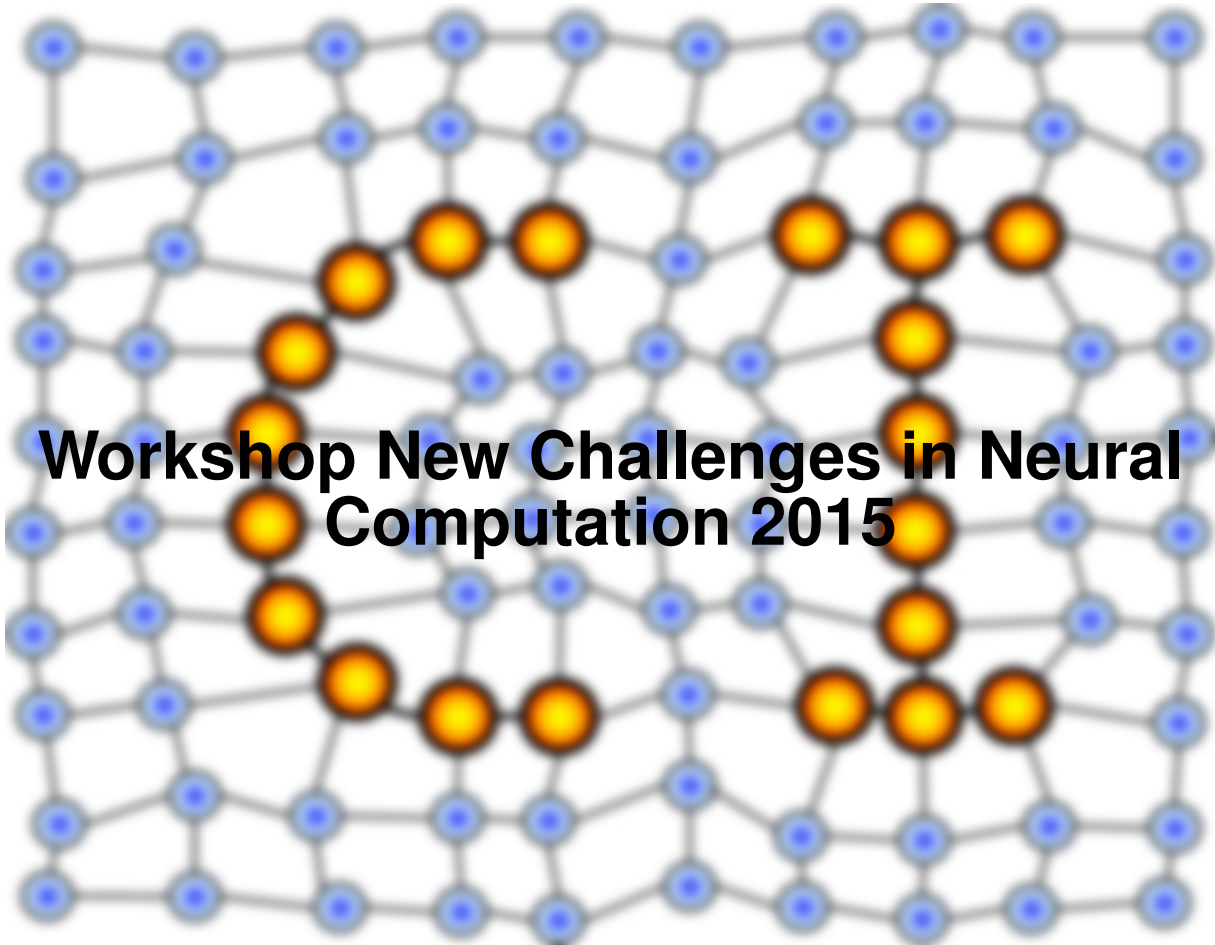


MACHINE LEARNING REPORTS



Workshop New Challenges in Neural Computation 2015

Report 03/2015

Submitted: 01.10.2015

Published: 10.10.2015

Barbara Hammer¹, Thomas Martinetz², Thomas Villmann³ (Eds.)

(1) CITEC - Centre of Excellence, University of Bielefeld, Germany

(2) Institute for Neuro- and Bioinformatics, University of Lübeck, Germany

(3) Faculty of Mathematics / Natural and Computer Sciences, University of Applied Sciences
Mittweida, Germany

Table of contents

<i>New Challenges in Neural Computation - NC² 2015</i> (B. Hammer, T. Martinetz, T. Villmann).....	5
<i>Keynote Talk: Representation Learning for Control</i> (J. Boedecker).....	6
<i>Keynote Talk: Machine Learning Meets Image Analysis: From looking inside ourselves to gazing at the stars</i> (C. Igel).....	7
<i>Archetypal Analysis as an Autoencoder</i> (C. Bauckhage, K. Kersting, F. Hoppe, C. Thureau).....	8
<i>Learning Transformation Invariance from Global to Local</i> (J. Hocke, T. Martinetz).....	16
<i>Polynomial approximation of spectral data in LVQ and Relevance Learning</i> (F. Melchert, U. Seiffert, M. Biehl).....	25
<i>Dissimilarity Extraction in a Median Variant of Learning Vector Quantization</i> (D. Nebel, M. Kaden).....	33
<i>Towards Dimensionality Reduction for Smart Home Sensor Data</i> (B. Mokbel, A. Schulz).....	41

Impact of Regularization on the Model Space for Time Series Classification
 (W. Aswolinskiy, R. F. Reinhart, J. Steil).....49

Ensembles of Neural Oscillators
 (D. Koryakin, F. Schrodtt, M. V. Butz).....57

Ensemble Methods and Active Learning in HCI
 (P. Thiam, M. Kächele, F. Schwenker, G.Palm).....65

Predictable Feature Analysis
 (S. Richthofer, L. Wiskott).....68

Incremental learning of action models as HMMs over qualitative trajectory representations
 (M. Panzner, P. Cimiano).....76

On the Applicability of Recurrent Neural Networks for Pattern Recognition in Electroencephalography Signals
 (M. Binz, S. Otte, A. Zell).....85

Population Monte Carlo Meets Contrastive Divergence Learning
 (O. Krause, A. Fischer, C. Igel).....93

CAPTCHA Recognition with Active Deep Learning
 (F. Stark, C. Hazırbaş, R. Triebel, D. Cremers).....95

Intrinsic Plasticity: A Simple Mechanism to Stabilize Hebbian Learning in Multilayer Neural Networks
 (M. Teichmann, F. H. Hamker).....103

Identifying bank stress by deep learning of news
(S. Rönnqvist, P. Sarlin).....112

Visualisation of heterogeneous data with simultaneous feature saliency using Generalised Generative Topographic Mapping
(S. Mumtaz, M. F. Randrianandrasana, G. Bassi, I. T. Nabney).....114

Incremental Class Learning and Novel Class Detection of Gestures Using Ensemble
(H. Al-Behadili, A. Grumpe, C. Dopp, C. Wöhler).....122

Attention as cognitive, holistic control of the visual system
(F. Beuth, F. H. Hamker).....133

Learning Conditional Mappings between Population-Coded Modalities
(F. Schrodt, M. V. Butz).....141

Nyström approximation toolbox
(A. Gisbrecht, F.-M. Schleif).....149

New Challenges in Neural Computation NC² – 2015

Barbara Hammer¹, Thomas Martinetz², and Thomas Villmann³

1 – Cognitive Interaction Technology – Center of Excellence,
Bielefeld University, Germany

2 – Institute for Neuro- and Bioinformatics, University of Lübeck, Germany

3 – Faculty of Mathematics / Natural and Computer Sciences,
University of Applied Sciences Mittweida, Germany

The workshop New Challenges in Neural Computation, NC², takes place for the sixth time, accompanying the prestigious GCPR conference. This year, GCPR is collocated with the VMV conference in Aachen, Germany. The town's history dates back to the Neolithic, and it is well known by its rich cultural, archeological, and architectural heritage. Its main cathedral has been the first German cultural monument (the second cultural monument worldwide) to become part of UNESCO's world heritage site. The workshop itself centres around challenges and novel developments of neural systems and machine learning, covering recent research in theoretical advances as well as practical applications. This year, twenty contributions from international participants have been accepted as regular contributions, spanning the range from deep learning, dimensionality reduction, information transfer and representation learning to models for time series data and sensor streams. In addition, we welcome two internationally renowned researchers as guest speakers, Prof. Dr. Christian Igel from Copenhagen University, Denmark, talks about 'Machine Learning Meets Image Analysis: From looking inside ourselves to gazing at the stars' and Dr. Joschka Boedecker from University of Freiburg, Germany, presents his work on 'Representation Learning for Control'. The workshop is supported by the German Neural Network Society (GNNS), and by the CITEC centre of excellence from Bielefeld University, Germany. Within the workshop, a meeting of the GI Fachgruppe on Neural Networks takes place.

We would like to thank our international program committee for their work in reviewing the contributions in a short period of time, the organizers of GCPR for their excellent support, as well as all participants for their stimulating contributions to the workshop.

Keynote talk: Representation Learning for Control

Joschka Boedecker, University of Freiburg, Germany

Abstract:

Defining features for control learning tasks with high-dimensional inputs is challenging. A balance needs to be found between compressing the state-dimensionality for fast convergence of the learning algorithm on the one hand, and retaining enough information about the full state of the system on the other. Learning features for these tasks automatically from data has received increasing interest lately. In this talk, I will review some of these approaches, highlighting our recent "Embed to Control" method which learns a latent representation and a locally linear transition model that facilitates optimal control by design.

**Keynote talk: Machine Learning Meets Image Analysis:
From looking inside ourselves to gazing at the stars**

Christian Igel, University of Copenhagen, Denmark

Abstract:

Machine learning (ML) plays an increasing role in image analysis. This talk presents recent examples from medical imaging and astronomy, ranging from applying standard ML algorithms to hand-crafted image features, over supervised feature learning using deep neural networks, to unsupervised image categorisation.

Archetypal Analysis as an Autoencoder

C. Bauckhage¹, K. Kersting², F. Hoppe³, and C. Thureau³

¹ Fraunhofer IAIS, St. Augustin, Germany

² TU Dortmund, Dortmund, Germany

³ Twenty Billion Neurons GmbH, Berlin, Germany

Abstract. We present an efficient approach to archetypal analysis where we use sub-gradient algorithms for optimization over the simplex to determine archetypes and reconstruction coefficients. Runtime evaluations reveal our approach to be notably more efficient than previous techniques. As an practical application, we consider archetypal analysis for autoencoding.

1 Introduction

Archetypal analysis is a matrix factorization method specifically conceived for latent factor analysis [8]. Since it allows for dimensionality reduction and sparse coding alike, it is applicable to feature extraction, clustering, or classification [3].

Contrary to related approaches, latent factors or *archetypes* found through archetypal analysis characterize extremes rather than averages. Archetypes do not rely on implicit density assumptions such as, for example, eigenvectors or cluster centroids which are tailored towards globally or locally Gaussian data. Rather, archetypal analysis introduces a form of symmetry into latent factor modeling: *archetypes are convex combinations of data points and data points are explained as of convex combinations of archetypes.*

It is therefore faithful to the nature of data. For instance, archetypes of non-negative data will be non-negative, too. Also, since archetypes are convex combinations of actual data, they closely resemble certain data points and thus do not require reification when interpreted. These properties are appealing in the sciences [5, 16, 23, 24] as well as in computer vision [2, 6, 19, 21, 25, 30].

However, computing optimal archetypes is an NP hard problem [1] and even though efficient approximations have become available [3, 10, 20, 22, 26], considerable computational costs still hamper the broader use of the method.

In this paper, we address this issue and propose a novel, highly efficient algorithm for archetypal analysis. Applying sub-gradient procedures for quadratic optimization over the simplex we observe clear runtime gains over previous methods so that archetypal analysis becomes applicable to very large data sets. As a corresponding practical application, we present and discuss first experiments on archetypal analysis as an autoencoder for joint feature learning in image analysis.

2 Archetypal Analysis, Properties, and Algorithms

The basic setting for archetypal analysis is as follows: Given an $m \times n$ data matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n]$ and an integer $k \leq \min\{m, n\}$, determine a column stochastic $n \times k$ matrix \mathbf{B} and a column stochastic $k \times n$ matrix \mathbf{A} such that $\mathbf{X} \approx \mathbf{XBA} = \mathbf{ZA}$.

The columns \mathbf{z}_j of the $m \times k$ matrix \mathbf{Z} are called the *archetypes* of the data. Since the column vectors \mathbf{b}_j of \mathbf{B} are stochastic, the entries of \mathbf{B} obey

$$b_{ij} \geq 0 \quad \wedge \quad \sum_{i=1}^n b_{ij} = 1 \quad (1)$$

and each archetype $\mathbf{z}_j = \mathbf{Xb}_j$ is a convex combination of the data vectors in \mathbf{X} . As the column vectors \mathbf{a}_i of \mathbf{A} are stochastic, too, the entries of \mathbf{A} obey

$$a_{ji} \geq 0 \quad \wedge \quad \sum_{j=1}^k a_{ji} = 1 \quad (2)$$

and we realize that archetypal analysis approximates each data vector $\mathbf{x}_i \approx \mathbf{Za}_i$ as a convex combination of the archetypes in \mathbf{Z} .

The problem of computing archetypal analysis can be cast as the following constrained quadratic optimization objective

$$\min_{\mathbf{A}, \mathbf{B}} E = \|\mathbf{X} - \mathbf{XBA}\|^2 \quad (3)$$

subject to the constraints in (1) and (2)

which is an NP-hard Euclidean sum of square clustering problem [1]. While E is convex in either \mathbf{A} or \mathbf{B} , it is not convex in their product \mathbf{AB} and typically has numerous local minima. Known solution strategies therefore randomly initialize both factor matrices and update them iteratively; we shall briefly discuss these algorithms below but first review some of the properties of archetypal analysis.

2.1 Properties of Archetypal Analysis

In [8], Cutler and Breiman prove that, if $k = 1$, the only archetype coincides with the sample mean; for $k > 1$, archetypes necessarily reside on the data convex hull and increasing the number of archetypes improves the approximation of the data convex hull (see Fig. 1).

Once suitable archetypes have been determined, each data point \mathbf{x}_i can either be reconstructed exactly or approximated as a convex combination of the \mathbf{z}_j (see Fig. 2). As the corresponding coefficient vector \mathbf{a}_i is stochastic, it can be interpreted as a distribution over the archetypes and thus be embedded in a simplex spanned by the archetypes (see Fig. 2). This allows for soft clustering or classification since the coefficients a_{ji} correspond to probabilities $p(\mathbf{x}_i | \mathbf{z}_j)$ which indicate membership to classes or concepts represented by the archetypes \mathbf{z}_k .

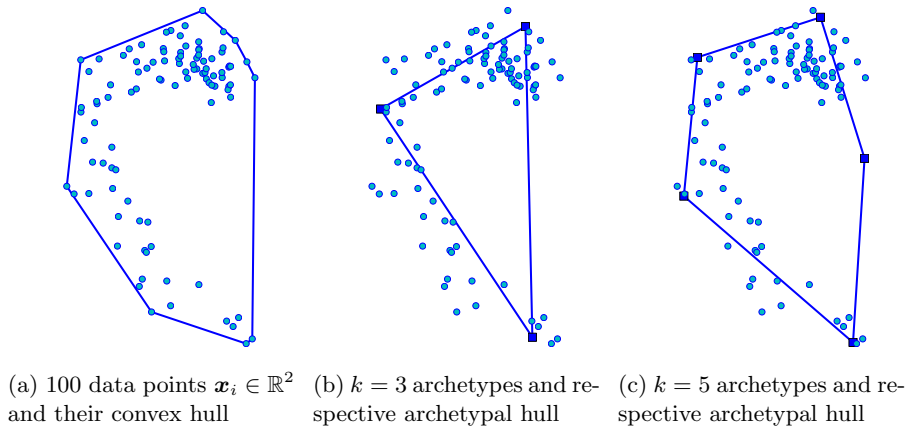


Fig. 1: Archetypal analysis approximates the convex hull of a set of multivariate data. Increasing the number k of archetypes improves the approximation.

2.2 Traditional Algorithms for Archetypal Analysis

Cutler and Breiman [8] proposed an alternating least squares approach where they randomly initialize \mathbf{B} and solve (3) for \mathbf{A} . Given \mathbf{A} , they solve (3) for \mathbf{B} and repeat. This procedure provably converges towards a local minimum and can be implemented using common solvers for quadratic programming. For better efficiency, Bauckhage and Thureau [3] suggested intelligent initialization strategies and an active set algorithm and thus achieved significant accelerations.

Morup and Hansen [20] proposed an alternating projected gradient approach. Observing that

$$E = \|\mathbf{X} - \mathbf{XBA}\|^2 = \text{tr} \left[\mathbf{X}^T \mathbf{X} - 2\mathbf{X}^T \mathbf{XBA} + \mathbf{A}^T \mathbf{B}^T \mathbf{X}^T \mathbf{XBA} \right] \quad (4)$$

we have $\nabla_{\mathbf{A}} E = 2[\mathbf{Z}^T \mathbf{Z} \mathbf{A} - \mathbf{Z}^T \mathbf{X}]$ and $\nabla_{\mathbf{B}} E = 2[\mathbf{X}^T \mathbf{XBA} \mathbf{A}^T - \mathbf{X}^T \mathbf{X} \mathbf{A}^T]$ so that archetypal analysis can also be computed by means alternating updates $\mathbf{A} \leftarrow \mathbf{A} - \eta_{\mathbf{A}} \nabla_{\mathbf{A}}$ and $\mathbf{B} \leftarrow \mathbf{B} - \eta_{\mathbf{B}} \nabla_{\mathbf{B}}$ where $\eta_{\mathbf{A}}$ and $\eta_{\mathbf{B}}$ are step size parameters. Since the gradient steps may lead out of the constraint sets, updated columns of \mathbf{A} and \mathbf{B} might not be stochastic and need to be projected back into their feasible regions which are the standard k and standard n simplex, respectively. Morup and Hansen, too, consider intelligent initializations for which they resort to the FASTMAP heuristic [11].

The methods in [3, 20] run much faster than the original one [8]. Still, they invoke rather costly quadratic optimization routines or require costly projections of gradients onto a feasible set. Next, we propose an approach to archetypal analysis that avoids such overhead.

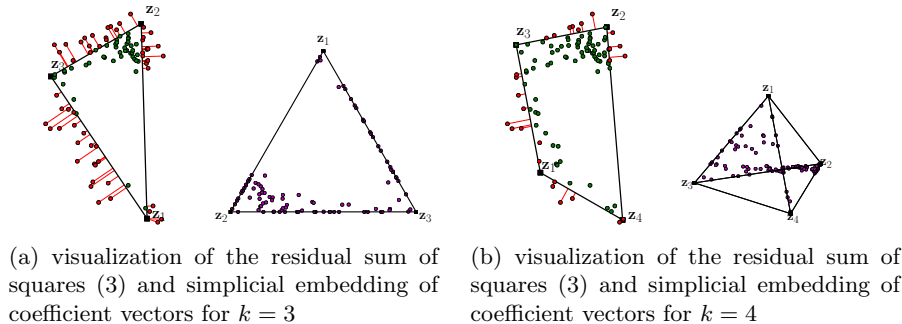


Fig. 2: While data inside an archetypal hull can accurately be expressed as convex combinations of archetypes, the constraints in (2) cause data on the outside to be mapped to the nearest point on the hull. For data point \mathbf{x}_i , the coefficient vector \mathbf{a}_i is stochastic and thus resides in a simplex whose vertices correspond to the archetypes \mathbf{z}_j .

Algorithm 1 greedy Frank-Wolfe procedure to compute matrix \mathbf{A} whose columns reside in the simplex Δ^{k-1}

Require: data matrix \mathbf{X} , matrix of archetypes \mathbf{Z} , and parameter $t_{\max} \in \mathbb{N}$

$\mathbf{A} \leftarrow [\mathbf{e}_1, \mathbf{e}_1, \dots, \mathbf{e}_1]$ where $\mathbf{e}_1 = [1, 0, \dots, 0]^T \in \mathbb{R}^k$ // initialize $k \times n$ matrix \mathbf{A}

$t \leftarrow 0$

repeat

$\mathbf{G} = \nabla_{\mathbf{A}} E = 2 [\mathbf{Z}^T \mathbf{Z} \mathbf{A} - \mathbf{Z}^T \mathbf{X}]$ // compute gradient matrix

for $i \in \{1, \dots, n\}$ **do** // update columns \mathbf{a}_i of \mathbf{A}

$j = \operatorname{argmin}_l G_{il}$

$\mathbf{a}_i \leftarrow \mathbf{a}_i + 2/(t+2) \cdot (\mathbf{e}_j - \mathbf{a}_i)$

$t \leftarrow t + 1$

until updates “become small” or $t = t_{\max}$

3 Rapid Archetypal Analysis

Our main observation w.r.t. the problem of efficient archetypal analysis is that the columns \mathbf{a}_i of \mathbf{A} and the columns \mathbf{b}_j of \mathbf{B} reside in the standard simplices Δ^{k-1} and Δ^{n-1} , respectively. In other words, the columns of either factor matrix are elements of a convex set. Furthermore, if the factor matrices are determined in an alternating manner, that is in a manner where we assume $\mathbf{Z} = \mathbf{X}\mathbf{B}$ to be given in order to update our current estimate of \mathbf{A} and then fix \mathbf{A} to update our current estimate of \mathbf{B} , the objective function in (3) becomes convex in either \mathbf{A} or \mathbf{B} . This, however, is to say that both update steps constitute a convex minimization problem over a convex set and can thus be tackled using the efficient Frank-Wolfe procedure [12].

Our idea is thus to refrain from using elaborate quadratic programming but to harness the efficiency of computing gradients $\nabla_{\mathbf{A}} E$ and $\nabla_{\mathbf{B}} E$ while avoiding costly back projections into the feasible set. This can be accomplished if sub-

Algorithm 2 greedy Frank-Wolfe procedure to compute matrix \mathbf{B} whose columns reside in the simplex Δ^{n-1}

Require: data matrix \mathbf{X} , matrix of coefficients \mathbf{A} , and parameter $t_{\max} \in \mathbb{N}$
 $\mathbf{B} \leftarrow [\mathbf{e}_1, \mathbf{e}_1, \dots, \mathbf{e}_1]$ where $\mathbf{e}_1 = [1, 0, \dots, 0]^T \in \mathbb{R}^N$ // initialize $n \times k$ matrix \mathbf{B}
 $t \leftarrow 0$
repeat
 $\mathbf{G} = \nabla_{\mathbf{B}} E = 2 [\mathbf{X}^T \mathbf{X} \mathbf{B} \mathbf{A} \mathbf{A}^T - \mathbf{X}^T \mathbf{X} \mathbf{A}^T]$ // compute gradient matrix
for $j \in \{1, \dots, k\}$ **do** // update columns \mathbf{b}_j of \mathbf{B}
 $i = \operatorname{argmin}_l G_{jl}$
 $\mathbf{b}_j \leftarrow \mathbf{b}_j + 2/(t+2) \cdot (\mathbf{e}_i - \mathbf{b}_j)$
 $t \leftarrow t + 1$
until updates “become small” or $t = t_{\max}$

gradient updates are performed along affine directions $\mathbf{e}_j - \mathbf{a}_i$ and $\mathbf{e}_i - \mathbf{b}_j$ within the simplices Δ^{k-1} and Δ^{n-1} , respectively. In a nutshell, this idea leads to the update algorithms 1 and 2 which are variants of a recent algorithm by Clarkson [7] which itself is a variant of the celebrated Frank-Wolfe procedure. A detailed analysis of this algorithm is beyond the scope of this paper but we point out that it quickly achieves ϵ -approximations of the optimal solution that are provably sparse. For a recent excellent survey of projection-free convex optimization, we refer to [17].

In extensive runtime evaluations (whose details we omit due to lack of space), we examined the behavior of this new algorithm under various choices of the number n of data, the dimensionality m of data, and the number k of archetypes to be determined and found our approach to be two to three orders of magnitude faster than the previous methods in [3, 20].

4 Application: Archetypal Analysis as an Autoencoder

In this section, we consider a practical application of archetypal analysis in the context of feature learning. Given that our new algorithm is much faster than previous methods, it now appears practical to apply archetypal analysis not only to sets of images [25] but to considerably larger sets of image patches.

Our application example is motivated by the observation that neural networks are back with a vengeance! Owing to the recent success of deep learning architectures in computer vision, speech recognition, or automatic translation [9, 13–15, 18, 27–29], research in these fields currently undergoes a paradigm shift. At the heart of many deep learning architectures especially for image analysis is the idea of using autoencoders for feature learning.

Looking at the overall objective of archetypal analysis, namely to find factor matrices such that $\mathbf{X} \approx \mathbf{X} \mathbf{B} \mathbf{A}$, we realize that it is indeed an autoencoder that maps \mathbf{X} onto itself and the preliminary experiments in this section are intended to fathom the potential of archetypal analysis for joint feature learning.

Figure 3(a) shows four images from which we extracted a total of 3844 patches of size 16×16 pixels which we represent in terms of data vectors $\mathbf{x}_i \in \mathbb{R}^{256}$.

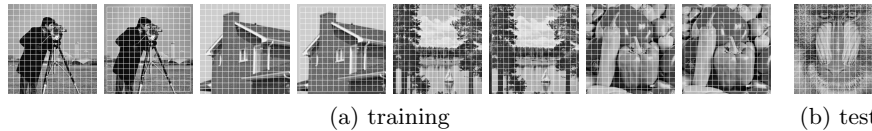


Fig. 3: Image patches (16×16 pixels) for archetypal autoencoding experiments.

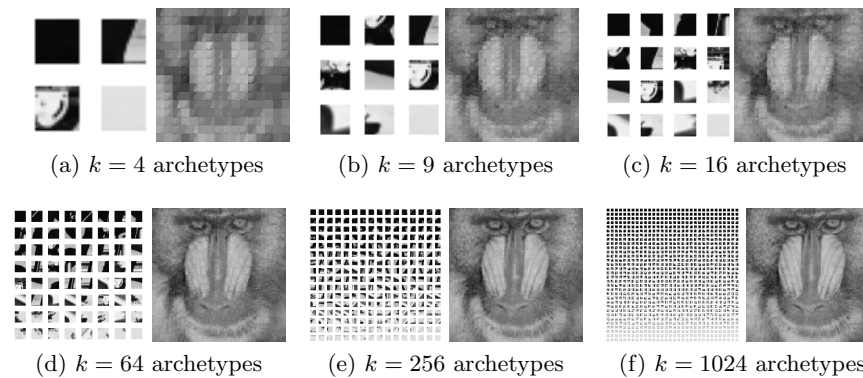


Fig. 4: Results of archetypal autoencoding using a growing number of archetypes. Note that the images on the left of each panel visualize archetypal image patches of size 16×16 in each case.

Greedy archetype computation on this data set runs in mere fractions of a second and, for $k \in \{4, 9, 16, 64, 256, 1024\}$ it determines the archetypes shown to the left of each panel in Fig. 4. Interestingly, especially for growing k , archetypes appear to be natural realizations (i.e. actual parts of images) of edge filters and the propensity of autoencoders to learn edge features is sometimes heralded as a special characteristic of deep learning approaches [18, 27]. Seen from the point of view of archetypal analysis, however, archetypal image patches are extreme in that they consist of very dark and very bright pixels, a condition typical for edges. This has already been noted in [3] and confirms early observations on neural feature learning [4].

Figure 3(b) shows a test image which we also subdivided into patches of size 16×16 and tried to reconstruct in terms of the archetypes determined from the training images. The corresponding results can be seen on the right of each panel in Fig. 4. As one would expect, for a growing number of archetypes, the reconstructions become better and we note that the reconstruction in Fig. 4(f) is actually an instance of sparse coding where the number of archetypes (1024) far exceeds the dimension of the data (256). Still, our algorithm did compute this reconstruction in less than a second. Overall, these results suggest that archetypal analysis indeed allows for joint feature learning for image representation. Archetypes were determined on training images independent from the test image, indicate edges like structures, and allow for reasonable reconstructions

of the test image. Given the favorable runtime characteristics of the algorithm proposed in this paper, these results therefore point at new directions for feature learning.

5 Conclusion

In this paper, we addressed the problem of efficient archetypal analysis for data matrix factorization. We proposed a novel algorithm which is based on greedy sub-gradient computations derived from the Frank-Wolfe algorithm.

As a practical application of our novel algorithm, we considered the use of archetypal analysis as an autoencoder for joint image feature learning. Archetypes were determined from a set of training images, were observed to represent edges or contours, and allowed for convincing reconstructions of test images. Running on the CPU of a single computer rather than on graphics hardware, our algorithm processed thousands of images patches in less than a second. Our results thus hint at possible applications of archetypal analysis in machine learning. In ongoing work, we are currently exploring the use of hierarchies of archetypal autoencoders where features learned on a lower level of the hierarchy are combined and form the input for the autoencoder on the next level so as to learn semantic representations of image content that are similar in spirit to those obtained from deep learning architectures.

References

1. Aloise, D., Deshapande, A., Hansen, P., Popat, P.: NP-Hardness of Euclidean Sum-of-Squares Clustering. *Machine Learning* 75(2), 245–248 (2009)
2. Asbach, M., Mauruschat, D., Plinke, B.: Understanding Multi-spectral Images of Wood Particles with Matrix Factorization. In: OCM. pp. 191–202. KIT Scientific Publishing, Karlsruhe (2013)
3. Bauckhage, C., Thureau, C.: Making Archetypal Analysis Practical. In: Denzler, J., Notni, G. (eds.) DAGM. LNCS, vol. 5748, pp. 272–281. Springer, Heidelberg (2009)
4. Bell, A., Sejnowski, T.: The Independent Components of Natural Images are Edge Filters. *Vision Research* 37(23), 3327–3338 (1997)
5. Chan, B., Mitchell, D., Cram, L.: Archetypal Analysis of Galaxy Spectra. *Monthly Notices of the Royal Astronomical Society* 338(3), 790–795 (2003)
6. Cheema, M., Eweiwi, A., Thureau, C., Bauckhage, C.: Action Recognition by Learning Discriminative Key Poses. In: ICCV (ICCV Workshops). pp. 1302–1309. IEEE Press, New York (2011)
7. Clarkson, K.: Coresets, Sparse Greedy Approximation, and the Frank-Wolfe Algorithm. *ACM Trans. on Algorithms* 6(4), 63:1–63:30 (2010)
8. Cutler, A., Breiman, L.: Archetypal Analysis. *Technometrics* 36(4), 338–347 (1994)
9. Deng, L., Hinton, G., Kingsbury, B.: New Types of Deep Neural Network Learning for Speech Recognition and Related Applications: An Overview. In: ICASSP. pp. 8599–8603. IEEE Press, New York (2013)
10. Eugster, M., Leisch, F.: Weighted and Robust Archetypal Analysis. *Computational Statistics & Data Analysis* 55(3), 1215–1225 (2011)

11. Faloutsos, C., Lin, K.I.: FastMap: A Fast Algorithm for Indexing, Data-mining and Visualization of Traditional and Multimedia Datasets. In: Proc. SIGMOD. pp. 163–174. ACM (1995)
12. Frank, M., Wolfe, P.: An Algorithm for Quadratic Programming. *Naval Research Logistics Quarterly* 3(1–2), 95–110 (1956)
13. Gao, J., X. He, W.Y., Deng, L.: Learning Continuous Phrase Representations for Translation Modeling. In: Proc. ACL (2014)
14. Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., Ng, A.: Deep Speech: Scaling up End-to-End Speech Recognition. arXiv:1412.5567 [cs.CL] (2014)
15. He, K., Zhang, X., Ren, S., Sun, J.: Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. arXiv:1502.01852 [cs.CV] (2015)
16. Huggins, P., Pachter, L., Sturmfels, B.: Toward the Human Genotype. *Bulletin of Mathematical Biology* 69(8), 2723–2735 (2007)
17. Jaggi, M.: Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization. *J. of Machine Learning Research* 28(1), 427–435 (2013)
18. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet Classification with Deep Convolutional Neural Networks. In: Proc. NIPS (2012)
19. Marinetti, S., Finesso, L., Marsilio, E.: Matrix factorization methods: application to Thermal NDT/E. In: Proc. Int. Workshop Advances in Signal Processing for Non Destructive Evaluation of Materials (2005)
20. Morup, M., Hansen, L.: Archetypal Analysis for Machine Learning and Data Mining. *Neurocomputing* 80, 54–63 (2012)
21. Prabhakaran, S., Raman, S., Vogt, J., Roth, V.: Automatic Model Selection in Archetype Analysis. In: Pinz, A., Pock, T., Bischof, H., Leberl, F. (eds.) DAGM. LNCS, vol. 7476, pp. 458–467. Springer, Heidelberg (2012)
22. Seth, S., Eugster, M.: Probabilistic Archetypal Analysis. arXiv:1312.7604v2 [stat.ML] (2014)
23. Stone, E., Cutler, A.: Exploring Archetypal Dynamics of Pattern Formation in Cellular Flames. *Physica D* 161(3–4), 163–186 (2002)
24. Thogersen, J., Morup, M., Damkiaer, S., Molin, S., Jelsbak, L.: Archetypal Analysis of Diverse *Pseudomonas Aeruginosa* Transcriptomes Reveals Adaptation in Cystic Fibrosis Airways. *BMC Bioinformatics* 14(1), 279 (2013)
25. Thureau, C., Bauckhage, C.: Archetypal Images in Large Photo Collections. In: ICSC. pp. 129–136. IEEE Press, New York (2009)
26. Thureau, C., Kersting, K., Wahabzada, M., Bauckhage, C.: Convex Non-negative Matrix Factorization for Massive Datasets. *Knowledge and Information Systems* 29(2), 457–478 (2011)
27. Tompson, J., Goroshin, R., Jain, A., LeCun, Y., Bregler, C.: Efficient Object Localization Using Convolutional Networks. In: CVPR. IEEE Press, New York (2015)
28. Vinyals, O., Toshev, A., Bengio, S., Erhan, D.: Show and Tell: A Neural Image Caption Generator. arXiv:1411.4555 [cs.CV] (2014)
29. Wiesler, S., Richard, A., Schlüter, R., Ney, H.: Mean-Normalized Stochastic Gradient for Large-Scale Deep Learning. In: ICASSP. pp. 180–184. IEEE Press, New York (2014)
30. Xiong, Y., Liu, W., Zhao, D., Tang, X.: Face Recognition via Archetypal Hull Ranking. In: ICCV. pp. 585–592. IEEE Press, New York (2013)

Learning Transformation Invariance from Global to Local

Jens Hocke, Thomas Martinetz

Institute for Neuro- and Bioinformatics, University of Lübeck

Abstract. Learning representations invariant to image transformations is fundamental to improving object recognition. We explore the connections between i-theory, Toroidal Subspace Analysis and slow subspace learning. All these methods can only achieve invariance to one transformation. Motivated by this limitation of these global methods we adapt the slow subspace approach to a local convolutional setting. Experimentally we show invariance to multiple transformations, and test object recognition performance.

1 Introduction

Changes of an object's pose are one of the big challenges in visual object recognition. The pixel representation of an object can change dramatically when the object's pose changes. Often this problem is met by presenting many training examples of the object in different poses. However, to achieve human like capability to learn from few samples it seems mandatory to separate the invariance learning from the object recognition problem.

Convolutional neural networks [1, 2] are an early example of an architecture that helps coping with shift invariance. These convolutional networks do not come with an objective function to learn invariance. Their main goal is classification, and invariance is learned as part of classification.

A well suited objective to achieve an invariant representation is slowness [3–5]. The main assumption of slowness is, that there is some slowly changing signal contained in a temporal stream of data. By optimizing for a slowly changing signal in the representation, invariance to the transformations contained in that temporal stream is achieved.

Pairs of filters coupled by their energy, so called subspaces, have shown to be a very useful architecture. The subspaces have been introduced in the domain of self organizing maps [6], and have been transferred to the representation learning domain in form of the Independent Subspace Analysis (ISA) [7]. Soon slowness and subspace architectures were combined in [8], minimizing the energy change of the subspaces over time. Newer approaches [9–11] also include sparsity [12].

An approach derived from group theory is the Toroidal Subspace Analysis (TSA) [13]. The resulting representation also uses subspaces. In contrast to the slowness based subspace approaches, the energy of the subspaces is fixed for pairs of transformed image patches, and the error for encoding one patch in terms of

the other is minimized. Similar to TSA, gated models [14] minimize the encoding distance between pairs of transformed images. However, there products of filters are used to encode the transformations.

An explanation of how invariance could emerge in the ventral stream is offered in the i-theory [15]. From these theoretical insights on invariance, implications for the network structure can be derived.

After explaining the connections between i-theory, TSA and slow subspace learning methods and their drawbacks, we adapt the local subspace learning method by W. Zou et al. [9, 10] to convolutional learning and test this method for invariance and unique representation.

2 Transformation Groups and Invariance

Orbits can be used to achieve invariance to a group G of transformations. This is the core observation of the i-theory [15] as well as integral invariants. The group elements g are transformations of images $\mathbf{x} \in \mathbb{R}^D$. We denote the group's actions on an image by $g(\mathbf{x})$. The orbit $O_{\mathbf{x}} = \{g_i(\mathbf{x}) | g_i \in G\}$ of some image \mathbf{x} is induced by applying all transformations $g_i \in G$ to \mathbf{x} . This orbit is invariant to the transformations in G and unique for the object in \mathbf{x} . But it is a very high dimensional representation.

The high dimensionality can be handled by one dimensional projections $\langle g_i(\mathbf{x}), \mathbf{p}_n \rangle$, where $\mathbf{p}_n, n = 1, \dots, D$ are arbitrary projection vectors. If there are enough different projection vectors, a unique representation can be achieved. Besides reducing the dimensionality, this helps to avoid transforming the input image \mathbf{x} by applying the inverse transformation to the projection vectors instead

$$\langle g_i(\mathbf{x}), \mathbf{p}_n \rangle = \langle \mathbf{x}, g_i^{-1}(\mathbf{p}_n) \rangle. \quad (1)$$

So now we have projected orbits of images. In the i-theory probability distributions over these vectors are used to obtain an invariant representation. This helps analyzing the invariance problem. However, we found it hard to learn good representations using this probabilistic framework [16]. Therefore, we stay in the deterministic domain.

If we assume that the transformations g_r have only one parameter r (e.g. degree for rotation) and they are ordered by this parameter, we can assemble a matrix W . This matrix $W = (g_{r_1}^{-1}(\mathbf{p}), g_{r_2}^{-1}(\mathbf{p}), \dots, g_{r_N}^{-1}(\mathbf{p}))$ is composed of column vectors $\mathbf{w}_r = g_r^{-1}(\mathbf{p})$. The parameters r_i for the transformations are uniformly distributed $r_i = N/I \cdot (i - 1)$ with I being the maximum transformation parameter. In the following line of thinking, we assume one projection vector \mathbf{p} . Here, we abbreviate g_{r_i} by g_i and w_{r_i} by w_i . The representation \mathbf{y} of the image vector \mathbf{x} is obtained by

$$\mathbf{y} = W^T \mathbf{x}. \quad (2)$$

For the transformed image $g_j(\mathbf{x})$ we obtain \mathbf{y}' . If we observe a single entry y_i of \mathbf{y} while applying transformation g_j

$$y_i = w_i^\top \mathbf{x} \quad (3)$$

$$= g_j(w_i)^\top g_j(\mathbf{x}) \quad (4)$$

$$= w_{i+j}^\top g_j(\mathbf{x}) = y'_{i+j}, \quad (5)$$

we see that the entries of the representation vector shift indices. Only the first or last elements of \mathbf{y}' , depending on the direction of the transformation, may not be related to \mathbf{y} . By restricting the applicable transformations G to the set of toroidal group transformations, a relation to all entries in \mathbf{y} can be established.

These toroidal group transformations are turned into circular shifts in the representation vector \mathbf{y} . So via the Fourier transform of \mathbf{y} amplitudes invariant to toroidal group transformations can be found, while the phases encode the transformation parameter. Via the n -dimensional Fourier transform an extension to n parameters is possible.

2.1 Relationship of Invariance Learning Methods

In case the transformation group is not known or the transformation is hard to model, it is beneficial to learn W . Let $\mathbf{x}(t) = g(\mathbf{x}(t-1))$ at time t be a transformed version of an image $\mathbf{x}(t-1)$ in a sequence. From above we know that the Fourier amplitudes will not change. Only the phase will change according to the Fourier shift theorem. Thus, we can reconstruct $\mathbf{x}(t)$

$$\mathbf{x}(t) = W^{-1} F^{-1} R(\phi) F W \mathbf{x}(t-1) \quad (6)$$

if the phase shift ϕ encoded in a diagonal matrix $R(\phi)$ is known. This can be turned in a learning algorithm, where this autoencoder like energy term

$$E = \sum_t \|\mathbf{x}(t) - W^{-1} F^{-1} R(\phi) F W \mathbf{x}(t-1)\|, \quad (7)$$

and $R(\phi)$ are optimized in an alternating manner. Since the Fourier transformation is just an unitary transformation, it can be absorbed into W

$$E = \sum_t \|\mathbf{x}(t) - W^\top R(\phi) W \mathbf{x}(t-1)\|. \quad (8)$$

This is the essence of TSA [13], where usually the complex unit vectors on the diagonal of $R(\phi)$ are not coupled.

Related to TSA are slow subspace approaches. They have two main ingredients. They encourage a representation that allows reconstruction of $\mathbf{x}(t)$ from $W^\top \mathbf{x}(t)$, which can be achieved via an orthogonal basis W , an autoencoder term or sparse coding. In order to find an invariant representation changes in the subspace energies

$$e_i(t) = \sum_{k=0}^{K-1} (\mathbf{w}_{iK+k}^\top \mathbf{x}(t))^2 \quad (9)$$

of K -dimensional subspaces indexed by i are penalized. This is done either by minimizing their distance in consecutive samples or by minimizing their variance¹, which also indirectly minimizes the subspace energy of samples following each other. In the following we assume $K = 2$.

The relation of subspace methods to TSA and the i-theory can be seen if all energy terms are zero for any pair of group transformed images. Then subspace methods have found a basis W_{slow} , that can reconstruct all sample pairs $\mathbf{x}(t)$ and $\mathbf{x}(t-1)$ from a sequence, while $\mathbf{e}(t)$ does not change for consecutive samples. Only the pairs of activations $w_{i2}^\top \mathbf{x}(t)$ and $w_{i2+1}^\top \mathbf{x}(t)$ can change over time. This change can be interpreted as an angle change in polar coordinates, which is the only change TSA allows to reconstruct $\mathbf{x}(t)$ from $\mathbf{x}(t-1)$. From that, we see, any input image can be group transformed using W_{slow} and some matrix $R(\phi)$ for the angle change. Therefore, W_{slow} is also an optimal solution for the TSA model. The other way round, an optimal basis W_{TSA} learned by TSA, will always have a fixed $\mathbf{e}(t)$, and perfect self-reconstruction is guaranteed via not transformed pairs $\mathbf{x}(t)$ and $\mathbf{x}(t-1)$. Thus, W_{TSA} is also an optimal solution for the subspace model.

2.2 A Slow Subspace Autoencoder Model

The model we chose to build on is a slow subspace autoencoder model [10], that has been applied successfully in object recognition tasks. It follows the scheme mentioned above. There is a reconstruction and a slowness term, and in addition also sparsity is encouraged. The terms

$$E_{rec} = \sum_t \|\mathbf{x}(t) - W^\top W \mathbf{x}(t)\|_2^2 \quad (10)$$

$$E_{slow} = \sum_t \|\mathbf{z}(t) - \mathbf{z}(t-1)\|_1 \quad (11)$$

$$E_{sparse} = \sum_t \|\mathbf{z}(t)\|_1 \quad (12)$$

with the amplitudes

$$z_i(t) = \sqrt{e_i(t)} \quad (13)$$

are combined via

$$E = E_{rec} + \alpha E_{slow} + \beta E_{sparse} \text{ s.t. } \|\mathbf{w}_i\| = 1. \quad (14)$$

Note the unit norm constraint on the weight vectors \mathbf{w}_i . This is necessary to avoid \mathbf{w}_i to become a zero vector if large values of α or β are used. This energy model can now be optimized via stochastic gradient descent.

¹ The principle of minimizing the variance over time is also fundamental to the Slow Feature Analysis [5]. However, SFA is not operating on subspaces.

In case the sparsity term is omitted TSA [13] like bases are found using the same training samples (Figure 1a and b). These bases are global and the amplitudes are perfectly transformation invariant. However, a large amount of spatial information is lost with the phases. This representation is sensible to background clutter and factorizing the invariance problem seems impossible (e.g. one module doing translation invariance, the next rotation etc.) [17]. But factorizing the range of possible transformation parameters is possible according to the i-theory [15]. Factorizing the invariance means we are restricting it to a local window, which can be achieved via a sparsity term (Figure 1c). Of course this will decrease the invariance [18], but due to the local similarity of most transformations to shifts, a diverse set of transformations can be handled. By adding additional layers trained like this first module, the range of invariance can be increased.

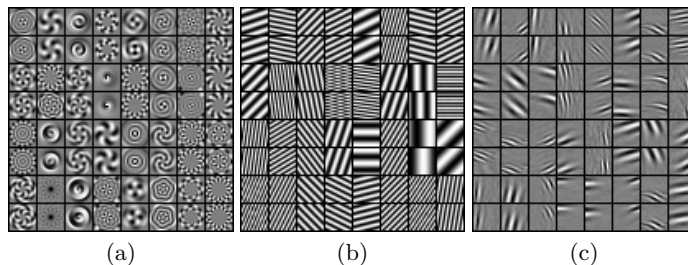


Fig. 1. The first 64 elements of global bases learned from rotated and shifted patches of random intensities are shown in (a) and (b). In (c) the first 64 elements of a basis learned with a sparsity prior from natural movie sequences is shown.

2.3 Convolutional Model

When the invariance is factorized to local windows by optimizing sparsity, many very similar shifted local subspaces are created (Figure 1c). We decrease this redundancy by training an adapted convolutional network with the energy terms

$$E_{rec} = \sum_t \|x(t) - \sum_j \left(\tilde{W}_j * u_s(d_s(W_j * x(t))) \right)\|_2^2 \quad (15)$$

$$E_{slow} = \sum_t \|d_s(z(t)) - d_s(z(t-1))\|_1, \quad (16)$$

where

$$z_i(t) = \sqrt{\sum_{k=0}^1 (W_{i2+k} * x(t))^2}. \quad (17)$$

Here, the vectors \mathbf{w} are replaced by filters W_j and their counter parts with all dimensions flipped \tilde{W}_j . The convolution operation is denoted by $*$. In addition the

downsampling and upsampling operators d_s and u_s with stride s are introduced, taking only every s -th value in each direction or reversing the downsampling by filling in zeros. Due to the network structure no sparsity needs to be enforced to learn local subspaces and the required computational resources stay moderate.

For training the first layer on sequence images, the images were preprocessed by ZCA filtering [19]. Using these preprocessed images, filters for the convolutional model were optimized by stochastic gradient descent. After training, the first layer output maps $d_s(z_i(t))$ can be computed via (17). The next layer is trained on the output of the first layer for unprocessed images. Because these outputs can be high dimensional, the number I of maps is reduced by filtering. As filters we use the principle components of $1 \times 1 \times I$ patches extracted from the outputs. This data is then ZCA filtered and used for optimizing the second layer filters. Higher layers can be computed analog to the second layer. For computing the outputs of higher layers, only the PCA step is needed, and thus, ZCA filtering is omitted.

3 Experiments

We trained a two layer version of the convolutional model using natural movie sequences from the van Hateren video database [20]. These are gray scale 128×128 pixel movies collected from television. For training the first layer with $\alpha = 50$, the stride was set to 6, the filter size was set to 15×15 pixels and 36 filters shown in Figure 2a were trained. Then using the learned filters, the first layer output was generated using stride 2. To train the second layer the 18 magnitude maps from the first layer output were reduced via PCA to three maps carrying more than 90% of the variance. Again for training the stride of the second layer was set to 6. The filter size was adapted to $15 \times 15 \times 3$ to handle all three maps and 108 filters were learned with $\alpha = 100$. We see the results for the top map in Figure 2b. Note, we did not analyze many of the parameters. One might find better choices. In particular, the second layer filters are critical and for many parameters no useful filters will be produced.

Clearly for both layers we obtain Gabor like filters (Figure 2). For the second layer the filters are repeated in every map, however with different intensities. These finding suggest invariance to small shifts in the first layer and an increased invariance to these shifts in the second layer. We tested this translation invariance and also rotation and scale invariance using 100 patches of 64×64 pixels from the van Hateren image database [21]. We measure the change in the output of each layer, as the input undergoes transformations. The MSE between the original and the transformed patch is taken and normalized against the largest MSE, assuming the patches are uncorrelated for these transformation parameters. The output of both layers were downsampled with stride 3.

The plots in Figure 3 validate our believe in invariance to small shifts. We also see invariance to rotation and scaling, because these transformations can locally be approximated by shifts. And additionally the invariance increases from the first to the second layer.

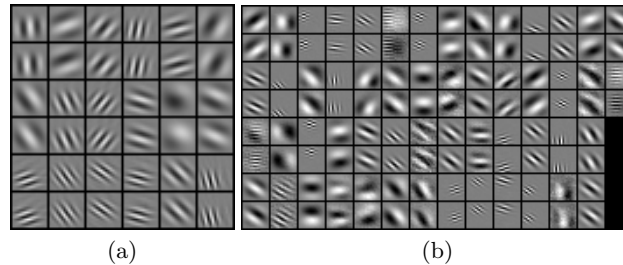


Fig. 2. Layer 1 filters are displayed in (a). In (b) only the top part of the second layer filters is shown. This top part, which is for the first output map, differs from the other parts only in the intensities of the filters.

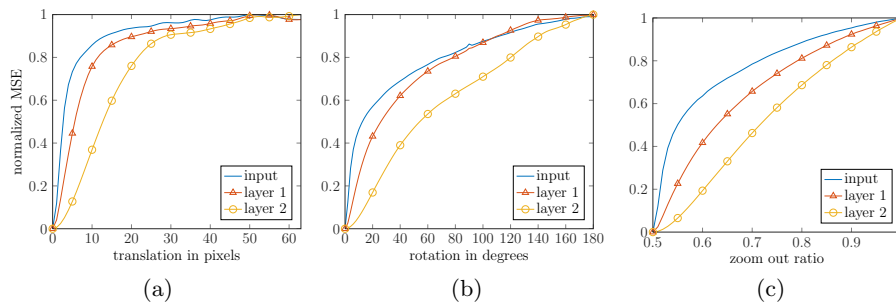


Fig. 3. Invariance experiment for varying degrees of shift (a), rotation (b), and scale (c). The normalized MSE in layer 1 and layer 2 is plotted along with results for the unprocessed input patches as reference.

Next we were interested in the effect of the stride. The strides for both layers were adapted simultaneously. Using the same approach as above we measured the MSE for different strides on shifted patches. The plots in Figure 4 show, that the first layer output is not affected. However, the second layer is. This is due to the change of the represented area. The larger the stride in the bottom layer the larger the area represented in the second layer.

These findings suggest using large strides. One of the main problems of invariant representations, however, is representing the input uniquely. To test how well information on fine image structures is retained at each layer we do k -NN classification ($k = 3$) on the MNIST [2] dataset. The classification error on the raw images is 3.09%. As we see in Table 1, there is a drop in the k -NN classification performance from layer 1 to layer 2, which can be reduced to a certain extent by choosing small stride sizes. This clearly indicates a loss of important information. Interestingly, the first layer error rates are significantly better than on the input images². We think this is due to the small non-affine transformations in MNIST, which may be handled well by the Gabor features.

² The state of the art error rate for MNIST is of 0.23% [22].

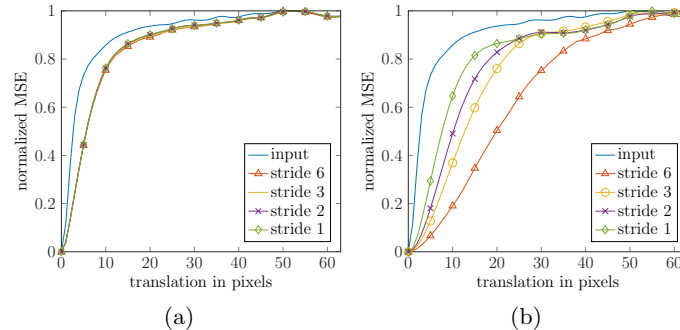


Fig. 4. The normalized MSE in layer 1 (a) and layer 2 (b) depending on the amount of shift is plotted for different strides. As reference also a curve for the input patches is shown.

	Layer 1	Layer 2
Stride 6	1.48	12.88
Stride 3	1.41	6.35
Stride 2	1.43	5.01
Stride 1	1.42	3.28

Table 1. Results for MNIST classification. The error rates are given in percent.

4 Conclusion

I-theory, TSA and slow subspace learning methods are closely related. Invariance learning based on anyone of these seems equally well suited, leaving aside optimization and implementation issues. However, if they learn global invariance, their application is very limited by their adaption to a single transformation group. Therefore, we implemented a convolutional method, which learns local invariance due to its structure. The experiments show indeed invariance to multiple transformations, with increase in invariance from layer to layer, while information loss also seems to be increased. This information loss remains an open problem to be solved before deeper networks using our training approach become useful. Interestingly, the first layer seems to be capable of handling non-affine transformations in MNIST, leading to improved classification results.

References

1. Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics* **36** (1980) 193–202
2. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **86**(11) (1998) 2278–2324
3. Hinton, G.E.: Connectionist learning procedures. *Artificial intelligence* **40**(1) (1989) 185–234

4. Földiák, P.: Learning invariance from transformation sequences. *Neural Computation* **3**(2) (1991) 194–200
5. Wiskott, L., Sejnowski, T.J.: Slow feature analysis: Unsupervised learning of invariances. *Neural computation* **14**(4) (2002) 715–770
6. Kohonen, T.: Emergence of invariant-feature detectors in the adaptive-subspace self-organizing map. *Biological Cybernetics* **75**(4) (1996) 281–291
7. Hyvärinen, A., Hoyer, P.: Emergence of phase-and shift-invariant features by decomposition of natural images into independent feature subspaces. *Neural computation* **12**(7) (2000) 1705–1720
8. Kayser, C., Einhäuser, W., Dümmer, O., König, P., Körding, K.: Extracting slow subspaces from natural videos leads to complex cells. In: *Artificial Neural Networks—ICANN 2001*. Springer (2001) 1075–1080
9. Zou, W.Y., Ng, A.Y., Yu, K.: Unsupervised learning of visual invariance with temporal coherence. In: *NIPS 2011 Workshop on Deep Learning and Unsupervised Feature Learning*. (2011)
10. Zou, W., Zhu, S., Yu, K., Ng, A.Y.: Deep learning of invariant features via simulated fixations in video. In: *Advances in Neural Information Processing Systems*. (2012) 3212–3220
11. Cadieu, C.F., Olshausen, B.A.: Learning intermediate-level representations of form and motion from natural movies. *Neural computation* **24**(4) (2012) 827–866
12. Olshausen, B.A., Field, D.J.: Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research* **37**(23) (1997) 3311–3325
13. Cohen, T., Welling, M.: Learning the irreducible representations of commutative lie groups. *arXiv preprint arXiv:1402.4437* (2014)
14. Memisevic, R.: Learning to relate images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **35**(8) (2013) 1829–1846
15. Anselmi, F., Leibo, J.Z., Rosasco, L., Mutch, J., Tacchetti, A., Poggio, T.: Unsupervised learning of invariant representations in hierarchical architectures. *CoRR abs/1311.4158* (2013)
16. Hocke, J., Martinetz, T.: Learning transformation invariance for object recognition. In: *Workshop New Challenges in Neural Computation 2014*. (2014) 20–25
17. Anselmi, F., Poggio, T.A.: Representation learning in sensory cortex: a theory. (2014)
18. Lies, J.P., Häfner, R.M., Bethge, M.: Slowness and sparseness have diverging effects on complex cell learning. *PLoS computational biology* **10**(3) (2014) e1003468
19. Bell, A.J., Sejnowski, T.J.: Edges are the” independent components” of natural scenes. In: *NIPS*. (1996) 831–837
20. van Hateren, J.H., Ruderman, D.L.: Independent component analysis of natural image sequences yields spatio-temporal filters similar to simple cells in primary visual cortex. *Proceedings of the Royal Society of London. Series B: Biological Sciences* **265**(1412) (1998) 2315–2320
21. van Hateren, J.H., van der Schaaf, A.: Independent component filters of natural images compared with simple cells in primary visual cortex. *Proceedings: Biological Sciences* **265**(1394) (Mar 1998) 359–366
22. Ciresan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE* (2012) 3642–3649

Polynomial Approximation of Spectral Data in LVQ and Relevance Learning

Friedrich Melchert^{1,2}, Udo Seiffert², and Michael Biehl¹

¹ University of Groningen, Johann Bernoulli Institute for Mathematics and Computer Science, P.O. Box 407, 9700 AK Groningen, The Netherlands

² Fraunhofer Institute for Factory Operation and Automation IFF, Sandtorstrasse 22, 39106 Magdeburg, Germany

Abstract. High dimensional data serves as input for a variety of classification tasks. In the case of spectral information, this data can be understood as discrete sampling of an (unknown) underlying function. In this paper we discuss an approach that improves classification performance for spectral data by expanding the data in terms of basis functions. Two real world spectral data classification problems demonstrate the advantages of the method.

Keywords: Classification; supervised learning; functional data; Learning Vector Quantization; relevance learning; dimensionality reduction

1 Introduction

A variety of real-world applications produce high-dimensional data that are usually difficult to handle with traditional methods. Apart from developing new methods suited for a high number of input dimensions, one possible solution is to use prior knowledge about the underlying structures of the input data for dimension reduction or data simplification. A very general and in most cases justifiable approach is to assume high dimensional data vectors are a discretized representation of a continuous function. This is true for different types of data, such as time series and spectral data, which are frequently high-dimensional.

Such data is usually recorded in order to serve as input data in a classification task. Different machine learning and classification algorithms can be applied, each having its own advantages and disadvantages. Prototype- and distance-based methods have the advantage of being intuitive to implement and interpret. In this paper an extension of the popular Learning Vector Quantization (LVQ) [6] is employed. In LVQ systems, prototypes serve as characteristic exemplars of their corresponding classes. In combination with an appropriate distance measure, they constitute an efficient method of classification [2].

The choice of the distance measure is the key to the design of an LVQ system [2]. In contrast to fixed Euclidean or other Minkowski measures, the Generalized Matrix LVQ (GMLVQ) makes use of the more flexible concept of relevance learning. In GMLVQ a parametrized distance measure is used and its parameters are determined in a data-driven training process [12]. Therefore, only the

basic structure of the distance measure has to be specified in advance. This approach offers greater flexibility and, moreover, the interpretation of the adaptive distance measure can give further insight into the structure of the data [14].

In this paper, we discuss the application of GMLVQ to functional input data. In [5] a functional representation of the relevances in GMLVQ was proposed and investigated, but samples and prototypes were still considered to be in original feature space. This paper presents an alternative approach that produces a functional representation of the data and performs GMLVQ adaptation in the space of coefficients for the functional representation. The idea was first presented in [13] where a wavelet expansion of mass spectrometry data was employed. Although this also led to a reduction of input dimensions, the wavelet expansion was motivated as an easy way of finding discriminative features of the sharply peaked mass spectra.

In [11] the SOM algorithm is modified for the unsupervised clustering of spectral information by expanding the input data in terms of B-Spline functions. This yields a reduction of input dimensions by a factor of two. In this paper we demonstrate that a similar approach using Chebyshev polynomials as functional basis can reduce the number of input dimensions even further without significant loss or even improvement of performance with respect to the full feature dataset. Since the potential benefit of the approach for the quality of the classification has been discussed in [8], here we focus on the aspect of dimensionality reduction.

2 Polynomial approximation of functional data

In order to reduce the number of input dimensions the approximation of the data using a weighted set of basis functions is considered. It is assumed, that the d -dimensional feature vectors $\mathbf{v}_i \in \mathbb{R}^d$ result from sampling a continuous (in general unknown) function $f_i(x)$.

$$\mathbf{v}_{i,j} = f_i(x_j) \quad \text{with } j = 1, 2, \dots, d. \quad (1)$$

For a given set of suitable basis functions $g_k(x)$ it is possible to find the coefficients $c_{i,j}$ so that

$$f_i(x) = \sum_{k=0}^{\infty} c_{i,j} \cdot g_k(x). \quad (2)$$

By limiting the maximum number of coefficients to n , and thus truncating the series, Eq. (2) becomes an approximation of the original function, which can be achieved by means of general approximation schemes like least mean squares or minimal maximum deviation.

From the obtained set of coefficients $c_{i,j}$ a new set of feature vectors $\mathbf{c}_i \in \mathbb{R}^n$ can be composed. Throughout the following we assume that $n < d$, obviously.

The quality of the functional approximation is highly dependent on the selection of an appropriate set of basis functions and, of course, on the number of approximation coefficients. Both choices should be guided by the specific properties of the input data. With respect to the number of functions and coefficients

the proposed approach is quite robust within a range of values n [8]. Possible basis functions include polynomials, trigonometric functions for periodic signals, wavelets or spline functions. In this paper, we focus on using a polynomial basis as an important example, more specifically the set of Chebyshev polynomials of the first kind. They are defined recursively by

$$\begin{aligned}T_0(x) &= 1 \\T_1(x) &= x \\T_n(x) &= 2xT_{n-1} - T_{n-2}(x)\end{aligned}\tag{3}$$

Using Chebyshev polynomials as basis functions Equation (2) becomes a Chebyshev series which is known to provide an efficient way to represent smooth non-periodic functions [3]. Within the context of this paper an open source MATLABTM library called *chebfun* [15] is employed to determine the coefficients of the series. Although the library provides a wide variety of functions in the context of Chebyshev polynomials, we only use the implementation of approximations for discrete data in this paper. For a more detailed description of the implementation see the documentation in [3].

3 Application to example datasets

The approach is demonstrated by applying it to two publicly available datasets as well as manually distorted copies thereof. All of the datasets contain spectral and thus functional input data. An illustration of the datasets is shown in Figure 1.

The first dataset, the *Wine* dataset (available from [9]), contains 124 samples of wine infrared absorption spectra in the range between 4000 and 400 cm^{-1} with 256 sampled values each. One sample of the dataset, which could be clearly identified as an outlier, was removed for the following analysis. The samples are labeled according their alcohol content: A two class problem is created by thresholding the alcohol level as described in [7], the resulting classes correspond to low and high alcohol content.

As a second dataset we consider the *Tecator* dataset (available from [16]), which comprises 215 reflectance spectra in the range from 850 to 1050 nm wavelength. The spectra are sampled equidistantly using 2 nm step size resulting in 100 sampled values per spectrum. The spectral information was acquired from meat probes and labeled according to their fat content. Similar to the *Wine* dataset the fat content is thresholded at its median in order to obtain two classes.

For further illustration of the presented approach both datasets are artificially distorted. For the *Wine* dataset, which seems to be pre-processed in terms of offset elimination, cf. Fig. 1a, a random offset is added to each of the spectra. This yields a dataset which will be referred to as *WineRO* (Wine with Random Offset) in the following. The spectra in the *Tecator* dataset already have different offsets (cf. Fig. 1b), so for distortion the offsets in the dataset are removed by subtracting the mean value of each spectrum. The resulting dataset will be referred to as the *TecatorNO* (Tecator with No Offset) dataset.

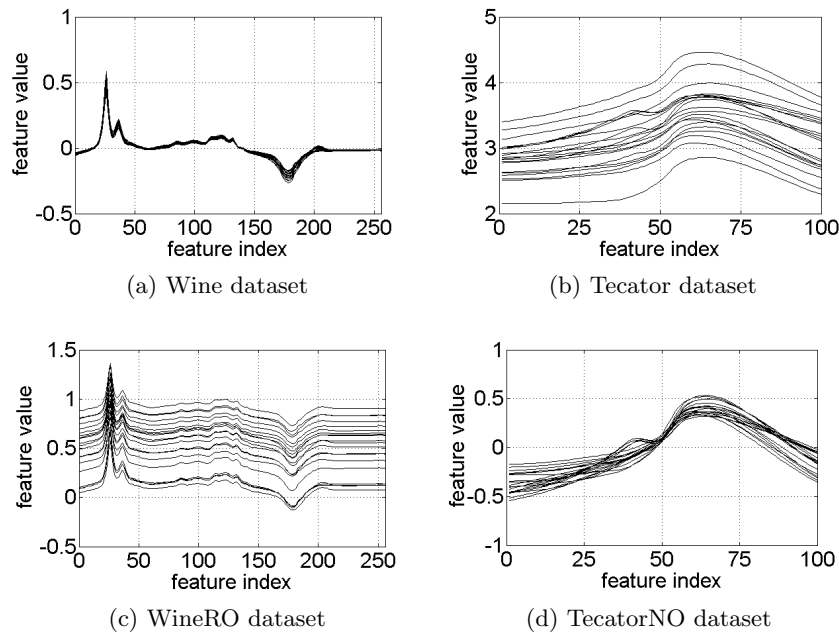


Fig. 1: Input spectra of the different datasets. For the sake of clarity only 20 examples are drawn for each dataset. The presence of offsets in the *WineRO* and *Tecator* datasets is clearly recognizable.

Two experiments are performed for each of the datasets: The first serves as a natural baseline for classification performance and disregards the functional characteristic of the input values entirely by training a GMLVQ system in the original feature space.

The second set of experiments involves a preprocessing of the data in terms of polynomial expansion as described in section 2. For each dataset an approximation is computed with $n = 5, 10, 15, \dots, 50$ polynomial coefficients, resulting in new input feature vectors $\mathbf{c}_i \in \mathbb{R}^n$.

Demonstration code (MATLABTM) for GMLVQ training is available from [1]. The settings and parameters kept constant for all experiments. For most parameters the default values as specified in [1] were used. In detail this means, all trained GMLVQ systems comprised only one prototype per class which were initialized as the class-conditional means in the training set. The relevance matrix was initialized as proportional to the identity and batch gradient descent optimization was performed employing an automated step size control as described in [1, 10]. As an additional preprocessing step the input data underwent a z-score transformation that achieves unit variance and zero mean for all input features. This transformation was done in order to balance varying orders

Table 1: Comparison of processing workflows for experiments with and without incorporation of the functional characteristics of the input data.

	original data	functional approximation
Preprocessing:		polynomial approximation with n coefficients $\mathbf{v}_i \rightarrow \mathbf{c}_i^n$
	z-score transformation $\mathbf{v}_i \rightarrow \mathbf{v}_i^Z$	z-score transformation $\mathbf{c}_i \rightarrow \mathbf{c}_i^{n,Z}$
Training:	Train GMLVQ on 90% of \mathbf{v}_i^Z	Train GMLVQ on 90% of $\mathbf{c}_i^{n,Z}$
Validation:	Validate GMLVQ on remaining 10% of \mathbf{v}_i^Z	Validate GMLVQ on remaining 10% of $\mathbf{c}_i^{n,Z}$

of magnitudes between the different features. Furthermore the transformation facilitates a better interpretation of the resulting relevance matrices [12].

A validation scheme dividing the datasets randomly into 90% training data and using the remaining 10% as a validation dataset is employed for each of the experiments. As a measure for classification performance the area under the ROC (AUROC) is evaluated with respect to the validation set [4]. The ROC is computed by varying a threshold when comparing the distances between the data points and the prototypes. The whole workflow is summarized in Table 1. All results were obtained by averaging over 10 random splits of the data. Figure 2 shows the obtained performance for all datasets in dependency of the number of polynomial approximation coefficients, as well as the performance of the classifier using the raw input data.

To illustrate the advantages of the approach we provide more detail on the *Wine* and *WineRO* datasets in Figure 3. The left-hand panels show the prototypes obtained using a 20 coefficient polynomial approximation. The prototypes are shown in the space of approximation coefficients, center panels display the corresponding relevance profiles. In the right-hand panels, the reconstructed prototypes are shown in the original feature space.

4 Discussion

The results depicted in Figure 2 reveal that the classification performance of the GMLVQ systems trained on the polynomial approximation coefficients are almost identical to or slightly better than the performance when using original data. However, in a polynomial approximation with, say, 20 coefficients, which performs well on all datasets (cf. Fig. 2), the number of input dimensions is drastically decreased by 80% for the *Tecator* datasets and by 92% for the *Wine* datasets.

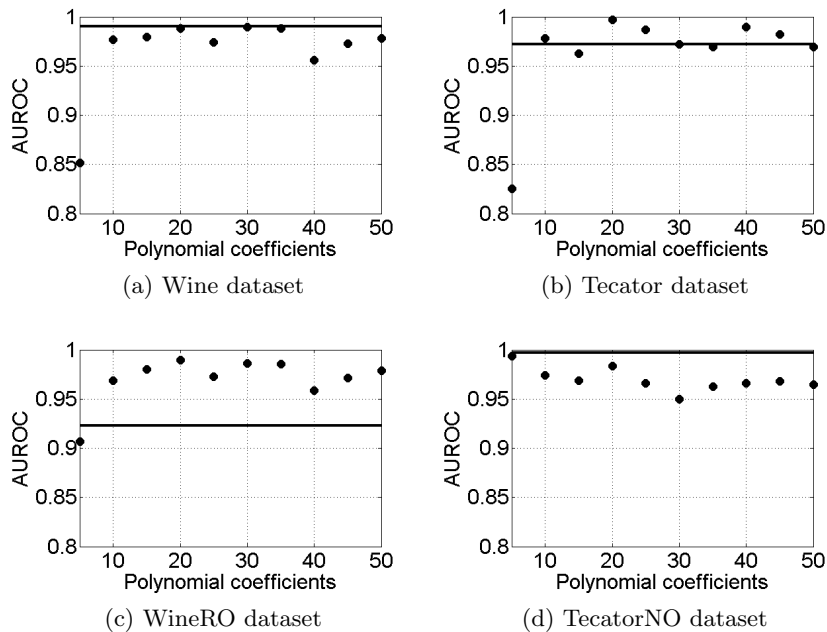


Fig. 2: Comparison of the achieved validation performance, i.e. the area under ROC for different datasets in dependence of the number of polynomial approximation coefficients. The solid line represents the value for the classification using the unprocessed spectral information as feature vectors. Filled circles represent results achieved using polynomial approximation coefficients.

Comparing the performance on datasets with and without artificial distortion, we conclude that the distortion has no significant effect on performance when the polynomial approximation is employed. However, the classification performance with original spectra as input data, is significantly better for the two datasets without offset (*Wine* and *TecatorNO*) than for their counterparts retaining the offsets.

The combination of polynomial approximation and relevance learning is able to suppress the influence of offsets. As shown in figure 3b and 3e the first polynomial coefficient, which represents $T_0(x) = 1$ and can therefore be understood as the constant part of the spectrum, is virtually disregarded by GMLVQ as indicated by a very low value of the corresponding diagonal element of the relevance matrix. Thus, the classification performance for both dataset versions, with and without offset, is nearly the same.

Another benefit of the polynomial approximation is an implicit denoising and smoothing of the data, as can be seen in Figure 3c and 3f. Apart from the (irrelevant) offset, the prototypes are almost identical. The significant smoothing

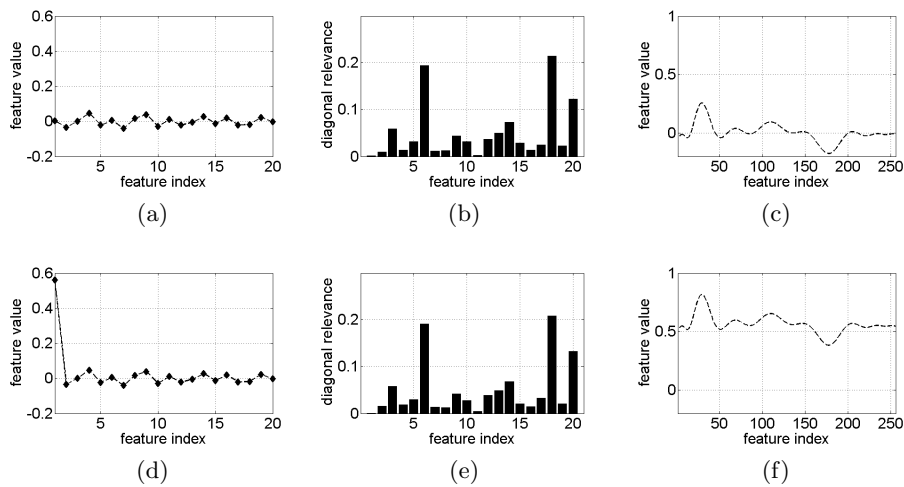


Fig. 3: Detailed comparison of one prototype for the *Wine* and *WineRO* dataset. The top panels (a,b,c) belong to the *Wine* dataset, the bottom (d,e,f) to the *WineRO* dataset. Left-hand panels (a,d) represent prototypes in space of polynomial coefficients, center panels (b,e) represent the relevance profiles obtained and the right-hand panels (c,f) represent the prototypes after retransformation to original feature space.

caused by the polynomial approximation becomes evident when comparing the prototypes to the input spectra in Figure 1.

5 Summary and Outlook

We presented a framework for reducing input dimensions for classification of functional data, by applying polynomial approximation and performing classification in the space of the approximation coefficients. We considered two real world spectral datasets, which were artificially distorted in order to illustrate the advantages of the presented approach. The results show that for a suitable number of polynomial coefficients the resulting classification performance is comparable or exceeds that for unprocessed data. In comparison to [11] using Chebyshev polynomials as basis functions, the number of input dimensions was more significantly decreased by up to 92%, thus drastically reducing the number of parameters, the risk of over-fitting, convergence problems and computational effort.

Furthermore, the robustness of the approach to offset distortion of the data was demonstrated for both example datasets. In forthcoming studies we will address the question whether this independence also holds for more complex distortions, such as different scaling of data or the superposition of trends or

other more complex offsets. Moreover, the intermediate functional representation of the data allows for a more convenient application of mathematical operations such as derivatives, integration or root finding to preprocess the data. These can be used to provide more complex descriptions of the underlying functions, e.g. the number of roots/maxima or maximum slope, to generate even lower-dimensional feature vectors, that can serve as classification input.

Acknowledgments. F. Melchert thanks for the support of an Ubbo-Emmius Sandwich Scholarship from the Faculty of Mathematics and Natural Sciences, University of Groningen.

References

1. Biehl, M.: A no-nonsense beginner's tool for GMLVQ. Available online, University of Groningen, <http://www.cs.rug.nl/~biehl>
2. Biehl, M., Hammer, B., Villmann, T.: Distance measures for prototype based classification. In: Grandinetti, L., Petkov, N., Lippert, T. (eds.) BrainComp 2013, Proc. International Workshop on Brain-Inspired Computing, Cetraro/Italy, 2013. Lecture Notes in Computer Science, vol. 8603, pp. 100–116. Springer (2014)
3. Driscoll, T.A., Hale, N., Trefethen, L.N.: Chebfun guide. Pafnuty Publ. (2014)
4. Duda, R., Hart, P., Stork, D.: Pattern Classification. Wiley (2000)
5. Kästner, M., Hammer, B., Biehl, M., Villmann, T.: Generalized functional relevance learning vector quantization. In: Verleysen, M. (ed.) Proc. Europ. Symp. on Artificial Neural Networks (ESANN). pp. 93–98. d-side (2011)
6. Kohonen, T.: Self-organizing maps. Springer, Berlin (1995)
7. Krier, C., François, D., Rossi, F., Verleysen, M., et al.: Supervised variable clustering for classification of nir spectra. In: Verleysen, M. (ed.) Proc. Europ. Symp. on Artificial Neural Networks (ESANN). pp. 263–268. d-side (2009)
8. Melchert, F., Seiffert, U., Biehl, M.: Functional representation of prototypes in LVQ and relevance learning. Preprint, submitted for publication (2016)
9. Meurens, P.M.: Wine mean infrared spectra dataset. University of Louvain, <http://mlg.info.ucl.ac.be/index.php?page=DataBases>
10. Papari, G., Bunte, K., Biehl, M.: Waypoint averaging and step size control in learning by gradient descent. Machine Learning Reports MLR-06/2011, 16 (2011)
11. Rossi, F., Conan-Guez, B., El Golli, A.: Clustering functional data with the SOM algorithm. In: ESANN. pp. 305–312 (2004)
12. Schneider, P., Biehl, M., Hammer, B.: Adaptive relevance matrices in Learning Vector Quantization. Neural Computation 21, 3532–3561 (2009)
13. Schneider, P., Biehl, M., Schleif, F.M., Hammer, B.: Advanced metric adaptation in Generalized LVQ for classification of mass spectrometry data. In: Proc. 6th Intl. Workshop on Self-Organizing-Maps (WSOM). Bielefeld University (2007), 5 pages
14. Strickert, M., Hammer, B., Villmann, T., Biehl, M.: Regularization and improved interpretation of linear data mappings and adaptive distance measures. In: Computational Intelligence and Data Mining (CIDM), 2013 IEEE Symposium on. pp. 10–17 (April 2013)
15. The Chebfun Developers: Chebfun - Numerical Computing with functions. Available online, University of Oxford, <http://www.chebfun.org>
16. Thodberg, H.H.: Tecator meat sample dataset. StatLib Datasets Archive, <http://lib.stat.cmu.edu/datasets/tecator>

Dissimilarity Extraction in a Median Variant of Learning Vector Quantization

D. Nebel and M. Kaden

Computational Intelligence Group,
University of Applied Sciences Mittweida, Germany
david.nebel@hs-mittweida.de

Abstract. Finding a suitable dissimilarity measure for a given classification problem is a challenging task. It can be very time consuming to learn and validate a classifier for each prospective dissimilarity measure separately. We propose an interpretative classification framework which is based on a linear combination of different dissimilarity measures. Evaluating the weights of this linear combination gives a hint of the most suitable dissimilarity measure. For demonstration purposes we integrate this linear combination into a median variant of the generalized learning vector quantization model. Further, we discuss aspects of dissimilarity learning on a toy example.

1 Introduction

Classification of complex data is a sophisticated task. Recently, investigations show that beside a good classification accuracy other properties of the classifier might be of particular importance [1]. Thus, in many applications the model complexity, the computational effort of (offline) training and (online) test as well as the interpretability of the classification model play an important role. Especially the latter aspect gains in importance in practical applications.

Distance and prototype based classification models, especially Learning Vector Quantizers (LVQ, [7]), are intuitive and mathematically proven methods. Due to their practical success, a lot of extensions and modifications of the fundamental LVQ principle exist: Generalized LVQ (GLVQ, [14]), Robust Soft LVQ (RSLVQ, [17]), Relational (RGLVQ, [4]) and Median GLVQ (MGLVQ, [12]). While most LVQ models require the data to be presented in vectorial form, MGLVQ drops this restriction. Since it is sufficient to provide dissimilarities of all data samples, this method can be used with numerical as well as non-numerical data, for example data with categorical or descriptive features. The prototypes themselves correspond with representative data samples. This leads to an intuitive and interpretable model.

Another interesting aspect of classification is the comparison of data objects. A lot of different dissimilarity measures or distances exist. Yet, to find a suitable dissimilarity measure for a given classification problem is a demanding task. In [11] the authors apply a linear combination of different dissimilarity measures on standard GLVQ. The weights of each distance are obtained using the stochastic gradient descent. This idea is adapted from GLVQ relevance learning (GRLVQ, [5]), where the weights receptively the importance of the single features of the data vectors are learned. Yet, in the approach proposed by [11] the single dissimilarity measures have to be differentiable. We propose a more general

framework based on MGLVQ. To demonstrate the functionality, we apply the new approach on a simple toy example and discuss critical aspects.

2 Dissimilarity Measures

Many classifiers use distances to compare data points. In this paper, we call a dissimilarity measure a *distance* if it is a positive measure and the reflexivity $d(\mathbf{v}, \mathbf{v}) = 0$ is fulfilled [13]. A common choice for comparing two vectors $\mathbf{v}, \mathbf{w} \in \mathbb{R}^{\mathcal{D}}$ is the (squared) Euclidean distance. Yet, in many practical applications the Euclidean distance is not appropriate, e. g. for high dimensional spectral data. A generalization is the l_p -distance (Minkowski distance)

$$d_{l_p}(\mathbf{v}, \mathbf{w}) = \left(\sum_{i=1}^{\mathcal{D}} (v_i - w_i)^p \right)^{\frac{1}{p}} \quad (1)$$

with $p > 0$. For $p = 2$ it is the Euclidean distance d_E .

An extension of the standard squared Euclidean distance is the weighted Euclidean distance

$$d_E^2(\mathbf{v}, \mathbf{w}, \Lambda) = (\mathbf{v} - \mathbf{w})^T \Lambda (\mathbf{v} - \mathbf{w}) \quad (2)$$

with a matrix $\Lambda \in \mathbb{R}^{\mathcal{D} \times \mathcal{D}}$. If Λ is a diagonal matrix the single dimensions are scaled independently. Otherwise, if Λ is the inverse correlation matrix of the data ($\Lambda = \Sigma^{-1}$), (2) is called Mahalanobis distance.

Functional data is a special type of data where neighboring dimensions are highly correlated. Examples are time series or spectra. Distances taking such spatial information into account are the (p)-Sobolev-Distance or the functional distance by Lee&Verleysen [9]. If additionally the data vectors are densities, divergences might be an appropriate choice. For example the set of γ -divergences and the sets of Bregmann- or Csiszar-f-divergences. Common for all divergence families is the Kullback-Leibler-Divergence:

$$d_{KL}(v, \omega) = - \sum_{i=1}^{\mathcal{D}} v_i \cdot \log \left(\frac{v_i}{w_i} \right) \quad (3)$$

which can always be obtained for a particular parameter setting, e. g. for $\gamma \rightarrow 1$ for the γ -Divergence.

Further, general dissimilarity measures are kernel distances. Kernels themselves are inner products of functional Hilbert spaces and therefore similarities. To obtain a dissimilarity we easily perform the transformation [13]:

$$d_{\kappa}^2(\mathbf{v}, \mathbf{w}) = \kappa(\mathbf{v}, \mathbf{v}) - 2\kappa(\mathbf{v}, \mathbf{w}) + \kappa(\mathbf{w}, \mathbf{w}) . \quad (4)$$

The most widely used kernel is the radial basis function kernel (RBF-Kernel)

$$\kappa_{rbf}(\mathbf{v}, \mathbf{w}) = \exp^{-\frac{d_E(\mathbf{v}, \mathbf{w})}{2\sigma^2}} . \quad (5)$$

Other examples are the linear kernel $\kappa_{lin} = \langle \mathbf{x}, \mathbf{w} \rangle_E$, where $\langle \cdot \rangle_E$ denotes the Euclidean inner product, or the sigmoid kernel (SGD-Kernel):

$$\kappa_{sgd}(\mathbf{v}, \mathbf{w}) = \tanh(a \cdot \kappa_{lin} + b) . \quad (6)$$

Due to the nice theory behind them [16], kernel methods are widespread and very popular in practical experiments.

In medical applications the Pearson correlation is often used to obtain a suitable distance measure:

$$d_{corr}(\mathbf{v}, \mathbf{w}) = \frac{\sum_{i=1}^{\mathcal{D}} (v_i - \mu_{\mathbf{v}})(w_i - \mu_{\mathbf{w}})}{\sqrt{\sum_{i=1}^{\mathcal{D}} (v_i - \mu_{\mathbf{v}})^2} \sqrt{\sum_{i=1}^{\mathcal{D}} (w_i - \mu_{\mathbf{w}})^2}} \text{ with } \mu_{\mathbf{x}} = \frac{1}{\mathcal{D}} \sum_{i=1}^{\mathcal{D}} x_i. \quad (7)$$

Beside data samples with numerical features, non-numerical data exists. These data can be texts, musics, or the samples can include categorical and descriptive features. Here, also a lot of distance measures are available. Examples are the Levenshtein-Distance for gene sequences [10] and the Kolmogorov-Complexity for texts or other information based measures [8].

Moreover, a suitable distance measures might be a linear combination of a set of M distances d_k :

$$D_{\lambda}(v, w) = \sum_{k=1}^M \lambda_k d_k(x, w) \quad (8)$$

with positive weights λ_k .

The previously listed distance measures are only a small selection. If any data set is given, the essential question is: Which dissimilarity measure should we chose? This questions is data dependent and no general answer exists. In the following section we extend the Median GLVQ to work with a linear combination of different dissimilarity measures and which learn the weights of this linear combination.

3 Generalized Learning Vector Quantization and a Median Variant

Assume a classification problem with training samples $\mathbf{v} \in V \subset \mathbb{R}^{\mathcal{D}}$ and their class labels $c(\mathbf{v}) \in \mathcal{C}$. Further, a set of $|W|$ prototypes $\mathbf{w}_j \in W \subset \mathbb{R}^{\mathcal{D}}$ assigned to class $y(\mathbf{w}_j) \in \mathcal{C}$ are given. Moreover, we define $d^+(\mathbf{v}) = d(\mathbf{v}, \mathbf{w}^+)$ as the smallest distance between the data point \mathbf{v} and the prototype \mathbf{w}^+ of the set of positive prototypes $W^+ = \{\mathbf{w} | y(\mathbf{w}) = c(\mathbf{v})\}$, i. e. all prototypes belonging to the same class as data point \mathbf{v} . Likewise, $\mathbf{w}^- \in W^- = \{\mathbf{w} | y(\mathbf{w}) \neq c(\mathbf{v})\}$ is the nearest prototype of another class than \mathbf{v} with distance $d^-(\mathbf{v}) = d(\mathbf{v}, \mathbf{w}^-)$.

The GLVQ cost function is given as

$$C_{GLVQ} = \sum_{\mathbf{v} \in V} f(\mu_W(\mathbf{v})) \quad (9)$$

with a monotonically increasing transfer function f and the classifier function

$$\mu_W(\mathbf{v}) = \frac{d^+(\mathbf{v}) - d^-(\mathbf{v})}{d^+(\mathbf{v}) + d^-(\mathbf{v})} \quad (10)$$

[14]. Thereby, the term $d^+(\mathbf{v}) - d^-(\mathbf{v})$ describes the hypothesis margin and the denominator of the classifier function limits the costs [3]. The transfer function f might be the identical function $f(x) = x$ or the sigmoid function $f_{\sigma}(x) = 1/(1 + \exp(-\sigma \cdot x))$ with $\sigma > 0$. If the latter is chosen, (9) approximates the classification error, i. e. for $\sigma \nearrow \infty$ the cost function counts the misclassifications.

The minimization of the in general non-convex and non-linear cost function (9) is usually done by stochastic gradient descent. Therefore, the cost function

has to be differentiable with respect to \mathbf{w}^\pm for a randomly chosen data point \mathbf{v} . The updates yield to

$$\mathbf{w}^\pm \leftarrow \mathbf{w}^\pm - \alpha \Delta \mathbf{w}^\pm \quad (11)$$

$$\text{with } \Delta \mathbf{w}^\pm = \frac{\partial f(\mu_W(\mathbf{v}))}{\partial \mu_W(\mathbf{v})} \cdot \frac{\partial \mu_W(\mathbf{v})}{\partial d^\pm(\mathbf{v})} \cdot \frac{\partial d^\pm(\mathbf{v})}{\partial \mathbf{w}^\pm} \quad (12)$$

and a learning rate $0 < \alpha \ll 1$.

After training, data point \mathbf{v} is assigned to the class of the winner prototype $\mathbf{w}_{s(\mathbf{v})}$ determined by the *winner-takes-all* rule (WTA):

$$s(\mathbf{v}) = \underset{j=1, \dots, |W|}{\operatorname{argmin}} d(\mathbf{v}, \mathbf{w}_j) .$$

Since the prototypes are adapted gradually according to the update rule (12) the dissimilarity measure $d(\mathbf{v}, \mathbf{w})$ has to be differentiable with respect to the prototypes. Yet, in many applications the data samples are given with non-numerical features or even only the dissimilarities between the data are known. In such cases, the standard variant of GLVQ being based on vectorial data samples and prototypes cannot be applied.

Median-GLVQ (MGLVQ) is a method to handle vectorial as well as non-vectorial data objects v . Thereby, the prototypes w_j are restricted to be data samples v and only the dissimilarities $d(v, w_j)$ between the data samples are required.

Analogously to the approach in [12], the GLVQ cost function (9) is translated to a maximization problem

$$C_{MGLVQ} = \log \left(\sum_v g_W(v) \right) \text{ with } g_W(v) = f(-\mu_W(v)) + 1 . \quad (13)$$

The cost functions C_{GLVQ} (9) and C_{MGLVQ} (13) both have the same optima parameter combination, i. e. the log-function influences the value of the optimum but not the choice of the optimal prototypes. The term $+1$ in $g_W(v)$ is added due to numerical reasons to avoid critical values close to zero [12].

The optimization problem of (13) can be solved using the generalized Expectation-Maximization-strategy (gEM) [2]. Therefore, an arbitrary density $\gamma(v)$ is introduced and the cost function (13) is decomposed into two terms [12]:

$$C_{LK} = \sum_{v \in V} (\mathcal{L}_v(\gamma, g_W) - \mathcal{K}_v(\gamma, p_W)) . \quad (14)$$

The second term is the Kullback-Leibler-divergence (KL) of γ and the density

$$p_W(v) = \frac{g_W(v)}{\sum_{v \in V} g_W(v)} \quad (15)$$

and can be calculated as

$$\mathcal{K}_v(\gamma, p_W) = \gamma(v) \log \left(\frac{p_W(v)}{\gamma(v)} \right) . \quad (16)$$

Further, we formally introduce

$$\mathcal{L}_v(\gamma, g_W) = \gamma(v) \log \left(\frac{g_W(v)}{\gamma(v)} \right) \quad (17)$$

$$\text{with } \mathcal{L} = \sum_{v \in V} \mathcal{L}_v(\gamma, g_W) \quad (18)$$

as the lower bound.

The optimization problem (14) can be interpreted as a maximum likelihood problem [2], which can be solved by a variant of the gEM-approach [12]. It results in the following steps:

MGLVQ-Algorithm:

1. Initialize W
2. **E-step** set $\gamma_W(v) \leftarrow p_W(v)$ and therefore, it yields $\mathcal{K}_v(\gamma, p_W) = 0$
3. **M-step** fix $\gamma_W(v)$ and determine a prototype set W such that the lower bound \mathcal{L} (18) is improved
4. *if* no new prototype set W is found in M-step: stop
else: go to E-step

In general, the MGLVQ ends up with an interpretative sparse classification model. Experiments show that in many cases the requirement of the prototypes to be data samples is not a significant restriction according to the classification error [12].

4 Dissimilarity Learning in Median GLVQ

In section 2 it is pointed out that a lot of dissimilarity measures exist. To determine a suitable measure for a given classification problem is a tricky task.

We adapt the idea of [11] and integrate a linear combination of distances $D_\lambda(v)$ in the MGLVQ. The cost function of the dissimilarity MGLVQ (DMGLVQ) is identical to C_{MGLVQ} (13), but the classifier function $\mu_W(v)$ depends on the linear combinations:

$$\hat{\mu}_W(v) = \frac{D_\lambda^+(v) - D_\lambda^-(v)}{D_\lambda^+(v) + D_\lambda^-(v)} \quad \text{with } D_\lambda^\pm(v) = \sum_{k=1}^M \lambda_k d_k^\pm(v). \quad (19)$$

The optimization of the prototype assignments is analogously to MGLVQ. Yet, a Metric-Step is added after the M-step in the *MGLVQ-Algorithm*. The Metric-Steps optimize the weights λ_k for a given prototype set. Unfortunately, a direct optimization of the non-convex and non-linear cost function is not possible. However, applying the stochastic gradient descent yields a suitable solution. The updates are:

$$\begin{aligned} \lambda_k &\leftarrow \lambda_k + \alpha_\lambda \Delta \lambda_k \\ \text{with } \Delta \lambda_k &= \frac{\partial f(\hat{\mu}_W(v))}{\partial \hat{\mu}_W(v)} \cdot \left(\frac{\partial \hat{\mu}_W(v)}{\partial D_\lambda^+(v)} \cdot \frac{\partial D_\lambda^+(v)}{\partial \lambda_k} + \frac{\partial \hat{\mu}_W(v)}{\partial D_\lambda^-(v)} \cdot \frac{\partial D_\lambda^-(v)}{\partial \lambda_k} \right) \\ &= \frac{\partial f(\hat{\mu}_W(v))}{\partial \hat{\mu}_W(v)} \cdot \frac{4\lambda_k}{(D_\lambda^+(v) + D_\lambda^-(v))^2} \cdot (D_\lambda^+(v)(d_k^-(v))^2 - D_\lambda^-(v)(d_k^+(v))^2) \end{aligned}$$

and $0 < \alpha_\lambda \ll 1$. Note that we drop the log-function of the cost function to determine the update-rules. Since the log-function is monotonically increasing, it has no influence on the choice of the optimal prototypes.

Fortunately, the update of λ does not depend on the derivatives of the distances d_k . Thus, we can apply the DMGLVQ still on non-vectorial data sets or with non-differential dissimilarity measures.

5 Experiments

The toy data set consists of two two-dimensional Gaussian distributed overlapping classes. The number of data points per class is 1000. We perform a 5-fold cross-validation including a validation set, i. e. the data set is split in five folds with three folds for training, one fold for test and one fold for validation. Overall, we end up with 20 runs. For all runs we provide one randomly initialized prototype per class.

The following dissimilarity measures are chosen: Euclidean distance d_E , l_1 -distance d_{l_1} , RBF-Kernel d_{rbf}^2 with $\sigma = [0.01, 1]$, Sigmoid kernel d_{sgd}^2 with $a = 0.01, b = 1$, the Pearson correlation d_{corr} and a random symmetric matrix with zeros on the diagonal d_{ran} . To obtain the same range of all distances, we normalize each matrix by dividing by the maximum value of the matrix. The corresponding dissimilarity matrices of the data samples are displayed in Fig. 1.

First, we run the MGLVQ and obtained a validation accuracy of around 86% for five of the seven distances measures (see Tab. 5). The RBF-kernel with $\sigma = 1$ and obviously the random *dissimilarity* fail. Thus, five distances are able to separate the data, i. e. separation information is included in five of the seven dissimilarity matrices.

In the second step the DGMLVQ with a linear combination of the seven dissimilarity measures is trained. The overall accuracy of 86.1% is similar to the single accuracies. Hence, in this classification problem a linear combination does not bring any benefit.

Interestingly, the correlation distance d_{corr} wins in the first run of DMGLVQ since the mean weighting value is equal one (see Tab. 2). To get a ranking of the applicability of the other distances, in the following steps we always dropped the distance with the highest weighting value λ_k . Therefor in the second step we dropped the Pearson correlation and run the DMGLVQ with a linear combination of the remaining six distances. Now, the Euclidean distance as well as the Sigmoid-Kernel distance are nearly weighted equally, i. e. $\lambda_k^2 = 0.493$ for Euclid and $\lambda_k^2 = 0.489$ for the Sigmoid-Kernel distance. Due to the slightly higher value of the Euclidean distance we dropped it and in the next run obviously the SGD-Kernel wins. This way we continued to drop the SDG-Kernel distance, the l_1 distance and the RBF-Kernel distance with $\sigma = 0.01$ in the following steps. Up to now the mean accuracies are only slightly decreasing. Finally, only the RBF-Kernel with $\sigma = 1$ and the random matrix, leading to a random class separation, are remaining. Thus, the DMGLVQ is doing its job well because the trained models point out the suitable distances. Of course, this is only a little toy example but it demonstrates the principle of metric extraction of this new approach.

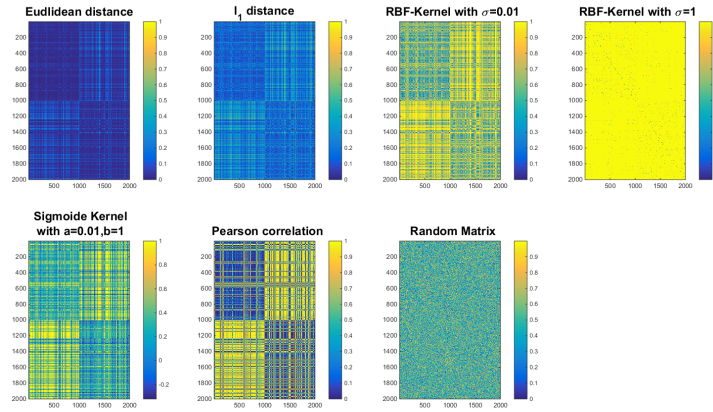


Fig. 1. Depicted are seven dissimilarity matrices of the data samples. The data samples are sorted, i. e. the first 1000 samples belong to the first class and the others belong to the second class. The structure of the dissimilarity matrices of d_E , d_{l1} , d_{krbf} with $\sigma = 0.01$, d_{ksgd} , and d_{corr} are very similar. The matrix of d_{krbf} with $\sigma = 1$ and the random matrix have different structures.

	MGLVQ							DMGLVQ
distances	d_E	d_{lp} $p = 1$	d_{krbf}		d_{ksgd}	d_{corr}	d_{rand}	all
			$\sigma = 0.01$	$\sigma = 1$	$a = 0.01, b = 1$			
accuracy	86.7%	84.5%	86.6%	72.7%	86.2%	86.4%	49.1%	86.1%
std dev.	$\pm 2.3\%$	$\pm 2.4\%$	$\pm 2.5\%$	$\pm 3.2\%$	$\pm 2.3\%$	$\pm 2.7\%$	$\pm 3.3\%$	$\pm 2.5\%$

Table 1. Mean accuracies with standard deviation of the MGLVQ for each dissimilarity d_k and the DMGLVQ applied on the linear combination D_λ of all seven dissimilarities.

steps	λ_k^2							accuracy
	d_E	d_{lp}	d_{krbf}		d_{ksgd}	d_{corr}	d_{rand}	
			$p = 1$	$\sigma = 0.01$	$\sigma = 1$	$a = 0.01, b = 1$		
1	0.007	0	0	0	0	0.993	0	$86.1\% \pm 2.5$
2	0.493	0.02	0	0	0.489	-	0	$85.7\% \pm 2.2$
3	-	0.33	0	0	0.70	-	0	$85.0\% \pm 2.5$
4	-	1	0	0	-	-	0	$84.7\% \pm 1.2$
5	-	-	1	0	-	-	0	$83.7\% \pm 1.1$
6	-	-	-	0.3	-	-	0.7	$50.6\% \pm 3.0$

Table 2. Mean weighting values λ_k^2 and accuracies of the DMGLVQ for the different runs removing the distance with the highest value λ_k by and by.

6 Conclusion and Future Work

We demonstrate a way of distance learning in a median variant of GLVQ. Based on several dissimilarity matrices of the data points, we propose an approach to learn the weights of each dissimilarity measures influencing the classification task. The weights directly correspond to the importance respectively suitability of the distance measure. Due to the restriction that the prototypes are data points, the resulting model is interpretable independently of the distance measure. A first toy example demonstrates the functionality of the novel method.

Future work should include the application on a real world data set with a detailed analysis of the result. Like in [6] or [11] a linear combination of different distances might bring an improvement.

Furthermore, investigations to the training of a classification correlation matrix known from the Generalized Matrix Learning Vector Quantization [15] is ongoing and results have to be analyzed.

References

1. A. Backhaus and U. Seiffert. Classification in high-dimensional spectral data: Accuracy vs. interpretability vs. model size. *Neurocomputing*, 131:15–22, 2014.
2. C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
3. K. Crammer, R. Gilad-Bachrach, A. Navot, and A. Tishby. Margin analysis of the LVQ algorithm. In S. Becker, S. Thrun, and K. Obermayer, editors, *NIPS 2002*, volume 15, pages 462–469, Cambridge, MA, 2003. MIT Press.
4. B. Hammer, D. Hofmann, F. Schleif, and X. Zhu. Learning vector quantization for (dis-)similarities. *Neurocomputing*, 131:43–51, 2014.
5. B. Hammer and T. Villmann. Generalized relevance learning vector quantization. *Neural Networks*, 15:1059–1068, 2002.
6. U. Knauer, A. Backhaus, and U. Seiffert. Fusion trees for fast and accurate classification of hyperspectral data with ensembles of γ -divergence-based RBF networks. *Neural Computing and Applications*, 26(2):253–262, 2015.
7. T. Kohonen. Learning vector quantization for pattern recognition. *Technical Report*, TKK-F-A601, 1986. Helsinki University of Technology.
8. A. Kolmogorov. On tables of random numbers. *Sankhya: The Indian Journal of Statistics. Ser. A*, 25:369–375, 1963.
9. L. Lee and M. Verleysen. Generalization of the l_p norm for time series and its application to self-organizing maps. In M. Corrtell, editor, *Proc. of Workshop on Self-Organizing Maps (WSOM)*, pages 733–740, Paris, Sorbonne, 2005.
10. V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Doklady Akademii Nauk SSSR*, 163(4):845–848, 1965.
11. E. Mwebaze, G. Bearda, M. Biehl, and D. Zühlke. Combining dissimilarity measures for prototype-based classification. In *23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2015.
12. D. Nebel, B. Hammer, and T. Villmann. A median variant of generalized learning vector quantization. In *ICONIP 2013*, pages 19–26, 2013.
13. E. Pekalska and R. P. Duin. *The Dissimilarity Representation for Pattern Recognition. Foundations and Applications*. World Scientific, Singapore, December 2005.
14. A. Sato and K. Yamada. Generalized learning vector quantization. In H. M. Touretzky DS, Mozer MC, editor, *Advances in neural information processing systems*, volume 8, pages 423–429. MIT Press, Cambridge, 1996.
15. P. Schneider, B. Hammer, and M. Biehl. Distance learning in discriminative vector quantization. *Neural Computation*, 21:2942–2969, 2009.
16. B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.
17. S. Seo and K. Obermayer. Soft learning vector quantization. *Neural Computation*, 15:1589–1604, 2003.

Towards Dimensionality Reduction for Smart Home Sensor Data

Bassam Mokbel, Alexander Schulz

Bielefeld University, CITEC Center of Excellence
Bielefeld, Germany
{bmokbel|aschulz}@techfak.uni-bielefeld.de

Abstract. In this paper¹, we investigate in how far nonlinear *dimensionality reduction* (DR) techniques can be utilized to tackle particular challenges of sensor data from smart home environments. Smart homes often contain a large number of sensors of various types, providing output in real time, which results in a sequence of high-dimensional, heterogeneous data vectors. We propose that DR techniques can provide a truthful low-dimensional representation (i.e. a compression) of this kind of data, together with a corresponding reconstruction (i.e. decompression). This yields an automatic fusion of uncoordinated raw sensor signals, as well as an economical storage format, with a certain robustness against sensor failure. In proof-of-concept experiments, we present first empirical results to test our approach based on real-world data.

1 Introduction

Motivation In recent years, the topic of *smart homes* gained major attention in the electronics and electrical appliances industry. The term describes environments, in which a collection of different sensors and individual devices are utilized for home automation, for assisted living in healthcare situations, or other interactive home scenarios to provide convenience and intelligent assistance in everyday life. In this context, efficient data mining and machine learning algorithms play a prominent role, in order to adapt these systems to the inhabitants and their environment, based on given sensor data. For example, smart devices may learn the daily habits of a user, and infer rules to detect the corresponding living situations automatically.

Data in smart home environments are typically characterized by temporal streams of heterogeneous, high-dimensional sensor readings as well as preprocessed sensor evaluations, e.g. temperature sensors, motion sensors, or cameras providing face detection and the location of inhabitants. In this context, we expect the following challenges for machine learning:

- (I) *High-dimensional data* Given the increasing ubiquity and precision of sensors, together with the limited memory capacity in typical home automation devices, it becomes challenging to analyze the output of all available sensors at once, and to store a sufficient amount of historical data in order to apprehend an adequate temporal context for analysis tasks.
- (II) *Heterogeneous data* At the same time, the large variety in the output data from different sensor types typically requires hand-tuning and expert knowledge to set up a successful machine learning system for the given data. Thereby, raw sensor readings are preprocessed

¹ Funding from DFG under grant number HA2719/7-1 and by the Cluster of Excellence Cognitive Interaction Technology ‘CITEC’ (EXC 277) at Bielefeld University is gratefully acknowledged.

and meaningfully represented in accordance with other sensor output. It would be beneficial to avoid costly human effort for such problem-specific adaptations.

(III) *Uncoordinated sensors* Similarly, it can be expected that the available collection of sensors and sensor modalities is very diverse in general, and may even be flexible over time: home environments are different in shape and size, so the technical configuration will also vary, while sensors may dynamically change or possibly fail. Therefore, a hand-crafted organization of sensors to gain higher-level abstract data representations seems no longer appropriate. Rather, machine learning techniques will have to deal with loose collections of sensors, which provide individual local views of the environment and may yield missing values.

Nonlinear dimensionality reduction Regarding these three challenges, the aspect of ‘high-dimensional data’ can be addressed directly by *dimensionality reduction* (DR) techniques: With a large collection of sensors available in a smart home, it is expectable that several signals will be correlated, and the intrinsic dimensionality is actually lower than the number of sensor outputs. Therefore, DR is a promising tool to establish a low-dimensional representation from the original sensor data, which is easier to handle in terms of storage space and complexity for data analysis and machine learning algorithms.

DR has been an emergent research topic over the last decade, with successful applications in a variety of scientific and industrial fields, especially for nonlinear DR methods. Recent advances include novel acceleration techniques for big data sets [11,4], as well as principled evaluation and parameterization approaches [7,6]. Hence, there are many DR algorithms readily available, see [8,12] for overviews. Some existing work in the literature demonstrates the general benefit of DR also in the context of sensor data: for example, linear DR [13], and sparse coding techniques [1] were used to achieve compressed representations from sensor signals.

However, given the heterogeneous characteristics of sensor data in smart homes, as described in our challenge (II), we can expect that correlations are usually not linear. Therefore, linear DR may fail to capture the relevant information in the data, unless problem-specific preprocessing steps (including nonlinear transformations) are established by human experts. Hence, we believe that nonlinear DR techniques would be more appropriate, which has not yet been sufficiently investigated in current literature.

There exist successful applications of manifold-based nonlinear DR techniques like the *self-organizing map* (SOM) [2] or *Isomap* [5] for spatio-temporal sensor data. However, those approaches realize a rather restrictive embedding: while the SOM depends heavily on the a-priori choice of a lattice structure for the embedded points, Isomap requires an appropriate neighborhood graph in the original data space. This makes both methods less flexible w.r.t. unseen data. Regarding challenges (II) and (III), we believe that the generalization ability of a DR embedding is a key ingredient for treating smart home data, and should be investigated further. Modern high-performance nonlinear DR techniques have proven to be successful in terms of generalization for biological data, see [3]. Therefore, these approaches seem like a viable alternative to tackle the challenges of smart home data. In this contribution, we present first empirical results by testing the t-SNE [12] method in particular, together with a recent extension by a kernel mapping approach [4], using a small data set of raw sensor signals for proof-of-concept experiments.

2 Dimensionality reduction for smart home data

The goal of DR is to find low-dimensional (low-D) vectors $\{\mathbf{y}_1, \dots, \mathbf{y}_N\} = Y \subset \mathbb{R}^L$, which resemble the structure of the original high-dimensional (high-D) data set $\{\mathbf{x}_1, \dots, \mathbf{x}_N\} = X \subset \mathbb{R}^H$.

The dimensionality L of the embedding space \mathbb{R}^L is defined a priori by the user, and is often chosen as 2 or 3 for the benefit of visualizing the embedded points in a scatter plot. Many nonlinear DR techniques optimize the low-D vector locations such that their neighborhoods resemble the neighborhood structure in the original data w.r.t. certain criteria of distance preservation. If the data’s intrinsic dimensionality is higher than L , not all pairwise distances between the data can be represented accurately in the embedding, and a certain loss of information is inevitable. To assess how truthful a given embedding represents the original high-D data, there exist independent quality criteria to evaluate the reliability of their low-D counterparts, see e.g. [9,10].

Typically, nonlinear DR methods emphasize the preservation of local neighborhoods in favor over a faithful reproduction of larger distances. In this work, we use the well-established nonlinear DR technique *t-distributed stochastic neighbor embedding* (t-SNE) [12], which aims to minimize the discrepancy between neighborhood probability distributions in the original, and the embedding space, as measured by the Kullback-Leibler divergence. The approximate size of the considered neighborhood can be controlled explicitly via the *perplexity* parameter in t-SNE. One limitation of t-SNE (and most other nonlinear DR techniques) is that the embedding Y is obtained by an optimization procedure based on the given data X , but there is no functional form available for the mapping. In case of time series as they occur in smart homes, this has certain downsides: On the one hand, we require an explicit function to add unseen data (e.g. incoming sensor readings) seamlessly to the existing low-D representation of previous data, i.e. an embedding function of the form $f : \mathbb{R}^H \rightarrow \mathbb{R}^L$ with $\mathbf{x}_i \mapsto f(\mathbf{x}_i) = \mathbf{y}_i$ for all $i = \{1, \dots, N\}$. On the other hand, we may see the embedding as a compressed data representation, which allows us to store and handle all \mathbf{y}_i as a compact alternative to \mathbf{x}_i . Hence, we also demand a corresponding ‘decompression’ function, by which we can reproduce original sensor signals from their low-dimensional counterparts, i.e. an inverse mapping $g : \mathbb{R}^L \rightarrow \mathbb{R}^H$ which approximately recovers high-D points $\tilde{\mathbf{x}}_i \in \mathbb{R}^H$ with $\mathbf{y}_i \mapsto g(\mathbf{y}_i) = \tilde{\mathbf{x}}_i \approx \mathbf{x}_i$ for all $i = \{1, \dots, N\}$.

To achieve these desired properties, we refer to a recent extension called *kernel t-SNE* [4]. This method uses the original high-D points together with their low-D counterparts embedded by the regular t-SNE, and establishes a mapping between the two via regression. Therefore, a generalized linear mapping is assumed, with parameters α and an appropriate basis function, in our case a normalized Gaussian kernel of bandwidth σ , see [4] for details. This addresses both of our requirements. On the one hand, we can train the kernel mapping to establish an embedding function $f(\mathbf{x}) = \mathbf{y}$ for unseen data \mathbf{x} . On the other hand, we can obtain an inverse function $g(\mathbf{y}_i) = \tilde{\mathbf{x}}_i \approx \mathbf{x}_i$ for reconstructing high-D points from low-D inputs, by using the same learning scheme in the opposite manner: the kernel mapping is then trained for given low-D vectors with their high-D counterparts as regression targets.

3 Experimental evaluation

Application background & technical setting In our experiments, we use real world sensor data from “*The Cognitive Service Robotics Apartment as Ambient Host*” project (CSRA) at the CITEC research facility in Bielefeld, Germany. The CSRA project involves an intelligent apartment for research purposes, where multiple sensors constantly monitor various conditions of the environment, as well as human interactions in each room. The apartment consist of a living room with an adjacent kitchen area, a small entrance hallway, and a fitness room, each holding appropriate furniture and home appliances. Details can be found on the project’s website².

² <http://cit-ec.de/en/content/cognitive-service-robotics-apartment-ambient-host>

We are addressing two modalities of sensor readings, both producing multi-dimensional data streams in real-time. With a rather small dimensionality and limited complexity of this data, we are not covering all three challenges from Section 1 to the full extent with our experiments. However, we believe that the data represents all three aspects to some degree, and our results may serve as a first proof-of-concept.

In the first modality, we use 3-dimensional person tracking data from three different overhead camera systems: These depth-image cameras are installed in the ceiling at the entrance, the kitchen, and the living room. Each camera preprocesses the depth image to yield a person location hypothesis in its local 3D coordinate system, whenever a person is detected in the camera's field of view. Since the fields of view of adjacent cameras are overlapping, a person hypothesis may be present in several local coordinate systems at a time, in case a user is moving through the overlapping areas. Note, that these depth cameras are low-cost devices similar to the Microsoft Kinect model, which seems realistic for person tracking in future home automation systems. Additionally, the fact that the cameras yield separate local tracking results (rather than unified spatio-temporal data in a global coordinate system) is in line with our premise of 'uncoordinated sensors' from Section 1. From all three cameras with individual 3D coordinate systems, we gain 9 real-valued data dimensions in each time step.

The second modality comes from a SensFloor[®] tactile floor mat, which is installed on the kitchen ground. It yields continuous pressure levels for 8 individual segments which are evenly partitioned in a clockwise fashion. Like in the case of a single camera, we can expect changing values only when a person is moving in the kitchen area. With these 8 pressure sensors, and the 9-dimensional data from person tracking, we are addressing 17 real-valued data dimensions in total. Since the distribution and correlation of values in the pressure sensors are much different from the camera tracking output, our scenario involves the challenge of 'heterogeneous data'.

In the following, we will address three distinct research problems in the context of smart home data, which relate to the challenges in Section 1. For each problem, we will present experimental results from our described setting. The data for our experiments consist of two sequences, in which a single person walks through the apartment in an exploratory manner. The duration of *Sequence A* is 86 seconds, while *Sequence B* lasts 50 seconds. In both sequences, the person enters the field of view of every camera at least once, and is walking over the tactile floor panels briefly.

Each sequence is a time series $(\mathbf{x}_1, \dots, \mathbf{x}_N)$ of 17-dimensional vectors $\mathbf{x}_i \in \mathbb{R}^{H=17}$. Every vector represents the sensor values provided within a 50 millisecond time frame. We applied only general, fairly simple preprocessing steps, which are not specific to the given scenario or sensor characteristics: If a sensor provides missing values (e.g. when a camera does not detect any person), we continue to write the last known values in the subsequent vectors for each following time step, until the sensor delivers values again. Next, every vector, where no change to its predecessor is observed, was removed from the sequence, in order to avoid many zero distances between distinct data points. To avoid strong discontinuities between subsequent vectors, we additionally smoothed³ the time series by setting each vector entry $[\mathbf{x}_i]_d$ to the mean value of corresponding entries in a window of 10 predecessors and successors. (This translates to 1 second in the original time series, if no steps were removed from the sequence.) Lastly, we used a z-score transformation for the whole sequence to balance the numeric representation of all dimensions.

³ Note that this common preprocessing step is not problem-specific. However, it is necessary, since discontinuities in the data can lead to clustered low-D representations (in our case due to the pressure sensors in particular). This would yield an insufficient basis for training robust kernel mappings.

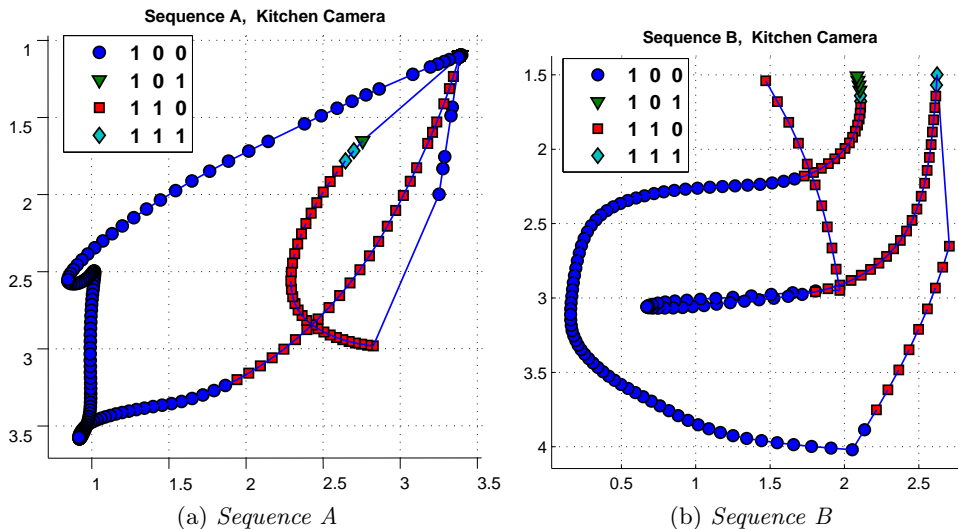


Fig. 1: Top-down view of the person tracking coordinates from the kitchen camera. The symbols indicate which of the three cameras is tracking the person at that time point.

As an example, the person tracking results from the kitchen camera are displayed in a top-down⁴ plot in Figure 1. A line connects the points in the order of their temporal progression. The symbolic labels indicate whether each of the three cameras is tracking the person at that time: the binary code signifies the tracking status of the first camera (in the kitchen), the second (in the living room), and the third (in the entrance area), e.g. “1 0 1” means that the kitchen camera *is* tracking the person, while the camera in the living room is *not* tracking, and the entrance camera *is* tracking as well⁵.

Problem 1 – low-dimensional representation Our first question is, whether we can obtain a reliable low-dimensional representation of the time series data, without a problem-specific preprocessing of the given sensor signals. Specifically, can we observe a rather smooth temporal trajectory in the embedding, despite the given heterogeneous multi-dimensional sensor data? This relates mainly to challenge (II) from Section 1.

We applied the t-SNE method with a target dimensionality of $L = 3$, for both given sequences A and B. Our assumption is that a 3-dimensional embedding can represent the original data rather truthfully, since only a single person is moving through the spatio-temporal sensor system. Since t-SNE relies on a non-convex optimization starting from a random initialization, we ran the method several times and evaluated the reliability of every embedding in terms of the quality criterion Q_{NX} , as proposed in [9]. Additionally, we evaluated the quality index $Q_{\text{NX}}^{x_i}$ for every single point, as proposed in [10], which yields the ratio of preserved neighbors in a point’s k -neighborhood. We chose the best respective 3D embedding in terms of this evaluation; the result for each sequence is presented in Figure 2.

In both embeddings, we can see that the time series is represented as a rather smooth trajectory, with only a few positions where the sequence is torn apart. The average quality $Q_{\text{NX}}^{x_i}$ over all points is 91.6% for A and 91.9% for B, in a $k = 10$ neighborhood. This substantiates our positive visual assessment. From these results, we can conclude, that DR can actually produce

⁴ Note that the tracking data is in fact 3D, however, the depth values follow a standard pattern of decreasing with the distance to the camera’s center, and are thus omitted in this top-down perspective.

⁵ Since we display the particular field of view of the kitchen camera in Figure 1, the first bit of the points’ labels is always 1 in these plots.

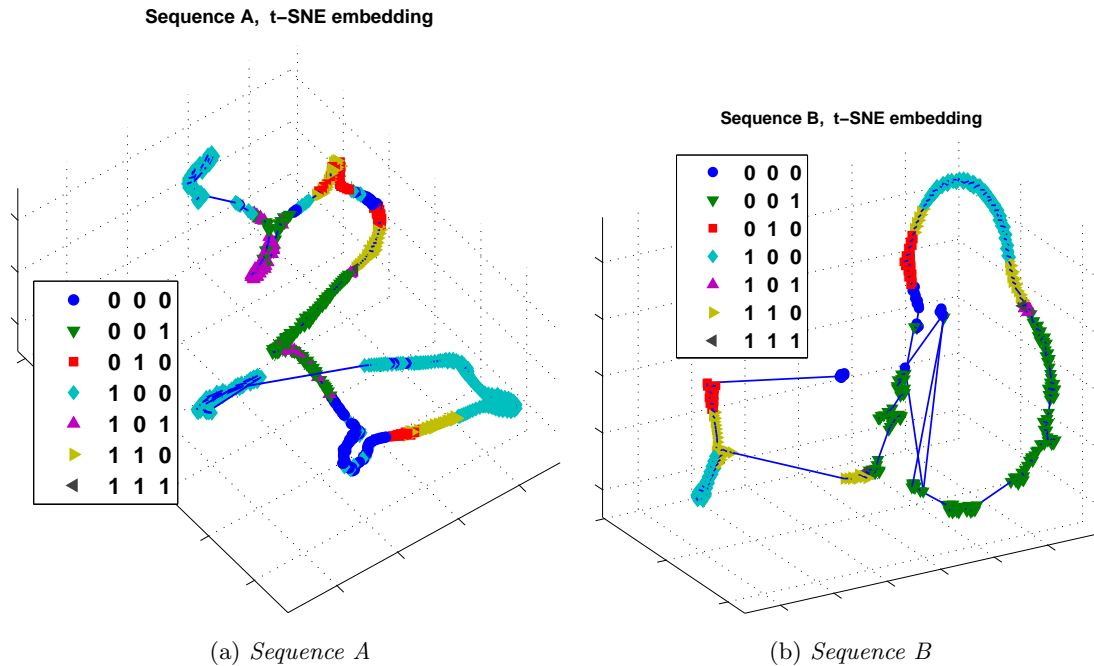


Fig. 2: 3D embedding obtained by the t-SNE method for both 17-dimensional sequences. The colored symbols indicate which of the three cameras is tracking the person at that time. The temporal progression of the sequence is mostly captured by a smooth trajectory in 3D.

a fairly accurate 3D embedding of the 17-dimensional original sensor data, even without hand-crafted preprocessing steps. The low-D representation yields a homogeneous representation of heterogeneous sensor signals.

Problem 2 – explicit parametric mapping & inverse Is it possible to obtain an explicit embedding function, which generalizes to unseen data, as well as an inverse mapping, which allows for an approximate reconstruction of the original sensor signals from the compressed low-dimensional representation? This relates mainly to challenge (I) from Section 1, since it would enable a compression and reconstruction of high-D sensor signals for more efficient storage and handling, without too much loss of information.

With the kernel t-SNE method, we now train a kernel mapping f based on the embeddings in Figure 2. To measure the discrepancy between the original embedding and the result of our trained mapping function, we can simply calculate the mean squared error (MSE) between both sets of vectors. We evaluate the generalization ability by averaging the training and test errors for random splits of the data over a 5-fold cross-validation with 10 repeats. After meta-parameters have been adjusted to the given data, an acceptable average error for both embedded sequences was observed, although with rather high standard deviations (std.): For Sequence A, we achieved a mean MSE of 9.5 (0.5 std.) on the training, and 31.1 (11.2 std.) on the test set. For Sequence B, the errors were generally lower with a mean MSE of 1.4 (1.2 std.) on the training, and 10.4 (2.0 std.) on the test set. From the experiment, we observed that the kernel mapping was often not able to capture the few discontinuous parts of the embedded trajectory. The other parts were reproduced more truthfully by the learned mapping. However, the training behavior in general tended to overfit. This may be due to the fact that the input data contains many degrees of freedom which are mapped to considerably less variables in the output.

Instead, training an inverse mapping function g was much easier in the experiments. We again used a repeated 5-fold cross-validation, which resulted in very low errors in general. The mean MSE for Sequence A was 0.007 (0.001 std.) on the training, and 0.017 (0.005 std.) on the test set.

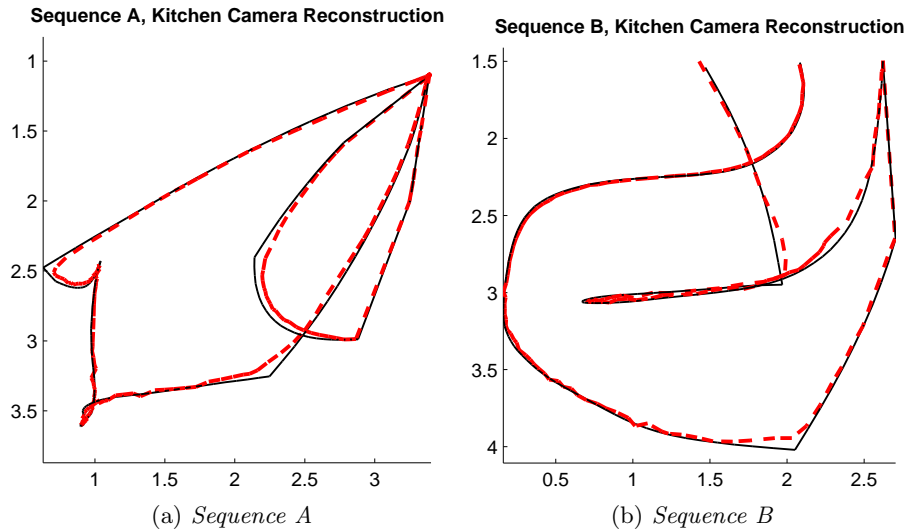


Fig. 3: The original person tracking data from the kitchen camera (the thin black line), together with the reconstruction (the red dashed line) based on a 3D embedding of the respective 17-dimensional sequence.

For Sequence B, we achieved 0.037 (0.006 std.) MSE on the training, and 0.741 (0.883 std.) on the test set. We can exemplify these very good results by visualizing the respective reconstructions of the person tracking data from Figure 1: In the Figure 3, we show the original person tracking from the kitchen camera, together with its reconstruction via the inverse mapping function for both sequences. In each case, the original trajectory is resembled rather smoothly, which verifies our quantitative evaluation showing very high performance on the test set.

We can conclude that training an accurate forward mapping function $f(\mathbf{x}_i)$ is harder than training the inverse mapping $g(\mathbf{y}_i)$. Therefore, the idea of using a DR technique for compressing sensor signals can be a viable option to handle smart home data efficiently, since we are able to reconstruct the approximate original signals via a trained kernel mapping $g(\mathbf{y}_i)$. However, the idea to use a kernel mapping $f(\mathbf{x})$ to extend the DR embedding to unseen data seems less robust with the techniques we applied, and needs further refinement.

Problem 3 – robustness against sensor failure Can we utilize the learned embedding $f(\mathbf{x})$ and reconstruction function $g(\mathbf{y}_i)$ to fill missing sensor readings with plausible values? This relates to challenge (III) from Section 1, since it would signify that the compressed representation is robust in a dynamically changing technical environment.

Therefore, we chose an arbitrary 30% time window in Sequence B, and manipulated the original data by filling one dimension with erroneous values. We replaced the y-coordinate of the entrance camera in that time frame with values that simulate a failing sensor: small Gaussian noise around the average output of the y-coordinate over the given sequence. Figure 4a shows the manipulated camera data. We then used the previously trained mapping f to embed these erroneous data vectors in 3D, and applied the trained inverse function g to reconstruct the original sensor signals. The result showed the interesting effect of reconstructing the original trajectory nearly perfectly, see Figure 4b. Hence, the trained kernel mappings seem robust against local failures of single sensors, and can help to fill missing values in dynamic sensor environments.

Conclusion Our preliminary results show, that modern nonlinear DR techniques offer promising features to tackle the challenges of smart home sensor data. Trained kernel mappings are able to capture the characteristics of our example data and did generalize well. Our ongoing research will include a comparative study with other DR methods (e.g. sparse coding).

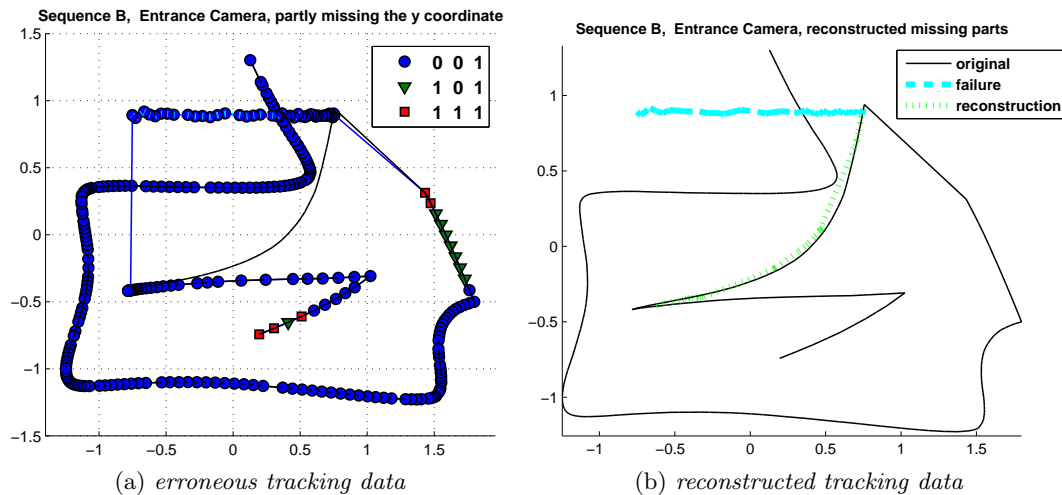


Fig. 4: Top-down plots of the person tracking data from the entrance camera in Sequence B, where parts of the y-coordinate were replaced by its mean value to simulate a sensor failure. On the left, the manipulated input data is shown, where the noisy/failing part is highlighted in cyan and the original trajectory is the black line. On the right, the kernel mapping was used to successfully reconstruct the original tracking data, where the reconstruction is drawn in green.

References

1. R. Calderbank, S. Jafarpour, and R. Schapire. Compressed learning: Universal sparse dimensionality reduction and learning in the measurement domain. Technical report, 2009.
2. E. Catterall, K. Van Laerhoven, and M. Strobbach. Self-organization in ad hoc sensor networks: An empirical study. In *Proceedings of the Eighth International Conference on Artificial Life, ICAL 2003*, pages 260–263, Cambridge, MA, USA, 2003. MIT Press.
3. A. Gisbrecht, B. Hammer, B. Mokbel, and A. Sczyrba. Nonlinear dimensionality reduction for cluster identification in metagenomic samples. In *IV 2013*, pages 174–179, 2013.
4. A. Gisbrecht, A. Schulz, and B. Hammer. Parametric nonlinear dimensionality reduction using kernel t-sne. *Neurocomputing*, 147:71–82, 2015.
5. O. C. Jenkins and M. J. Matarić. A spatio-temporal extension to isomap nonlinear dimension reduction. In *Proceedings of the Twenty-first International Conference on Machine Learning, ICML '04*, pages 56–, New York, NY, USA, 2004. ACM.
6. J. A. Lee, D. H. Peluffo-Ordóñez, and M. Verleysen. Multiscale stochastic neighbor embedding: Towards parameter-free dimensionality reduction. In *ESANN 2014*, 2014.
7. J. A. Lee, E. Renard, G. Bernard, P. Dupont, and M. Verleysen. Type 1 and 2 mixtures of Kullback-Leibler divergences as cost functions in dimensionality reduction based on similarity preservation. *Neurocomputing*, 112:92–108, July 2013.
8. J. A. Lee and M. Verleysen. *Nonlinear dimensionality reduction*. Springer, 2007.
9. J. A. Lee and M. Verleysen. Quality assessment of dimensionality reduction: Rank-based criteria. *Neurocomput.*, 72(7-9):1431–1443, 2009.
10. B. Mokbel, W. Lueks, A. Gisbrecht, and B. Hammer. Visualizing the quality of dimensionality reduction. *Neurocomputing*, 112:109–123, 2013.
11. L. van der Maaten. Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research*, 15:3221–3245, 2014.
12. L. van der Maaten and G. Hinton. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, November 2008.
13. Y. Zhu, E. Song, J. Zhou, and Z. You. Optimal dimensionality reduction of sensor data in multisensor estimation fusion. *Signal Processing, IEEE Transactions on*, 53(5):1631–1639, May 2005.

Impact of Regularization on the Model Space for Time Series Classification

Witali Aswolinskiy, René Felix Reinhart and Jochen Steil

Research Institute for Cognition and Robotics - CoR-Lab
Universitätsstraße 25, 33615 Bielefeld, Germany
{waswolinskiy, freinhart, jsteil}@cor-lab.uni-bielefeld.de
<http://www.springer.com/lncs>

Abstract. Time series classification is an active research field and applicable in many domains, e.g. speech and gesture recognition. A recent approach to classify time series is based on modelling each time series by an Echo State Network and then to classify the time series in the readout weight or model space of these networks. In this paper, we investigate the effect of Echo State Network regularization on the model space. The results show that regularization has a strong impact on the model space structure and the separability of the time series in the model space.

Keywords: model space, echo state networks, reservoir computing, time series classification, time series clustering, regularization

1 Introduction

In time series classification, a label is assigned to a sequence of data points. One main challenge is that the time series can have different lengths. A recent approach is learning in the model space: For each time series a model is trained and the model's parameters are used as features in a consecutive classification stage [2]. Typically, the models are trained to minimize the one-step-ahead prediction error on the time series. The number of model parameters is independent of the time series length, which allows to employ any feature based classifier from machine learning theory in the classification stage.

Echo State Networks (ESNs) [5] are promising candidates for models in this context, because they offer temporal integration of the input, nonlinear computation and quick training. Learning in the model space of ESNs was evaluated for time series classification [2] and clustering [3] with promising results. Here, we investigate the influence of regularization during training of ESNs on classification performance and model space structure.

Regularization is a technique to reduce model complexity in order to prevent overfitting and comes in the context of ESNs often in form of ridge regression. In ridge regression not only the sum of squared residuals is minimized, but also the sum of the squared regression coefficients. This shrinks the coefficients and thereby reduces their variance. Without regularization, and non-orthogonal data vectors, the coefficients can have a large magnitude and be unstable - minor

changes in the data can lead to major changes of the coefficients [4, 10]. This is especially important in the context of learning in the model space, where these coefficients form the model space. For a better understanding of the model space we investigate the effects of different forms of regularization on the model space. We show that L2 but not L1 regularization achieves good classification results, and that the modelling error is not necessarily predictive for the classification performance in the model space.

2 Learning in the Model Space of Echo State Networks

A classic ESN architecture consists of a reservoir with recurrently connected neurons and a linear regression readout. The reservoir states \mathbf{x} and the readouts \mathbf{y} are updated according to

$$\mathbf{x}(k+1) = (1 - \lambda)\mathbf{x}(k) + \lambda f(\mathbf{W}^{rec}\mathbf{x}(k) + \mathbf{W}^{in}\mathbf{u}(k+1)) \quad (1)$$

$$\mathbf{y}(k) = \mathbf{W}^{out}\mathbf{x}(k), \quad (2)$$

where λ is the leak rate, f is the activation function, e.g. tangens hyperbolicus, \mathbf{W}^{rec} the recurrent weight matrix, \mathbf{W}^{in} the weight matrix from the inputs to the neurons and \mathbf{W}^{out} the weight matrix from the neurons to the readouts \mathbf{y} . Only \mathbf{W}^{out} is trained with linear regression - the other weights are initialized randomly and remain fixed.

For classification with ESNs, input time series are fed into the reservoir and the readout is typically trained on the reservoir states with linear regression to predict the class label for each step of the time series. In contrast, in learning in the model space of ESNs, the ESNs are an intermediate step to create a time-independent representation of the time series. Let a dataset consist of N discrete time series $\mathbf{u}_i, i = 1 \dots N$ with varying lengths $K_i: \mathbf{u}_i(0), \dots, \mathbf{u}_i(k), \dots, \mathbf{u}_i(K_i)$. For each time series \mathbf{u}_i , an ESN is trained to predict from the previous step $\mathbf{u}_i(k)$ the next step $\mathbf{u}_i(k+1)$ in the time series. The ESNs are trained independently, but share the same reservoir parameters \mathbf{W}^{in} and \mathbf{W}^{rec} in order to create a coherent model space. The prediction error

$$E(\mathbf{W}_i^{out}) = \frac{1}{K_i} \sum_{k=1}^{K_i} (\mathbf{u}_i(k) - \mathbf{W}_i^{out}\mathbf{x}_i(k))^2 + \alpha \|\mathbf{W}_i^{out}\|_L \quad (3)$$

for time series i is minimized, where α is the regularization strength and L is the regularization norm. The resulting readout weights \mathbf{W}_i^{out} form the model space, where the classification takes place.

In the model space, arbitrary classification algorithms can be used. This approach will be denoted here as model space learning (MSL) and is visualized in Fig. 1. The example shows the transformation of the data - here noisy sum of sine waves - via regression on the reservoir activations to the readout weights. In this example, the time series and hence the reservoir activations are 100 steps long. The dimensionality of the model space depends on the dimensionality of

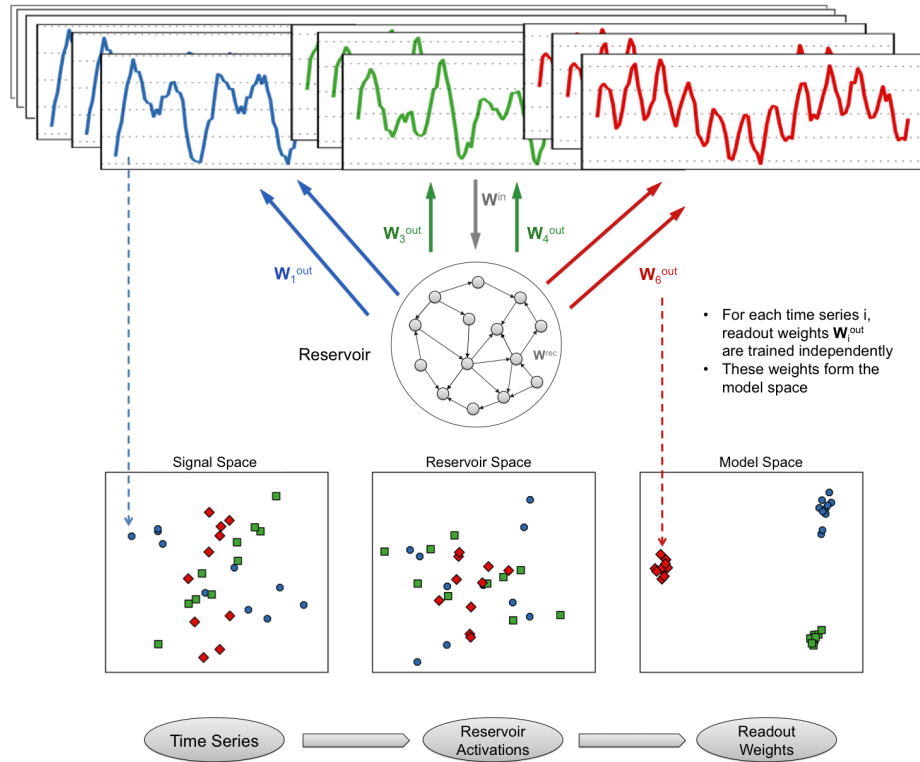


Fig. 1. Learning in the Model Space of ESNs. In the upper part of the diagram, noisy sine waves are shown, which are fed into the reservoir. For each time series a readout is trained to predict the next input. In the lower part of the diagram, the time series, the reservoir echos and the readout weights are visualized with matching colors. The signal, reservoir activation and readout weight matrices were projected to a plane using PCA. In the model space, the time series of different classes are easily separable.

the signal and the number of reservoir neurons. For signals with d dimensions and reservoirs with n neurons the model space dimensionality is $d \cdot n$ or $d \cdot (n + 1)$ if a regression intercept is used. For comparison, in the lower part of Fig. 1, each time series is represented as a point in signal space, reservoir space (the space of reservoir neuron activations) and model space. For visualization purposes, the data was projected to two dimensions via principal component analysis. In this example, time series from different classes can be easily separated in the model space.

3 Effect of Regularization on the Model Space

In MSL regularization can occur at two stages. First, while training the ESN models and second, while training a classifier in the model space where the ESN

readout weights serve as features. Here, we focus on the effect of regularizing the ESN readout weights \mathbf{W}^{out} . The regularization strength for ESN training is given by α in Eq.(3).

3.1 L2 and L1 Regularization

In L2-regularization or ridge regression the sum of squared residuals is minimized. In L1-regularization or Lasso regression the magnitude of the weights is minimized [9]. This results in a more sparse weight distribution. We evaluated the effect of regularization in the ESN on the model space on two multivariate datasets from the the UCI repository [7]. Since a direct analysis of model space properties is difficult, we assess the model space structure indirectly by evaluating pattern separability in the classification stage.

The Australian Sign Language Signs (AUSLAN) dataset consists of 2565 samples (27 repetitions x 95 signs), recorded from a native signer using hand position trackers. The sequences are between 45 and 136 steps long and have 22 input dimensions.

The Spoken Arabic Digit (SAD) dataset [6] consists of 8800 samples (10 digits x 10 repetitions x 88 speakers) with 13 input dimensions - Mel Frequency Cepstral Coefficients (MFCCs). The sequences are between 4 to 93 steps long.

Classification error rates were obtained via five times repeated random sub-sampling, also known as Monte Carlo cross validation [8]. In the AUSLAN dataset, from the 2565 samples, in each fold, 600 randomly selected samples were used for training and the rest for testing. In the SAD dataset the split was half-and-half.

After transforming the time series to model parameters by training the ESNs, in the model space, two classifiers were evaluated: Ridge classifier and support vector machine (SVM) with radial basis function kernel. The ridge classifier is a linear classifier trained with ridge regression: The K class labels are encoded in a 1-of-K scheme and the regressed scores transformed to estimated class labels with the winner-takes-all method. Since the random initialization of the reservoir weights \mathbf{W}^{in} and \mathbf{W}^{rec} affects the performance, only the results of the best out of five reservoirs were used. In order to mitigate the influence of the classifier parameters on the model space analysis, for each α value and classifier, several classifier parameter values were evaluated and from these evaluations the lowest error rate was chosen. The used ESN and classifier parameters are listed in the Appendix.

L2 Regularization: The classification error rates for the datasets at different regularization strengths are depicted in Fig. 2 (A,B). The diagram can be divided in three areas: Overfitting on the left, underfitting on the right and the lowest error rates in the middle. Interestingly, the SVM classifier does not achieve lower error rates than the linear ridge classifier. This suggests, that the considered classification tasks can be solved linearly in the model space.

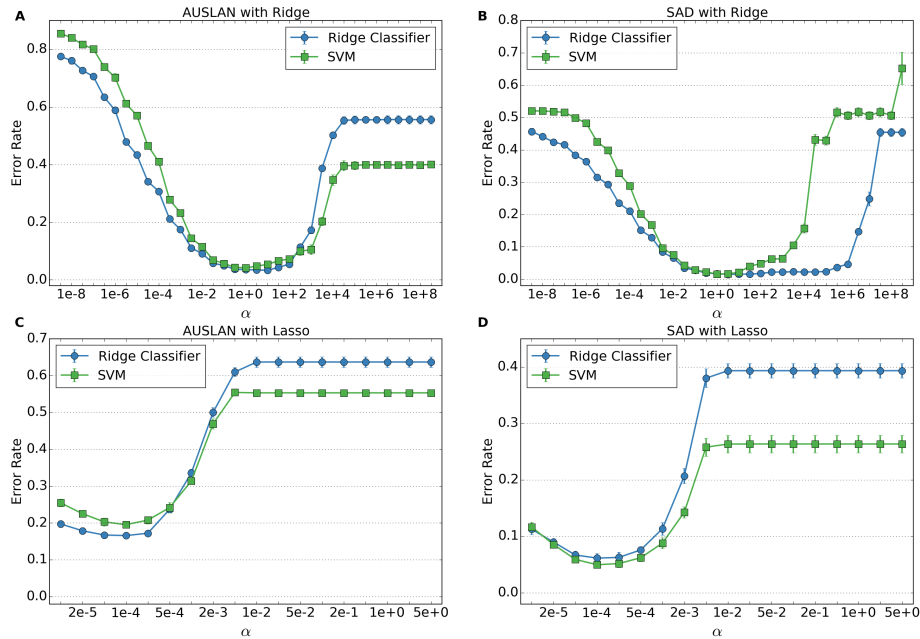


Fig. 2. Classification error rates in the AUSLAN and SAD datasets with ridge (A,B) and Lasso (C,D) regression for different ESN regularization strengths α and two classifiers. The average error rate of 5-fold Monte Carlo cross validation is shown. The very small standard deviations are indicated by error bars.

L1 Regularization: Fig. 2 (C,D) shows the classification error rates for Lasso regression. Compared to Fig. 2 (A,B) the performance is considerably worse. With ridge regression the error rate in AUSLAN was $3.31 \pm 0.4\%$ and in SAD $1.5 \pm 0.5\%$. With Lasso regression, the lowest error rate in AUSLAN was $16.6 \pm 0.4\%$ and in SAD $5.0 \pm 0.6\%$. These results indicate that sparseness in the model space is detrimental for classification performance. A possible explanation is the nature of L1-regularization: The explanatory variables mostly correlated with the target variable are selected. In MSL, Lasso is executed for each time series independently and thus, for similar but slightly different time series of the same class, different reservoir neurons may be selected. This effect emphasizes the difference between the samples in the model space rather than their similarity. A manual inspection of the model space validated the presence of this effect.

3.2 Difficulty of the Task

In order to investigate, whether there is dependence between regularization and the difficulty of the classification task, we devised a synthetic dataset. The task

is to differentiate between time series

$$\begin{aligned} u(k) &= \sin(0.2k) + \sin(0.311k) + \sin(0.42k) \\ u(k) &= \sin(0.2k) + (1 - \Delta) \sin(0.311k) + (1 + \Delta) \sin(0.42k) . \end{aligned}$$

The difficulty of the task is controlled with the parameter Δ : The smaller the value of Δ , the more similar are the respective time series and thus the more difficult the task. The dataset consists of 200 time series of length 100 created by generating 10,000 steps of each sequence and dividing them in 100 parts each. Half of the time series were used for training and half for testing.

Fig. 3 shows how the model prediction error (A) and the classification error in the model space (B) depend on the regularization strength α and task simplicity Δ . The prediction error is the normalized root mean square error (NRMSE) computed for predicting the next input during training of the ESN models and was averaged over all models. With $\Delta=0$, the two time series classes are the same and thus not separable (see first column in Fig. 3B). With increasing Δ the difference between the time series classes increases and the classification error rate decreases. Weak regularization with $\alpha < 10^{-5}$ leads to overfitting: Low model prediction error, but high classification error in the model space (see the upper parts of Fig. 3A and 3B). Regularizing with $\alpha > 10$ leads to underfitting during model training, but the classification error rates remain low until α exceeds 10^7 (see the lower parts of Fig. 3A and 3B). Regularizing with $10 < \alpha < 10^3$ achieves the best results. Noteworthy is that no correlation between the prediction error and classification error rate is observable. This means that for pattern separation in the model space, it is less important how exact a model fits the data.

3.3 Noise and Overfitting

The main goal of regularization is to prevent the learning of noise which leads to smaller errors on the training data, but decreases the generalization capability and therefore causes larger errors on the test data. As the amount of noise in the UCI datasets is unknown, we use again a synthetic dataset to study the effect of regularization on the model space in the presence of noise. The task is to differentiate between time series of the form

$$\begin{aligned} u(k) &= \sin(0.2x) + \sin(0.311x) + \sin(0.42x) + \epsilon \\ u(k) &= \sin(0.2x) + 0.7 \sin(0.311x) + 1.3 \sin(0.42x) + \epsilon . \end{aligned}$$

This corresponds to the synthetic dataset from the previous section with $\Delta=0.3$ and added noise. During the simulations, the standard deviation σ_ϵ of the Gaussian distributed noise ϵ was varied. Fig. 3C and 3D show the the average of the prediction error and the classification error rate for different regularization strengths α and noise levels σ_ϵ . Best results are achieved with α around 1. Here too, no correlation between the model prediction error and the classification performance in the model space can be observed. Low classification rates are possible even with large model prediction errors, e.g. a NRMSE of 1.0.

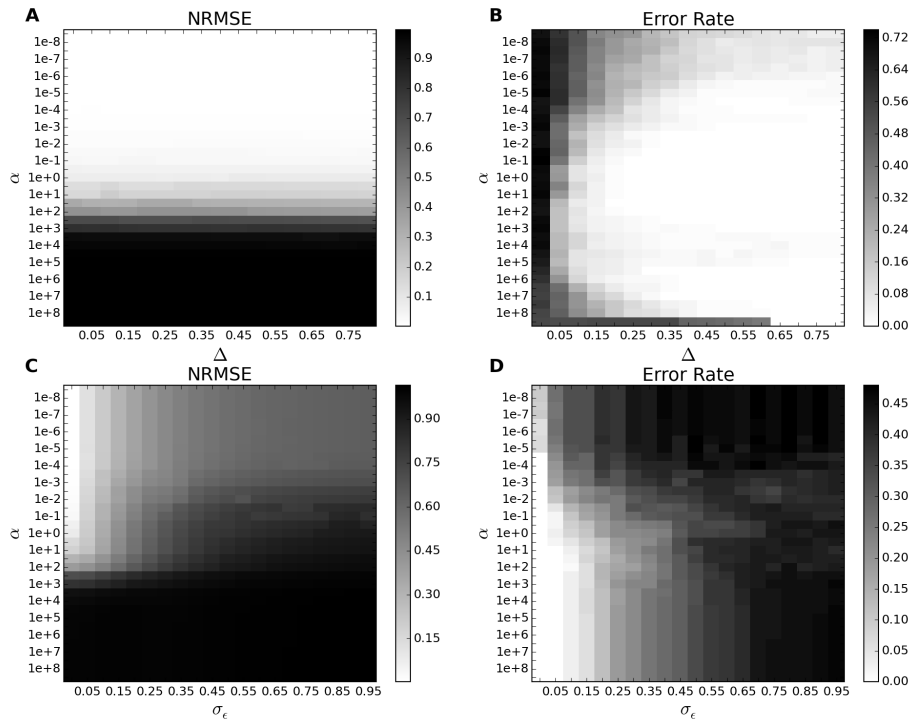


Fig. 3. Average NRMSE for predicting the next value in a time series with the ESNs (A,C) and classification error rate in the model space (B,D). Dark cells denote high values and bright cells low values.

4 Conclusion

In this paper we investigated the effect of regularization during training of ESNs on the classification performance in the readout weight space (i.e. model space) of these ESNs. Regularization improved the performance for more difficult tasks as well as in the presence of noise. L2 regularization performed considerably better than L1 regularization. In conclusion, L2 regularization is more suitable for MSL, and the model prediction error can not be used to estimate the expected classification performance in the model space.

5 Appendix

Preprocessing: All datasets were scaled to the range $[-1/d, 1/d]$.

Reservoir Parameters: n : Number of reservoir neurons; Input Scaling: Scaling of the weights from the input to the reservoir; Rec. Connectivity: Density of the recurrent weight matrix \mathbf{W}^{rec} , Rec. Scaling: Spectral radius of the re-

current weights \mathbf{W}^{rec} , λ : Leak rate of the neurons, Bias Scaling: Scaling of the neuron biases.

- Synthetic: n : 50, Input Scaling: 1, Rec. Connectivity: 10%, Rec. Scaling: 0.9, λ : 1, Bias Scaling: 0.1
- AUSLAN: n : 100, Input Scaling: 1, Rec. Connectivity: 10%, Rec. Scaling: 0.9, λ : 1, Bias Scaling: 0.1
- SAD: n : 50, Input Scaling: 2, Rec. Connectivity: 10%, Rec. Scaling: 0.9, λ : 0.7, Bias Scaling: 0.1

Classifier parameters: For both synthetic datasets, a linear regression classifier without regularization was used. In AUSLAN and SAD:

- Ridge Classifier: Ridge from $1e-5$ to $1e+5$ in logarithmic scale.
- SVM Classifier [1]: Penalty C from 0.1 to 10000 in logarithmic scale, Kernel coefficient gamma from 0.01 to 100 in logarithmic scale.

Acknowledgments. This project is funded by the German Federal Ministry of Education and Research (BMBF) within the Leading-Edge Cluster Competition “it’s OWL” (intelligent technical systems OstWestfalenLippe) and managed by the Project Management Agency Karlsruhe (PTKA). The authors are responsible for the contents of this publication.

References

1. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27 (2011), software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>
2. Chen, H., Tang, F., Tino, P., Yao, X.: Model-based kernel for efficient time series analysis. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 392–400 (2013)
3. Chen, H., Tino, P., Rodan, A., Yao, X.: Learning in the model space for cognitive fault diagnosis. IEEE Transactions on Neural Networks and Learning Systems 25(1), 124–136 (2014)
4. Hoerl, A.E., Kennard, R.W.: Ridge regression: applications to nonorthogonal problems. Technometrics 12(1), 69–82 (1970)
5. Jaeger, H.: The “echo state” approach to analysing and training recurrent neural networks-with an erratum note. GMD Technical Report 148, 34 (2001)
6. Kadous, M.W.: Temporal classification: Extending the classification paradigm to multivariate time series. Ph.D. thesis, The University of New South Wales (2002)
7. Lichman, M.: UCI machine learning repository (2013), <http://archive.ics.uci.edu/ml>
8. Picard, R.R., Cook, R.D.: Cross-validation of regression models. Journal of the American Statistical Association 79(387), 575–583 (1984)
9. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological) pp. 267–288 (1996)
10. Vinod, H.D.: A survey of ridge regression and related techniques for improvements over ordinary least squares. Review of Economics and Statistics pp. 121–131 (1978)

Ensembles of Neural Oscillators

Danil Koryakin, Fabian Schrodt, Martin V. Butz

Cognitive Modeling, Department of Computer Science,
University of Tübingen, Sand 14, 72076 Tübingen, Germany

Abstract. Modularization is a promising direction for the further development of artificial neural networks (ANNs), and a large variety of modularized ANNs have been proposed. Possibly the main advantage of modularization is that, due to the wiring and learning mechanisms, different modules in the ANN can be biased towards developing particular problem solution substructures – allowing the incorporation of a priori problem knowledge. We present a modular Echo-State Network (mESN) architecture, where modules process independent recurrences. The structure enables the modeling of complex, periodic functions by means of an additive combination of elementary oscillations. We compare the mESN to monolithic networks on problems of different complexity and confirm superior performance. Finally, we sketch out potential applications and future work directions.

Keywords: Modular Neural Network, Recurrent Neural Networks, Echo State Network, Periodic Dynamics

1 Introduction

Artificial Neural Networks (ANNs) are very general machine learning systems, which have been applied in many areas of science and engineering. To tune ANNs for solving particular problems, researchers have proposed numerous ANN architectures, with different topologies, types of neurons involved, learning mechanisms, and modularizations by constraining neural connections.

Modularity typically groups neurons in an ANN towards specific functionalities and restricts communication between the groups of neurons in a particular manner. For example, in [7] and [8] ANNs are composed of several feed-forward neural networks. These modules are mediated by a gating network, which chooses the output of the most relevant module. The modules are combined either hierarchically or recursively depending on the problem to be solved. Such modular networks have been applied to engineering applications, such as trajectory modeling [8] and recognition [3, 7]. More complex modularized ANNs have been considered in cognitive science for understanding processes in the human brain [9] or for controlling complex humanoid robots [1].

In this work we propose a modular, recurrent ANN (RNN) architecture for modeling time series. Although our architecture generally allows the usage of any type of RNN as a module, we focus on Echo-State Networks (ESNs). Thus, we

call the investigated architecture modular ESN (mESN). As it was shown in [10], despite their compact size, ESNs are able to model quite complex oscillators. They are flexible and can be driven either by a signal from alternative input neurons or operate as autonomous oscillators.

The possibility to use ESNs as autonomous oscillators makes them especially suitable for applications where once learned target dynamics must be reproduced on request. In such applications the networks can serve as a memory for complex time series. Compactness of ESNs would bring added value to such devices. Despite this potential benefit, applications for input-less reproductions of time series are sparse. They include applications for cyclic rehearsal [4], [5] and for artificial problems [4], [5], [11].

In the next section we detail the mESN architecture. Section 3 evaluates the mESN in the task of reproducing of mixtures of periodic signals and analyzes its performance. Section 4 sketches possible practical applications of the presented model. In the conclusions, we summarize our work and draw future research perspectives.

2 Modular ESN

The modular Echo-State Network (mESN) consists of several modules, each of them being an ESN. These modules do not have direct connections to each other and therefore operate independently. The mESN is shown schematically in Figure 1. The output of an mESN is computed as a linear combination of outputs of its modules as follows.

$$o(n) = \sum_{i=1}^M r_i o_i(n) \quad (1)$$

where M is the number of modules, $o_i(n)$ is the output of the i^{th} module at time step n , and r_i is its responsibility for generating the output.

Using standard ESNs as modules in mESN is particularly advantageous for modeling time series that consist of multiple oscillators. ESNs have been shown to be able to produce accurate, oscillating output signals for a long period of time without further external stimulation, solely driven by their own output feedback. In mESN, the output signal $o_i(n)$ of one ESN module i is computed as follows:

$$o_i(n) = \mathbf{W}_{OUT,i} [\mathbf{f}_i (\mathbf{W}_i \mathbf{x}_i(n-1) + \mathbf{W}_{OFB,i} o_i(n-1))], \quad (2)$$

where n is the current time step, $\mathbf{x}_i(n-1)$ is the vector of reservoir states at the previous time step, \mathbf{W}_i is the matrix of reservoir weights, \mathbf{f}_i is the vector of activation functions of all reservoir neurons, $\mathbf{W}_{OUT,i}$ is the matrix of output weights, and $\mathbf{W}_{OFB,i}$ is the matrix of output feedback weights. Expression (2) is derived from the more general expression of ESNs, ignoring signals from possibly additional input neurons. The general expression for updating an ESN output, rules for designing the matrix \mathbf{W} as well as a training procedure for the standard ESNs can be found in [6].

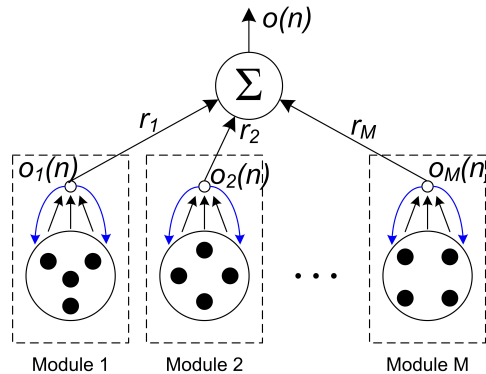


Fig. 1. Structure of the mESN consisting of M modules. Responsibilities r_i of the modules define a portion of each module in the total network output $o(n)$, which is computed as a linear combination of the individual module outputs.

The design of the mESN requires the choice of macro parameters for each ESN module as well as the independent training of the output weights $\mathbf{W}_{OUT,i}$. To train the modules, a preliminary decomposition of a training sequence Y into a set of sequences is necessary:

$$Y = (Y_1 \oplus Y_2 \oplus \dots \oplus Y_M) \quad (3)$$

where Y_i is the time series assigned to the i^{th} module. The operator \oplus denotes an application-dependent split of every value $y(n)$ of the sequence Y into a set of values $y_i(n)$, each of them belonging to the corresponding component Y_i at the current time step n .

In our study the decomposition was performed by an expert using a priori knowledge about the target time series. Thus, additional information about a problem was incorporated into the model during the design phase. A similar approach was described elsewhere [12, 2]. In these studies feedforward neural network modules were trained independently of each other on already segmented sequences. In our study, ESN modules were also trained independently of each other using the training procedure

$$\mathbf{W}_{OUT,i} = \mathbf{M}_i^{-1} \mathbf{T}_i, \quad (4)$$

where \mathbf{M}_i^{-1} is the inverted matrix of states of the i^{th} module on its training sequence and \mathbf{T}_i is the corresponding sequence of target outputs.

Modularity provides an opportunity to choose the macro parameters independently for each module. These parameters include the module's reservoir size, a spectral radius of the matrix \mathbf{W}_i , and an interval for initialization of the feedback weights $\mathbf{W}_{OFB,i}$. An independent parameter choice is especially useful for time series that consist of components with very different characteristics, such as very slow and fast dynamics – where larger and smaller spectral radii are

more suitable, respectively – or differing component complexities – where the reservoir sizes should be adapted accordingly [10].

3 Experiments

In this section we demonstrate the merit of mESNs when modeling time series of different complexity. We focus on smooth sine waves and triangular signals. The latter are not continuously differentiable and consequently more challenging. We compared the performance of our mESN implementation with our corresponding single-reservoir ESN implementation.

3.1 Experimental Setup

In the experiments we focus on dynamic reservoirs that were randomly generated for different combinations of ESN parameter settings. These parameters were reservoir size, connectivity of a dynamic reservoir, and scaling of the feedback weights \mathbf{W}_{OFB} . Ranges of connectivity and of \mathbf{W}_{OFB} were the same for standard ESNs and for the mESNs, and were set to $\{0.1, 0.2, \dots, 1.0\}$ and $\{10^{-10}, 10^{-9}, \dots, 3.9, 4.0\}$, respectively. Ranges of reservoir sizes were different for the evaluated ESNs and mESNs. Modules in mESNs were equipped with only 4 to 10 reservoir neurons. The single-reservoir in the ESNs had from 10 to 100 neurons. For each combination of the parameter values, 500 networks were generated, trained, and evaluated on a test sequence. Networks with the smallest normalized root mean squared errors (NRMSE) were chosen as ESN modules for the mESN. The mESN performance was compared with the performance of the best single-reservoir ESN.

We considered three sets of target dynamics. The first time series was composed of sine waves, the second was composed of triangular signals, and the third was composed of a mixture of the two. For each target dynamics, a sequence of 700 time steps was generated, which was split into a washout sequence (the first 100 time steps), a training sequence (following 300 steps), and a test sequence (the last 300 time steps).

Mixtures of sine waves are known in the literature as Multiple Superimposed Oscillators (MSO). The number of sine waves defines the complexity of a dynamics: more sine waves constitute more difficult dynamics. The whole family of the MSO dynamics can be described as follows:

$$y(n) = \sum_{i=1}^s \sin(\alpha_i n), \quad (5)$$

where s is the number of sine waves and α_i specify their respective frequencies. We generated the sequences for the following standard set of the frequencies $\alpha_1 = 0.2$, $\alpha_2 = 0.311$, $\alpha_3 = 0.42$, $\alpha_4 = 0.51$, $\alpha_5 = 0.63$, $\alpha_6 = 0.74$, $\alpha_7 = 0.85$, and $\alpha_8 = 0.97$. Because of the smoothness of the individual components, the MSOs are moderately non-linear and continuously differentiable over the whole time axis.

Like the MSOs, the mixtures of triangular signals (MTS) are linear combinations of their components. Each component is a periodic triangular signal characterized by a period and an amplitude. The amplitude of all components was one. The periods were set to the following integers $\{32, 24, 20, 12, 8, 4\}$, which yields periods similar to the MSO ones. Despite this similarity, the MTS are more difficult because of much higher non-linearity at peaks of the triangular signals. Figure 2 shows curves of the least and most complex MTS dynamics, MTS2 and MTS6. As can be seen, an MTS with more components resembles a chaotic attractor. But in contrast to known chaotic attractors, both MSOs and MTSs have internal structures that are very suitable for modularization.

For example, MTS2 consists of two distinct components Y_1 and Y_2 , with periods 32 and 24 time steps, respectively. To model MTS2 with mESN, two ESN modules are employed. Each module is then trained on one of the components and the best ESN is chosen, respectively, by means of the stochastic search procedure detailed above. A dynamic reservoir for every candidate was updated using the formula (2) at every time step of the washout and training sequences. The output weights $\mathbf{W}_{OUT,i}$ of module i were trained using formula (4) given the training sequence Y_i , that is, the corresponding target values $o_i(n)$ and \mathbf{T}_i , which appear in formulae (2) and (4), were taken from the sequence Y_i .

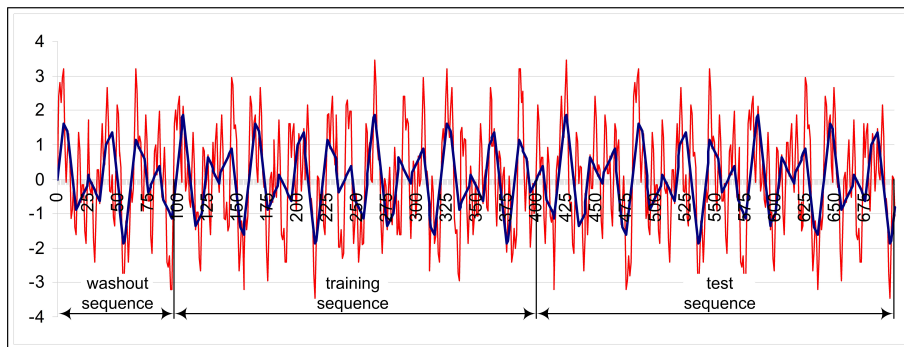


Fig. 2. Mixtures of triangular signals showing the simplest (MTS2, blue curve) and most complex dynamics (MTS6, red curve) considered. The sequences are split into washout, training and test intervals.

4 Results

Table 1 shows reached performances and sizes of the best mESNs and ESNs found for time series with two, four, six, and eight dynamic components. Performance of the standard ESNs varies over a wide range from 10^{-12} to 10^{-2} . Standard ESNs had big difficulties on more complex dynamics and could not model the most difficult target MST4.4 at all. At the same time, as expected,

Table 1. Performance of the best mESN and ESN on the respective target dynamics of different complexity. MSOs are mixtures of sine waves, "TriX" stands for a mixture of X different triangular signals, and "MSTX.Y" stands for a combination of X sine waves with Y triangular signals. The number of neurons is the summed number of reservoir neurons in all modules in the best mESN / ESN found for the respective problem.

mESN			
Number of Components	MSO & NRMSE & size	Triangular NRMSE & size	MST & NRMSE & size
2	MSO2: 5.62×10^{-10} with 8 neurons	Tri2: 7.99×10^{-7} with 10 neurons	MST1.1: 8.08×10^{-7} with 10 neurons
4	MSO4: 6.47×10^{-10} with 16 neurons	Tri4: 8.37×10^{-7} with 18 neurons	MST2.2: 8.63×10^{-7} with 18 neurons
6	MSO6: 7.83×10^{-10} with 24 neurons	Tri6: 9.45×10^{-7} with 26 neurons	MST3.3: 1.04×10^{-6} with 26 neurons
8	MSO8: 1.07×10^{-9} with 32 neurons	-	MST4.4: 1.59×10^{-6} with 34 neurons
ESN			
Number of Components	MSO & NRMSE & size	Triangular NRMSE & size	MST & NRMSE & size
2	MSO2: 2.51×10^{-12} with 5 neurons	Tri2: 2.42×10^{-6} with 30 neurons	MST1.1: 3.29×10^{-6} with 20 neurons
4	MSO4: 5.72×10^{-8} with 9 neurons	Tri4: 7.31×10^{-4} with 90 neurons	MST2.2: 3.16×10^{-3} with 90 neurons
6	MSO6: 8.43×10^{-5} with 14 neurons	Tri6: 1.83×10^{-3} with 100 neurons	MST3.3: 5.80×10^{-2} with 90 neurons
8	MSO8: 2.73×10^{-4} with 68 neurons	-	MST4.4: no ESN could model the dynamics

the complexity of the time series had only a minor impact on the performance of the mESNs. Their test errors varied only within four orders of magnitude. Also the most difficult dynamic, MST4.4, which consists of four sine components and four triangular components, was solved with high accuracy. The high performance of the mESNs was also reached thanks to an individual choice of critical parameters for each component, which was especially helpful on mixtures of MSOs and MTSs. Whereas the sine waves needed similar \mathbf{W}_{OFB} settings below 10^{-6} , some triangular signals required reservoir neurons to operate in a saturation range with \mathbf{W}_{OFB} up to 3.8. Such a wide parameter spread is infeasible for a single-reservoir ESN.

Besides the performance gain, modularity is very favorable for the generation of compact models of complex target dynamics. The largest reduction in model size was observed in the sequence "Tri4", where mESN required 18 neurons whereas 90 neurons were needed with standard ESNs. In order to reach higher performance, a single-reservoir ESN increases a variety of reservoir states through formation of complementary neural paths. This automatically requires a larger reservoir. On the contrary, in the mESN splitting target components leads to decoupling of internal dynamics. As a result, smaller ESN modules provide a sufficient variety of internal dynamics for the corresponding target component. However, splitting the components causes a slight size overhead in mESNs. This can be seen in the easiest target dynamics, such as MSO2, where mESN re-

quired 8 neurons while a single-reservoir ESN solved the problem best with only 5 neurons.

5 Discussion

The experiments showed advantages of an mESN ensemble over a single-reservoir ESN. The main advantage is the possibility to decouple internal dynamics from each other. Like other modular architectures, it allows incorporating a priori knowledge to do a focused choice of modules' parameters and to reach higher accuracy with more compact models. Besides that, such an organization offers flexibility and robustness for potential applications. Modules of different types can be plugged into the ensemble. Switching off a malfunctioning module allows avoiding an abrupt reduction in system performance.

The mESN is useful for modeling periodic patterns of any complexity and any period, especially when consisting of different components. Currently we see at least three potential applications. The first one is an analysis of a time series through mESN synchronization. It will produce an mESN whose active ESN modules will indicate which components are present and how they are mixed with each other.

The second application is a flexible control of a robot arm shown in Figure 3. Its end effector may be required to draw a complex trajectory periodically. Each ESN module may be linked to its own joint and produce a specific trajectory independently of the other modules – leading to the generation of the overall, target trajectory with the end effector. Alternatively, the end-effector trajectory may be controlled by selectively switching mESN components on and off over time, possibly enabling the generation of digits on any surface and with any surface orientation that is reachable with the robot arm.

Another relevant application is the generation of central pattern generators (CPG), where primitive rhythmic signals are combined into more complex patterns. An mESN will represent a CPG with a population of tiny ESN modules.

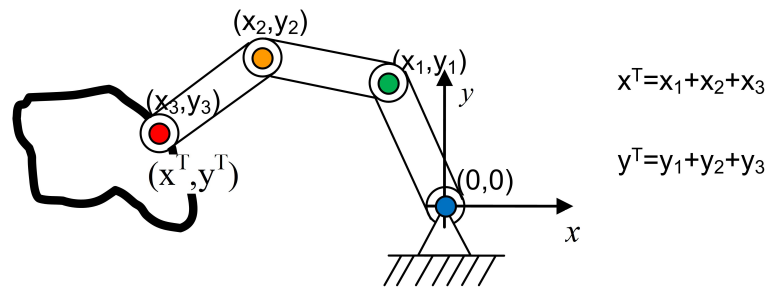


Fig. 3. mESN-based loopless control of a robot arm. Coordinates (x^T, y^T) of the end effector (red point) on a trajectory (thick line) are a sum of projections of coordinates of individual joints. (x_i, y_i) are coordinates of i^{th} joint in a coordinate system with the origin at the $(i - 1)^{th}$ joint. Coordinates (x_i, y_i) are output of the i^{th} ESN module.

A linear combination of their outputs will be used to tune the CPG to a target behavior. CPG parameterization may be realized by augmenting the individual neural oscillators with input neurons.

6 Conclusions and Outlook

In this paper we presented the modularized echo state network architecture mESN. In mESN modules are combined without a switching block common for mixtures of experts. The independent operation of the neural oscillators realizes component decoupling, enabling local parameter and meta parameter optimization for each module and time series component. The proposed model is useful for practical applications that deal with decomposable and switching processes. Currently, we are investigating possibilities to tune an ensemble of neural oscillators after changes in the target dynamics, such as amplitude and phase. Moreover, we are working on automatizing the mESN modularization.

References

1. Byadarhaly, K.V., Perdoor, M.C., Minai, A.A., A modular neural model of motor synergies, *Neural Networks* 32, pp. 96–108 (2012)
2. Gradjevic, N., Gencay, R., Kukulj, D., Option Pricing With Modular Neural Networks, *IEEE Transactions On Neural Networks* 20, pp. 626–637 (2009)
3. Happel, B.L.M., Murre, J.M.J., The Design and Evolution of Modular Neural Network Architectures, *Neural Networks* 7, pp. 985–1004 (1994)
4. Holzmann, G., Hauser, H., Echo State Networks with Filter Neurons and a Delay&Sum Readout, *Neural Networks* 23, pp. 244–256 (2010)
5. Jaeger, H., The "echo state" approach to analysing and training recurrent neural networks, Technical Report 148, German National Research Institute for Computer Science (2001)
6. Jaeger, H., A tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach, Technical Report 159, German National Research Center for Information Technology (2002)
7. Jacobs, R.A., Jordan, M.I., Barto, A.G., Task Decomposition Through Competition in a Modular Connectionist Architecture: The What and Where Vision Tasks, *Cognitive Science* 15, pp. 219–250 (1991)
8. Jordan, M.I. Jacobs, R.A., Hierarchical mixtures of experts and the EM algorithm, *Proceedings of 1993 International Joint Conference on Neural Networks*, pp.1339–1344 (1993)
9. Kharratzadeh, M., Shultz, T., Neural-network modelling of Bayesian learning and inference, 35th Annual Conference of the Cognitive Science Society, pp. 2686–2691. Berlin, Germany, Cognitive Science Society (2013)
10. Koryakin, D., Lohmann, J., Butz M.V., Balanced echo state networks, *Neural Networks* 36, pp. 35–45 (2012)
11. Reinhart, F., Steil, J., Reservoir regularization stabilizes learning of Echo State Networks with output feedback, In *Proceedings of ESANN2011, European Symposium on Artificial Neural Networks* (2011)
12. Soltani, S., On the use of the wavelet decomposition for time series prediction, *Neurocomputing* 48, pp. 267–277 (2002)

Ensemble Methods and Active Learning in HCI

Patrick Thiam, Markus Kächele, Friedhelm Schwenker, and Günther Palm

Institute of Neural Information Processing,
Ulm University, Germany
{firstname.lastname}@uni-ulm.de

1 Introduction

An explosion in terms of amount and complexity of data is being witnessed in current days. The globally available amount of data is increasing so fast that a proper assessment and annotation of the data is almost impossible. Consequently, annotated data is scarce while unannotated data is available in huge amounts. Supervised Learning [4] and Semi-Supervised Learning [3] rely on annotated data in order to successfully perform classification or regression tasks. The robustness as well as the performance of a model trained with such techniques highly depend not only on the amount of annotated samples available, but also on the quality of the annotation process. Thus the creation of a large and reliable annotated corpus is the bottleneck in the domain of Supervised and Semi-Supervised Learning. Meanwhile, data annotation is known to be hard, expensive both in time and costs, and error prone [6].

Active learning [8] is a technique that can be used to address these issues. The main characteristic of active learning is the ability of the learner to select the samples from which it learns. More specifically, the learner selects the most informative unannotated samples to be annotated by an oracle (e.g. a human annotator). In this way the learner is able to improve its performance and accelerate its learning process, while reducing the costs for the annotation of an entire corpus. The following study focuses on developing and implementing active learning methods for unimodal and multimodal emotion recognition in human computer interaction.

2 Proposed Approach

We propose a pool-based active learning approach based on novelty detection [10] combined with binary classification. The very first step consists of using the local binary patterns on three orthogonal planes (LBP-TOP) [11] to extract the features from the video sequences. The algorithm then begins with a small set of annotated samples L and a large set of unannotated samples U . At each iteration the most uncharacteristic samples of the unannotated set are detected using a committee [9] of randomly generated Support Vector Data Description (SVDD) [5] models. Each model is trained and tested on the entire pool U . The samples are then selected by majority vote. Subsequently there are two variants

of the algorithm.

The first variant consists in annotating the whole set of detected outliers, adding the annotated samples to the pool L , training a binary Support Vector Machine (SVM) [1] on L and applying the model to U . This process is iterated until a termination criterion is satisfied. The binary SVM is used in this variant just to measure the performance of the active learning method and does not get involved in the selection process of the samples to be annotated.

The second variant consists in involving the binary SVM in the selection process of the samples to be annotated. At each iteration, the binary SVM model trained at the previous iteration is tested on the set of uncharacteristic samples selected by the committee of SVDD models. Depending on the chosen query strategy, a subset of the detected outliers is annotated instead of the whole selection as in the previous variant. Those samples that are annotated are added to L , while the other samples are left in U . This process is iterated until a termination criterion is satisfied. Several query strategies are experimented with among others, random sampling, uncertainty sampling (Shannon entropy, learning at the border), and certainty sampling (farthest from the border) [2] [7] [8]. This variant of the algorithm helps the annotator by enabling him to choose the number of samples to be annotated at each iteration. This is not possible while using the first variant of the algorithm, since the amount of unannotated samples classified as uncharacteristic by each member of the committee is unforeseeable.

3 Discussion/Conclusion

The approach is currently being tested on a dataset consisting of 30 video sequences of 30 minutes each, depicting in each case an experiment involving a participant who interacts by speech and touch input with a multi-modal system. The experiment involves resolving a series of timed puzzles with increasing levels of difficulty. As an incentive, the participant can earn money based on his/her performance. The longer it takes to solve a puzzle, the less money is rewarded. These settings induce sporadic and spontaneous facial reactions from the participants during the interaction process. The objective is to enable the system to distinguish facial expressions characterized by the presence of observable facial gestures from those without any facial gestures.

So far a total of 6 participants was drawn from the dataset and manually annotated in order to have the ground truth needed in order to assess the performance of the approach. A binary SVM was then trained and tested on the annotated corpus of each participant and the results of the classification were used as baseline. Finally the described approach was tested on each selected participant's set and compared with the baseline. The first results of the undertaken experiments suggest that the proposed active learning approach performs as well as a system trained on the fully annotated corpus, while dramatically reducing the cost of annotation. Further experiments involving more participants are to be undertaken for a better assessment of the performance of the approach. Moreover the applicability of the approach to multimodal datasets should be investigated.

References

1. Abe, S.: Support Vector Machines for Pattern Classification. Springer, London, England (2005)
2. Angluin, D.: Queries revisited. In: Proceedings of the 12th International Conference on Algorithmic Learning Theory. pp. 12–31. Springer, London, UK (2001)
3. Chapelle, O., Schölkopf, B., Zien, A.: Semi-Supervised Learning. The MIT Press, Cambridge, Massachusetts (2006)
4. Cunningham, P., Cord, M., Delany, S.J.: Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval, chap. Supervized Learning, pp. 21–49. Springer, Berlin, Heidelberg (2008)
5. David, M.T., Robert, P.D.: Support vector data description. Machine Learning (2004)
6. Kächele, M., Schels, M., Meudt, S., Kessler, V., Glodek, M., Thiam, P., Tschechne, S., Palm, G., Schwenker, F.: On annotation and evaluation of multi-modal corpora in affective human-computer interaction. In: Böck, R., Bonin, F., Campbell, N., Poppe, R. (eds.) Multimodal Analyses enabling Artificial Agents in Human-Machine Interaction, pp. 35–44. Lecture Notes in Computer Science, Springer International Publishing (2015)
7. Roy, N., McCallum, A.: Toward optimal active learning through sampling estimation of error reduction. In: Proceedings of the Eighteenth International Conference on Machine Learning. pp. 441–448. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001)
8. Settles, B.: Active learning literature survey. Computer sciences technical report, University of Wisconsin–Madison (2009)
9. Seung, S.H., Opper, M., Sompolinsky, H.: Query by committee. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory. pp. 287–294. ACM, New York, NY, USA (1992)
10. Thiam, P., Meudt, S., Kächele, M., Palm, G., Schwenker, F.: Detection of emotional events utilizing support vector methods in an active learning hci scenario. In: Proceedings of the 2014 Workshop on Emotion Representations and Modelling for HCI Systems. pp. 31–36. ERM4HCI '14, ACM, New York, NY, USA (2014)
11. Zhao, G., Pietikinen, M.: Dynamic texture recognition using local binary patterns with an application to facial expressions. IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 915–928 (2007)

Predictable Feature Analysis

Stefan Richthofer and Laurenz Wiskott
{stefan.richthofer, laurenz.wiskott}@ini.rub.de

Institut für Neuroinformatik,
Ruhr-Universität Bochum

Abstract. Inspired by the idea of Slow Feature Analysis (SFA), we have developed an algorithm that selects features by predictability rather than slowness. While SFA has proven valuable in finding invariances and performing recognition tasks, the goal of Predictable Feature Analysis (PFA) is to perform planning and control-tasks, as the predictable features can be used to estimate the consequences of possible actions of an agent or controller. From SFA we adopt the problem-constraints to avoid trivial or repeated solutions. To define predictability we focus on a given prediction model but support a variety of models that fulfill certain requirements, using linear, autoregressive processes as the default model. Since fitting a prediction model on training-data is an optimization problem by itself, we face a hard to solve hen-and-egg-problem regarding model-fitting and selecting the best suitable features. In this work we provide a tractable algorithm to solve this nested problem with reasonable accuracy.

1 Introduction

Consider a typical reinforcement-learning setting, where an agent is placed in an environment and aims to achieve some goal. In contrast to most common discrete, board-game-like scenarios we consider a – more natural – continuous setup. The input is a high-dimensional, continuous signal over time, like vision or some other sensoric input.

PFA is intended as a tool to organize the vast amount of incoming data. Only data that helps to understand and manipulate the agent’s state in the desired way is considered useful. Allowing to predict outcomes of possible actions is a key-feature the agent’s learning model must provide. [1] gives an excellent review of former proposals to use predictable features for tasks like this.

Our work is strongly inspired by Slow Feature Analysis (SFA) – an algorithm that has proven valuable in several fields and problems concerning signal- and data analysis. It achieves a drastic, yet reasonable dimensionality reduction by focusing on slowly varying subsignals, so-called “slow features”. Since slowness usually indicates invariance and invariant problem representations are crucial for typical data-analysis and recognition tasks, many of these tasks become much more feasible after applying SFA. Examples are the self-organization of complex-cell receptive fields, the recognition of whole objects invariant to

spatial transformations, the self-organization of place-cells, extraction of driving forces, or nonlinear blind source separation. (see [2,3,4,5,6]).

Instead of slowness, PFA selects subsignals by predictability with respect to a certain prediction model that fulfills the criteria we give in Section 3. In contrast to model-independent notions of predictability like in the information bottleneck approach [7,8], PFA directly provides a usable prediction model together with the feature selection. However, simultaneous feature selection and model fitting is a difficult nested optimization problem that appears to be only addressable with a relaxed formulation that we describe in Section 2.

Using some notion of predictability for learning is of course not a new idea. ForeCA (Forecastable Component Analysis), an independently developed method, is based on the same paradigm as PFA, but proposes a model-independent approach [9]. A further difference is that PFA searches for well predictable Systems, while ForeCA selects best predictable single components. There also exists an ICA-based approach to predictability-driven dimensionality reduction, see [10].

2 Extracting predictable features

Given an input-signal $\mathbf{x}(t)$ with n components, our goal is to extract r well predictable output-components, referred to as “predictable features”. In order to measure predictability, we use a **linear, auto-regressive prediction** model, because it is simple, popular and has been successfully used in many fields for modeling temporal processes. Like in SFA, we optimize the parameters over a training phase Ω_t and also adopt the SFA constraints to avoid trivial or repeated solutions. We also allow for an optional, non-linear expansion \mathbf{h} (e.g. monomials of low degree). The first steps of PFA are indeed equal to those in SFA, i.e. we also apply a non-linear expansion and start with a sphering-step. Mean is defined using $\langle s(t) \rangle := \frac{1}{|\Omega_t|} \sum_{t \in \Omega_t} s(t)$ (average of a signal over the training phase). To fulfill the constraints, the expanded signal is sphered over the training-phase:

$$\tilde{\mathbf{z}}(t) := \mathbf{h}(\mathbf{x}(t)) - \langle \mathbf{h}(\mathbf{x}(t)) \rangle \quad (\text{apply expansion; make mean-free}) \quad (1)$$

$$\mathbf{z}(t) := \mathbf{S}\tilde{\mathbf{z}}(t); \mathbf{S} := \langle \tilde{\mathbf{z}}\tilde{\mathbf{z}}^T \rangle^{-\frac{1}{2}} \quad (\text{normalize covariance}) \quad (2)$$

A single signal component is regarded well predictable, if each value can be approximated by a linear combination of p recent values. Expressing this formally, we face the problem of finding vectors \mathbf{a} and \mathbf{b} such that

$$\mathbf{a}^T \mathbf{z}(t) \stackrel{!}{\approx} b_1 \mathbf{a}^T \mathbf{z}(t-1) + \dots + b_p \mathbf{a}^T \mathbf{z}(t-p) = \mathbf{a}^T \text{hist}_{\mathbf{z},p}(t) \mathbf{b} \quad (3)$$

with hist defined as the signal’s history over the recent p timesteps:

$$\text{hist}_{\mathbf{z},p,\Delta}(t) := \sum_{i=1}^p \mathbf{z}(t-i\Delta) \mathbf{e}_i^T = (L^1, \dots, L^p) \mathbf{z} \quad (4)$$

with $\mathbf{e}_i \in \mathbb{R}^p$, $(\mathbf{e}_1, \dots, \mathbf{e}_p) = \mathbf{I}_{p,p}$. Here $\mathbf{I}_{p,p}$ denotes the p -dimensional identity, thus \mathbf{e}_i denotes the i -th p -dimensional Euclidean unit vector. L denotes the lag operator (with Δ -length timesteps implied). Δ defaults to 1: $\text{hist}_{\mathbf{z},p} := \text{hist}_{\mathbf{z},p,1}$.

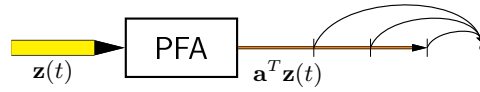


Fig. 1. Illustration of PFA

To extract multiple components, we define the extraction matrix $\mathbf{A}_r := (\mathbf{a}_1, \dots, \mathbf{a}_r) \in \mathbb{R}^{n \times r}$ and the reduced identity $\mathbf{I}_r \in \mathbb{R}^{n \times r}$ consisting of the first r Euclidean unit vectors as columns. Since \mathbf{z} is sphered, we can obtain uncorrelated components by constraining \mathbf{A}_r to be orthogonal, i.e. $\exists \mathbf{A} \in O(n): \mathbf{A}_r = \mathbf{A}\mathbf{I}_r$ where $O(n) \subset \mathbb{R}^{n \times n}$ denotes the space of orthogonal transformations, i.e. $\mathbf{A}\mathbf{A}^T = \mathbf{I}$. We denote the extracted signal with $\mathbf{m} := \mathbf{A}_r^T \mathbf{z}$.

The common way to extend (3) to multiple dimensions can be written as

$$\mathbf{m}(t) \stackrel{!}{\approx} \mathbf{B}_1 \mathbf{m}(t-1) + \dots + \mathbf{B}_p \mathbf{m}(t-p) \quad \text{with } \mathbf{B}_i \in \mathbb{R}^{n \times n}, \text{ diagonal} \quad (5)$$

In this form, it does not fulfill all criteria from Section 3 (it is not orthogonal agnostic for $n > 1$), which are crucial for PFA to work. Nevertheless, there are strategies to extract features that are predictable in terms of (5). It is straight forward to find closed-form solutions for \mathbf{a} and \mathbf{b} from (3), if one of the vectors is given. If \mathbf{b} is given, choose \mathbf{a} as the eigenvector corresponding to the smallest eigenvalue in

$$\langle \mathbf{z}\mathbf{z}^T \rangle - 2 \langle \mathbf{z}\mathbf{b}^T \text{hist}_{\mathbf{z},p} \rangle + \langle \text{hist}_{\mathbf{z},p} \mathbf{b}\mathbf{b}^T \text{hist}_{\mathbf{z},p}^T \rangle \quad (6)$$

If \mathbf{a} is given, choose \mathbf{b} as

$$\mathbf{b}^T := \langle \mathbf{z}^T \mathbf{a}\mathbf{a}^T \text{hist}_{\mathbf{z},p} \rangle \langle \text{hist}_{\mathbf{z},p}^T \mathbf{a}\mathbf{a}^T \text{hist}_{\mathbf{z},p} \rangle^{-1} \quad (7)$$

One can choose a starting-value and apply (6) and (7) in turns, but this usually runs into local optima. Our alternative approach is described below.

But first we proceed by refining (5) to be suitable for PFA:

$$\mathbf{m}(t) \stackrel{!}{\approx} \mathbf{B}_1 \mathbf{m}(t-1) + \dots + \mathbf{B}_p \mathbf{m}(t-p) \quad \text{with } \mathbf{B}_i \in \mathbb{R}^{n \times n} \quad (8)$$

The difference to the first formulation is that \mathbf{B} does not have to be diagonal, i.e. each extracted component's prediction may depend on all other extracted components. A massive advantage of model (8) is that we can initially fit it to our data in full dimension and search for the best-fitting components afterwards. We formalize the need for such a model-property in Section 3.

Let \mathbf{g} be a general prediction-model:

$$\mathbf{g} \in \mathcal{G} : \quad \mathbf{z}(t) \stackrel{!}{\approx} \mathbf{g}(\text{hist}_{\mathbf{z},p,\Delta}(t)) \quad (9)$$

where \mathcal{G} is the model-class, i.e. the set of possible realizations of \mathbf{g} . We measure the prediction error in an average least squares sense by

$$\text{err}(\mathbf{g}, \mathbf{z}) := \langle \|\mathbf{z} - \mathbf{g}(\text{hist}_{\mathbf{z}, p, \Delta})\|^2 \rangle \quad (10)$$

With $\mathbf{g}_{\mathbf{z}}^* := \text{argmin}_{\mathbf{g} \in \mathcal{G}} \text{err}(\mathbf{g}, \mathbf{z})$, we define the optimal error of \mathbf{z} :

$$\text{err}(\mathbf{z}) := \text{err}(\mathbf{g}_{\mathbf{z}}^*, \mathbf{z}) \quad (11)$$

To formalize (8) as a prediction model in this notation, we combine the coefficient-matrices \mathbf{B}_i to a single wide matrix and define $\mathbf{g}_{\mathbf{B}}$ and \mathcal{G}_{lin} :

$$\mathbf{B} := (\mathbf{B}_1, \dots, \mathbf{B}_p) \in \mathbb{R}^{n \times np} \quad (12)$$

$$\mathbf{g}_{\mathbf{B}}(\text{hist}_{\mathbf{m}, p}(t)) := \mathbf{B} \text{vec}(\text{hist}_{\mathbf{m}, p}(t)) \quad (13)$$

$$\mathcal{G}_{\text{lin}} := \{ \mathbf{g}_{\mathbf{B}} : \mathbf{B} \in \mathbb{R}^{n \times np} \} \quad (14)$$

By analytical optimization, we obtain the following regression formula to fit $\mathbf{g}_{\mathbf{B}} \in \mathcal{G}_{\text{lin}}$ to $\mathbf{m} = \mathbf{A}_r^T \mathbf{z}$:

$$\mathbf{B}_{\mathbf{z}}(\mathbf{A}_r) = \left(\mathbf{A}_r^T \langle \mathbf{z} \zeta^T \rangle \underline{\mathbf{A}}_r \right) \left(\underline{\mathbf{A}}_r^T \langle \zeta \zeta^T \rangle \underline{\mathbf{A}}_r \right)^{-1} \quad (15)$$

with $\zeta(t) := \text{vec}(\text{hist}_{\mathbf{z}, p}(t))$ and the following notation defined for any matrix \mathbf{A} :

$$\text{vec}(\mathbf{A}) := \begin{pmatrix} \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_n \end{pmatrix} \quad (\mathbf{a}_i \text{ columns of } \mathbf{A}) \quad \underline{\mathbf{A}} := \mathbf{I}_{p,p} \otimes \mathbf{A} = \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ & \ddots \\ \mathbf{0} & \mathbf{A} \end{pmatrix} \quad (16)$$

p times \mathbf{A}

PFA as described below relies on some criteria to work properly. In situations where these criteria are not met, PFA can still be applied, but is likely to produce suboptimal results. Using the iteration (6) vs. (7) on the result as a postprocessing step can improve this. So far this iteration was only described for single components though. While (15) is a multi-component generalization of (7), obtaining a multi-component variant of (6) is far more difficult. It would allow us to apply the iteration on extracted signal-systems and can be formulated as follows:

$$\underset{\mathbf{A} \in \mathcal{O}(n)}{\text{minimize}} \langle \|\mathbf{A}_r^T \mathbf{z} - \mathbf{B}^T \underline{\mathbf{A}}_r^T \zeta\|^2 \rangle \quad (17)$$

We can write the optimization term as

$$\text{vec}(\mathbf{A}_r^T)^T \langle (\mathbf{Z} \otimes \mathbf{I}) \tilde{\mathbf{B}} \tilde{\mathbf{B}}^T (\mathbf{Z}^T \otimes \mathbf{I}) \rangle \text{vec}(\mathbf{A}_r^T) \quad (18)$$

with $\mathbf{Z} := (\mathbf{z}, \text{hist}_{\mathbf{z}, p})$ and $\tilde{\mathbf{B}}^T := (\mathbf{I}, -\mathbf{B}^T)$. By some index-manipulation, the central constant matrix can be efficiently computed from $\langle \zeta \zeta^T \rangle$ and $\langle \mathbf{z} \zeta^T \rangle$, which PFA computed anyway for (15). An optimal $\text{vec}(\mathbf{A}_r^T)$ can be obtained as the eigenvector corresponding to the smallest eigenvalue of the central matrix. However, this would in general not yield an orthogonal \mathbf{A}_r , but we can project

it to the nearest orthogonal \mathbf{A}_r using singular value decomposition. Since the projection-error for the smallest eigenvalue can still render that result suboptimal, we take several small eigenvalues into account as candidates. This is usually sufficient to achieve a good improvement. Let us now return to the actual procedure. If $r = n$, $\mathbf{A} = \mathbf{I}$ and thus $\mathbf{A}_r = \mathbf{I}_r$, we write $\mathbf{W} := \mathbf{B}_z(\mathbf{I}) = \langle \mathbf{z}\zeta^T \rangle \langle \zeta\zeta^T \rangle^{-1}$. For $r = n$ and $\mathbf{A} \in O(n)$, we have $\mathbf{B}_z(\mathbf{A}) = \mathbf{A}^T \mathbf{W} \mathbf{A}$. Since \mathbf{z} is sphered, we can state the following compact notation of the PFA-problem:

$$\underset{\mathbf{A} \in O(n)}{\text{minimize}} \quad \text{err}(\mathbf{A}_r^T \mathbf{z}) \quad (19)$$

Inserting our default model, we have $\text{err}(\mathbf{A}_r^T \mathbf{z}) = \langle \|\mathbf{A}_r^T \mathbf{z} - \mathbf{B}_z(\mathbf{A}_r) \mathbf{A}_r^T \zeta\|^2 \rangle$. However, because (15) is an involved term, mainly due to the projection under the inversion symbol, (19) appears to be untractable by every method known to us³. Instead of solving it directly, we propose the following tractable relaxation:

$$\underset{\mathbf{A} \in O(n)}{\text{minimize}} \quad \langle \|\mathbf{A}_r^T \mathbf{z} - \mathbf{I}_r^T \mathbf{B}_z(\mathbf{A}) \mathbf{A}_r^T \zeta\|^2 \rangle = \langle \|\mathbf{A}_r^T (\mathbf{z} - \mathbf{W}\zeta)\|^2 \rangle \quad (20)$$

Informally speaking, problem (20) asks for components that are optimally predictable, if the prediction may be based on the entire input signal, rather than just on the extracted components themselves. From now on we denote a global optimum of (19) with \mathbf{A}_r^* and of (20) with $\mathbf{A}_r^{(0)}$.

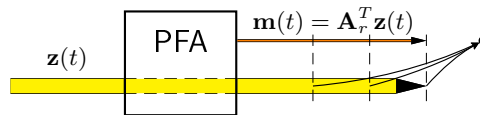


Fig. 2. Illustration of relaxation (20)

To solve (20) globally, we write it as

$$\underset{\mathbf{A} \in O(n)}{\text{minimize}} \quad \text{Tr} \left(\mathbf{A}_r^T \langle (\mathbf{z} - \mathbf{W}\zeta) (\mathbf{z} - \mathbf{W}\zeta)^T \rangle \mathbf{A}_r \right) \quad (21)$$

and choose \mathbf{A} such that it diagonalizes $\langle (\mathbf{z} - \mathbf{W}\zeta) (\mathbf{z} - \mathbf{W}\zeta)^T \rangle$ and sorts the r smallest eigenvalues to the upper left. (Use $\langle (\mathbf{z} - \mathbf{g}_z^*(\text{hist}_{z,p})) (\mathbf{z} - \mathbf{g}_z^*(\text{hist}_{z,p}))^T \rangle$ for a general model.) This procedure can be described as performing principal component analysis (PCA) on the residues of the prediction (but selecting the smallest eigenvalues). In Section 4 we prove that if $\text{err}((\mathbf{A}_r^*)^T \mathbf{z}) = 0$, then $\mathbf{A}_r^{(0)}$ is also a global solution of (19) (given that the model fulfills certain conditions).⁴

³ Not counting evolutionary and other inherent local optimization approaches, since we aim for the global solution. Experiments showed us, that locally optimal solutions are usually still of high error and of low relevance for the model.

⁴ Note that if $\mathbf{A}_r^{(0)}$ is used as solution for (19), the prediction model must be refitted to the reduced signal to get optimal prediction. For this, calculate $\mathbf{B}_z(\mathbf{A}_r^{(0)})$ as defined in (15).

More precisely speaking, the relaxation gap of (20) depends on $\text{err}((\mathbf{A}_r^*)^T \mathbf{z})$ in a continuous manner and is zero, if that error is zero. If the optimal subsignal has a significant prediction error, the solution obtained as $\mathbf{A}_r^{(0)}$ usually suffers from overfitting and is suboptimal for (19). The (multi-component) iterative algorithm described above can be used to improve results in such cases.

3 Criteria for suitable prediction models

In this section we discuss what properties of a prediction model are crucial to make the procedure described in Section 2 feasible.

Definition 1 (Orthogonal agnosticity criterion). *We say that a prediction-model \mathcal{G} is **orthogonal-agnostic** on Ω_t , if for every $\mathbf{A} \in \text{O}(n)$, $\mathbf{g} \in \mathcal{G}$:*

$$\text{err}(\mathbf{z}) = \text{err}(\mathbf{A}^T \mathbf{z}) \quad (22)$$

Equation (22) means that the model fits equally well to any orthogonal transformation of the data. In Section 4 we will need a more restrictive variant of this criterion, that additionally considers projections of the data to subspaces:

Definition 2 (Projective orthogonal agnosticity criterion). *We say, that a prediction-model \mathcal{G} is **projective orthogonal-agnostic** on Ω_t , if for every $\mathbf{A} \in \text{O}(n)$, $r \leq n$ the following holds:*

$$\langle \|\mathbf{A}_r^T \mathbf{z} - \mathbf{I}_r^T \mathbf{g}_{\mathbf{A}^T \mathbf{z}}^*(\mathbf{A}^T \text{hist}_{\mathbf{z},p})\|^2 \rangle = \langle \|\mathbf{A}_r^T \mathbf{z} - \mathbf{A}_r^T \mathbf{g}_{\mathbf{z}}^*(\text{hist}_{\mathbf{z},p})\|^2 \rangle \quad (23)$$

Note that for $r = n$, (23) simplifies to (22) and projective orthogonal agnosticity becomes equivalent to ordinary orthogonal agnosticity. An even stronger and very intuitive criterion is the following:

Definition 3 (Commuting with orthogonal transformations). *We say, that a prediction-model \mathcal{G} **commutes with orthogonal transformations**, if for every $\mathbf{A} \in \text{O}(n)$ the following holds:*

$$\mathbf{g}_{\mathbf{A}^T \mathbf{z}}^* = \mathbf{A}^T \mathbf{g}_{\mathbf{z}}^*. \quad (24)$$

This criterion implies projective and ordinary orthogonal agnosticity. To assure projective orthogonal agnosticity, it is a straightforward procedure to construct models such that they commute with orthogonal transformations.

Definition 4 (Information consistency criterion). *We say that a prediction-model \mathcal{G} is **information-consistent** on Ω_t , if for every $\mathbf{A} \in \text{O}(n)$, $r \leq n$ the following holds:*

$$\langle \|\mathbf{A}_r^T \mathbf{z} - \mathbf{g}_{\mathbf{A}^T \mathbf{z}}^*(\mathbf{A}_r^T \text{hist}_{\mathbf{z},p})\|^2 \rangle \geq \langle \|\mathbf{A}_r^T \mathbf{z} - \mathbf{A}_r^T \mathbf{g}_{\mathbf{z}}^*(\text{hist}_{\mathbf{z},p})\|^2 \rangle \quad (25)$$

An information-consistent model always benefits from more data rather than getting confused by it. For $r = n$, (25) follows from orthogonal agnosticity.

Theorem 1 \mathcal{G}_{lin} is projective orthogonal agnostic and information consistent.

Proof. Projective orthogonal agnosticity follows because the model commutes with orthogonal transformations, as $\mathbf{B}_z(\mathbf{A}) = \mathbf{A}^T \mathbf{B}_z(\mathbf{I}) \mathbf{A}$. A formal proof for information consistency of \mathcal{G}_{lin} is technically more challenging, but straight forward based on the provided calculus. However, we skip it here.

4 Relaxation Gap Theorem

Theorem 2 For any prediction model class \mathcal{G} that is projective orthogonal agnostic and information consistent, the following holds:

$$\text{If } \exists r \leq n, \mathbf{A}^* \in \mathcal{O}(n): \text{err}(\mathbf{A}_r^{*T} \mathbf{z}) = 0 \quad (26)$$

$$\text{and } \nexists \tilde{r} > r, \mathbf{A}^* \in \mathcal{O}(n): \text{err}(\mathbf{A}_{\tilde{r}}^{*T} \mathbf{z}) = 0 \quad (27)$$

$$\text{then } \text{err}(\mathbf{A}_r^{(0)T} \mathbf{z}) = 0 \quad (28)$$

(27) has the purpose to ensure, that the maximal r holding (26) is used in (28).

Proof. Define $\mathcal{O}(r, s) := \text{diag}(\mathcal{O}(r), \mathcal{O}(s))$ as the set of *space-partition preserving orthogonal transformations*, i.e. every $\tilde{\mathbf{A}} \in \mathcal{O}(r, s) \subset \mathcal{O}(r + s)$ has the form $\begin{pmatrix} \mathbf{A}_{rr} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{ss} \end{pmatrix}$ with $\mathbf{A}_{rr} \in \mathcal{O}(r)$ and $\mathbf{A}_{ss} \in \mathcal{O}(s)$. We derive that any global solutions $\mathbf{A}^{(0)}, \mathbf{B}^{(0)}$ of (20) can only differ by a transformation contained in $\mathcal{O}(r, s)$:

$$\exists \tilde{\mathbf{A}} \in \mathcal{O}(r, n - r): \mathbf{A}^{(0)} = \mathbf{B}^{(0)} \tilde{\mathbf{A}} \quad (29)$$

One can show that

$$\forall r \leq n, \mathbf{A} \in \mathcal{O}(n), \tilde{\mathbf{A}} \in \mathcal{O}(r, n - r): \text{err}(\mathbf{I}_r^T \mathbf{A}^T \mathbf{z}) = \text{err}(\mathbf{I}_r^T (\mathbf{A} \tilde{\mathbf{A}})^T \mathbf{z}) \quad (30)$$

This implies that solutions of (19) persist, if transformed by any $\tilde{\mathbf{A}} \in \mathcal{O}(r, n - r)$. By condition (26), we have $\text{err}(\mathbf{A}_r^{*T} \mathbf{z}) = 0$. By information consistency, it follows that $\langle \|\mathbf{A}_r^{*T} \mathbf{z} - \mathbf{I}_r^T \mathbf{g}_{\mathbf{A}^* \mathbf{z}}^*(\mathbf{A}^{*T} \text{hist}_{\mathbf{z}, p})\|^2 \rangle = 0$. So \mathbf{A}^* is a common global optimum of (19) and (20). By (29) it holds that: $\exists \tilde{\mathbf{A}} \in \mathcal{O}(r, n - r): \mathbf{A}^{(0)} = \mathbf{A}^* \tilde{\mathbf{A}}$. Finally by (30) we conclude that $\mathbf{A}^{(0)}$ must also be an optimum of (19).

By continuity arguments, the implication of theorem 2 extends to signals with components of low error – the lower the error is, the more precise we can find an optimum of (19) by solving (20).

5 Conclusion

With Predictable Feature Analysis we present a method to extract predictable features in a rigorously practical sense, directly providing a model that actually performs the prediction. While the algorithm is based on involved calculus, it boils down to completely classical methods in the end (Eigenvalue-, Singular

value decomposition), also inheriting computational cost and scalability from these. Note that parallelizable eigenvalue solvers exist, also designed to run efficiently on a Graphics Processing Unit (GPU), a highly parallelized processor type widely available on graphics cards.

Further we have provided some theory to describe the conditions under which PFA works best and can be used with other prediction models. We expect to find applications for PFA in the field of goal-driven control-tasks and reinforcement learning. For instance we plan to use PFA to obtain well predictable proto value functions for use in continuous navigation tasks.

Acknowledgment

The authors acknowledge support from the German Federal Ministry of Education and Research within the National Network Computational Neuroscience - Bernstein Fokus: "Learning behavioral models: From human experiment to technical assistance", grant FKZ 01GQ0951.

References

1. W. Bialek, I. Nemenman, and N. Tishby. Predictability, complexity, and learning. *Neural Comput*, 13:2409–2463, Nov 2001. [DOI:10.1162/089976601753195969] [PubMed:11674845].
2. Mathias Franzius, Niko Wilbert, and Laurenz Wiskott. Invariant object recognition and pose estimation with slow feature analysis. *Neural Computation*, 23(9):2289–2323, 2011.
3. Mathias Franzius, Henning Sprekeler, and Laurenz Wiskott. Slowness and sparseness lead to place-, head direction-, and spatial-view cells. In *Proc. 3rd Annual Computational Cognitive Neuroscience Conference, Nov. 1–2, San Diego, USA*, pages III–8, 2007.
4. Laurenz Wiskott. Estimating driving forces of nonstationary time series with slow feature analysis. arXiv.org e-Print archive, 2003.
5. Sven Daehne, Niko Wilbert, and Laurenz Wiskott. Self-organization of v1 complex cells based on slow feature analysis and retinal waves. In *Proc. Bernstein Conference on Computational Neuroscience, Sep 27–Oct 1, Berlin, Germany*, 2010.
6. Tobias Blaschke, Tiziano Zito, and Laurenz Wiskott. Independent slow feature analysis and nonlinear blind source separation. *Neural Computation*, 19(4):994–1021, 2007.
7. Felix Creutzig, Amir Globerson, and Naftali Tishby. Past-future information bottleneck in dynamical systems. *Physical Review E*, 79:041925, 2009.
8. Felix Creutzig and Henning Sprekeler. Predictive coding and the slowness principle: An information-theoretic approach. *Neural Computation*, 20(4):1026–1041, 2008.
9. Georg Goerg. Forecastable component analysis. In Sanjoy Dasgupta and David Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 64–72. JMLR Workshop and Conference Proceedings, May 2013.
10. Aapo Hyvaerinen. Complexity pursuit: Separating interesting components from time series. *Neural Computation*, 13(4):883–898, 2001.

Incremental learning of action models as HMMs over qualitative trajectory representations

Maximilian Panzner and Philipp Cimiano

Semantic Computing Group, CITEC, Bielefeld University,
mpanzner@cit-ec.uni-bielefeld.de
<http://www.sc.cit-ec.uni-bielefeld.de>

Abstract. In this paper we present an incremental approach to learning generative models of object manipulation actions as HMMs over qualitative relations between two objects. We compare the incremental approach against a traditional batch training baseline and show that the resulting qualitative action models are capable of one-shot learning after just one seen example while displaying good generalization behavior as more data becomes available.

Keywords: hidden markov models, model merging, action recognition, online classification, qualitative trajectory calculus

1 Introduction

Acquiring representations or models of actions is important for many embodied intelligent systems that need to act in the world. We present a system which incrementally learns action models as Hidden Markov Models (HMMs) over qualitative relations between two objects. The incremental nature of the model building process allows the system to learn and adapt continuously. The resulting action specific models are capable of one-shot learning after just one seen example while displaying good generalization behavior as more and more data becomes available. There has been a lot of work in the field of intelligent systems on developing formalisms for learning and representing actions, ranging from task-space representations [1] to high-level symbolic description of actions and their effects on objects [2]. With our system we aim at bridging the gap between low-level representations and high-level object manipulation plans in a way that facilitates transfer of learned concepts between the representational levels. On the one hand, we abstract away from continuous task-space values such as joint angles and Euclidian positions. On the other hand, in contrast to purely symbolic descriptions like logic oriented action descriptions, we model actions as distributions and are thus able to capture variation in action performance. As an intermediate-level representation between two objects, we chose to build on the qualitative trajectory calculus as a formal foundation, which discretizes the relative position and movement of the objects into qualitative relations. The temporal progression of action instances is modeled using HMMs.

As we target applications in the field of human robot interaction, where the human tutors expect the robot to be responsive toward new tasks or stimuli, we adopt an incremental model merging scheme to estimate the HMM parameters on-line [3].

2 Method

In this approach action models are represented as Hidden Markov Models (HMM) over sequences of qualitative relations between a trajector and a landmark expressed in the Qualitative Trajectory Calculus (QTC). This model was already successfully applied to a co-development task using both action and linguistic cues to emerge structured, joint representations of action performances along with the grammatical structure of natural language sentences describing the respective action [4].

2.1 Qualitative action models

To describe the relative position and movement between landmark and trajector we build on the qualitative trajectory calculus - double cross (QTC_{C_1}) [5] as a formal foundation. In general, QTC describes the interaction between two moving point objects k and l with respect to the reference line RL that connects them at a specific point t in time. The QTC framework defines 4 different subtypes as a combination over different basic relations between the two objects. As we only have one actively moved object in our experiments, we decided on QTC_{C_1} to give the best trade off between generalization and specificity of the qualitative relations. QTC_{C_1} consists of a 4-element state descriptor (C_1, C_2, C_3, C_4) where each $C_i \in \{-, 0, +\}$ represents a so called constraint with the following interpretation:

- C_1 Distance constraint: Movement of k with respect to l at time t_1 :
 - k is moving towards l
 - 0 k is not moving relative to l
 - + k is moving away from l
- C_2 Distance constraint: Movement of l with respect to k at time t_1 : same as above but with k and l interchanged
- C_3 Side constraint: Movement of k with respect to RL at time t_1 :
 - k is moving to the left-hand side of RL
 - 0 k is moving along RL or not moving at all
 - + k is moving to the right-hand side of RL
- C_4 Side constraint: Movement of l with respect to RL at time t_1 analogously to C_3

As the positions in our dataset were sampled at a fixed rate, we could have missed some situations where one or more state descriptor elements transition through 0. These discretization artifacts are compensated by inserting the missing intermediate relations one at a time from left to right. QTC_{C_1} is a rather

coarse discretization, leading to situations where the qualitative relation between the two objects can hold for a longer portion of the trajectory and is, due to the fixed rate sampling, repeated many times. Unlike many spatial reasoning systems, where repeating states are simply omitted, we use a logarithmic compression of repetitive subsequences:

$$|\hat{s}| = \min(|s|, 10 \ln(|s| + 1)) \quad (1)$$

where $|s|$ is the original number of repeated symbols in the sequence and $|\hat{s}|$ is the new number of repeated symbols. By applying this compression scheme we preserve information about the acceleration along the trajectory, which increases the overall performance especially for very similar actions like “jumps over” and “jumps upon”, while still allowing to generalize over high variations in relative pace of the action performances. The logarithmic compression of repetitive symbols in a sequence is in line with findings from psychophysics known as the Weber-Fechner law[6].

2.2 Learning qualitative action models

Differing from previous work [4], where we learned action models by batch training over all underlying trajectories, we now apply an incremental learning scheme utilizing the best first model merging [7,8] framework. Model merging is inspired by the observation that, when faced with new situations, humans and animals alike drive their learning process by first storing individual examples (memory based learning) when few data points are available and gradually switching to a parametric learning scheme to allow for better generalization as more and more data becomes available [9]. Our approach mimics this behavior by starting with simple models with just one underlying sequence, which evolve into more complex models generalizing over a variety of different sequences as more data becomes available.

The process to evolve simple models into complex ones relies on three basic operations. **Data incorporation** integrates a new observation sequence into an existing, possibly empty, model. **State merging** consolidates the resulting model in a way which allows it to generalize to yet unseen trajectories by intertwining paths corresponding to different action performances. **Model evaluation** approximates how well a given model fits its constituting dataset. This incremental learning scheme allows our models to display good generalization performance when faced with new samples while still being capable of one-shot learning after just one seen example. Learning, as generalization over the concrete observed examples, is driven by structure merging in the model in a way that we trade model likelihood against a bias towards simpler models, known as the Occam’s Razor principle, which among equally well predicting hypothesis prefers the simplest explanation requiring the fewest assumptions. As graphical models, HMMs are particularly well suited for a model merging approach because data incorporation, state merging and model evaluation are straightforward to apply in this framework and implemented as graph manipulation operations:

Data incorporation: When a new sequence is to be integrated into a given model we construct a unique path between the initial and the final state of the model where each symbol in the sequence corresponds to a fresh state in the new path. Each of these states emits its respective symbol in the underlying sequence and simply transitions to the next state with probability 1, yielding a sub path in the model which exactly reproduces the corresponding sequence.

State merging: The conversion of the memory based learning scheme with unique sub paths for each sequence in the underlying dataset into a model which is able to generalize to a variety of similar trajectories is achieved by merging states which are similar according to their emission and transition densities. Merging two states q_1 and q_2 means replacing these states with a new state \hat{q} whose transition and emission densities are a weighted mixture of the densities of the two underlying states.

Model evaluation: We evaluate the models resulting in the merging process using a mixture composed of a structural model prior $P(M)$ and the data dependent model likelihood $P(X|M)$:

$$P(M|X) = \lambda P(M) + (1 - \lambda)P(X|M) \quad (2)$$

The model prior $P(M)$ acts as a data independent bias. Giving precedence to simpler models with fewer states makes this prior the primary driving force in the generalization process:

$$P(M) = e^{-|M|}, \quad (3)$$

where the model size $|M|$ is the number of states in the model. It is also possible to include the complexity of the transitions and emissions per state. For our dataset we found that using only the number of states generates the best performing models. While the structural prior favors simpler models, its antagonist, the model likelihood, has its maximum at the initial model with the maximum likelihood sub-paths. The exact likelihood of the dataset X given the model M is computed as:

$$P(X|M) = \prod_{x \in X} P(x|M) \quad (4)$$

with

$$P(x|M) = \sum_{q_1 \dots q_l \in Q^l} p(q_I \rightarrow q_1) p(q_1 \uparrow x_1) \dots p(q_l \uparrow x_l) p(q_l \rightarrow q_F) \quad (5)$$

where l is the length of the sample and q_I, q_F denote the initial and final states of the model. The probability to transition from a state q_1 to q_2 is given as $p(q_1 \rightarrow q_2)$ and $p(q_1 \uparrow x_1)$ denotes the probability to emit the symbol x_1 while being in state q_1 . As we do not want to store the underlying samples explicitly, we use an approximation, which considers only the terms with the highest contribution, the Viterbi path:

$$P(X|M) \approx \prod_{q \in Q} \left(\prod_{q' \in Q} p(q \rightarrow q')^{c(q \rightarrow q')} \prod_{\sigma \in \Sigma} p(q \uparrow \sigma)^{c(q \uparrow \sigma)} \right) \quad (6)$$

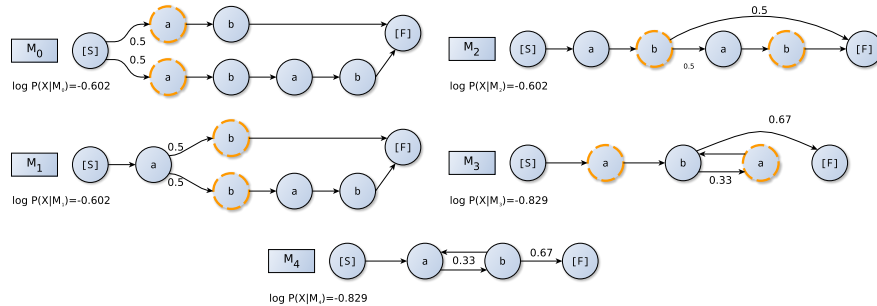


Fig. 1. Sequence of models obtained by merging samples from an exemplary language $(ab)^+$ and subsequently merging the highlighted states. Reproduced from [7].

where $c(q \rightarrow q')$ and $c(q \uparrow \sigma)$ are the total counts of transitions and emissions occurring along the Viterbi path associated with the samples in the underlying dataset (see [7] for details).

The simplest model in our approach is a model which simply produces a single sequence. These models are called maximum likelihood models because they produce their respective sequences with the highest possible probability. Starting from maximum likelihood models over individual sequences we build more general HMMs by merging simpler ones and iteratively joining similar states to intertwine sub-paths constructed from different sequences, allowing them to generalize across different instances of the same action class. The first model M_0 of the example in figure 1 can be seen as a joint model of two maximum likelihood sequences $\{ab, abab\}$. When generating from such a model, the actual sequence which will be generated is determined early by taking one of the possible paths emanating from the start state. Only the transitions from the start state display stochastic behavior, the individual sub-paths are completely deterministic and generate either ab or $abab$. Intertwining these paths is done through state merging, where we first build a list of possible merge candidates using a measure of similarity between state emissions and transition probability densities. In this approach we use the symmetrized Kullback-Leibler (KL) divergence. Then we greedily merge the best pair of states and re-evaluate the model likelihood. In the example above, the first two merges lead to model M_3 where we experienced a drop in log likelihood from -0.602 to -0.829 . We continue the merging process until we reach a point where merging more states would deteriorate the model likelihood to a point where it is no longer compensated by the prior favoring simpler models (eq. 3). The final model M_4 is now able to generate the whole set of sequences from the exemplary language $(ab)^+$ the two initial samples where drawn from.

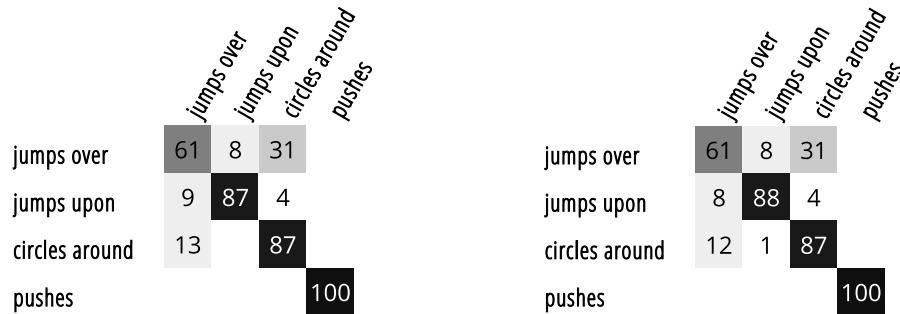


Fig. 2. Averaged class confusion matrix for the batch (left) and the incremental (right) approach.

3 Experiments

3.1 Dataset

To acquire a dataset we implemented a simple game where the test subjects were asked to perform an action with two objects according to a given instruction. The game screen was divided into two parts. The upper part was the actual gamefield with the two freely movable objects and below the gamefield was a textfield, where the test subjects could see the instruction describing the desired action performance. We had a total of 12 test subjects yielding a dataset with 1200 sample trajectories balanced over the four action classes “jumps over”, “jumps upon”, “circles around” and “pushes”. See [4] for a complete description of the dataset.

3.2 Batch vs. Incremental

To validate the incremental model building scheme we evaluate the performance of the incremental process against a traditional Baum Welch parameter estimation baseline. We consider a setting where the system is trained using recorded action performances from 11 test subjects and is then presented with a 12th set of action trajectories recorded from a new performer. Following this scheme the dataset is partitioned into 12 folds with 1100 training examples and 100 test

	F_1	precision	recall	σ
Batch	0.86	0.82	0.90	0.11
Incremental	0.86	0.82	0.90	0.12

Table 1. Results of the 12-fold cross-validation for the batch and the incremental approach.

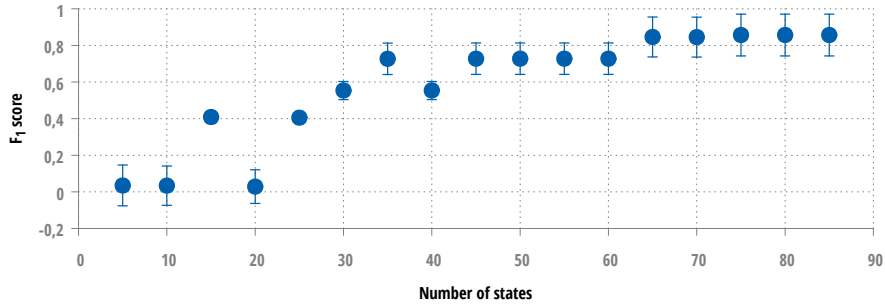


Fig. 3. Development of F_1 scores as the number of hidden states is increased for the batch training approach. The errorbars indicate the standard deviation of the F_1 scores across the 12 folds.

examples for each fold. We trained one HMM for each of the four action classes. Test sequences were classified according to the action specific HMM having the highest probability to have produced the respective sequence. Both approaches showed almost identical results (see table 1) with both having an averaged F_1 score of 0.86. The only difference is a slightly higher standard deviation between the results of the 12 folds for the incremental approach. The class confusion matrices in figure 2 show again only slight differences for both approaches. A noteworthy observation here is that the class confusions are not symmetric.

3.3 Parameter Sensitivity

In this experiment we evaluate the sensitivity of the batch and the incremental approach to their respective free parameters. As can be seen in figure 3 the classical Baum-Welch approach to HMM parameter estimation is highly sensitive to its structural parameter, the number of hidden states. F_1 scores range from 0.03 to 0.86 as the number of hidden states in the model is increased. The incremental approach on the other hand displays an almost linear response over

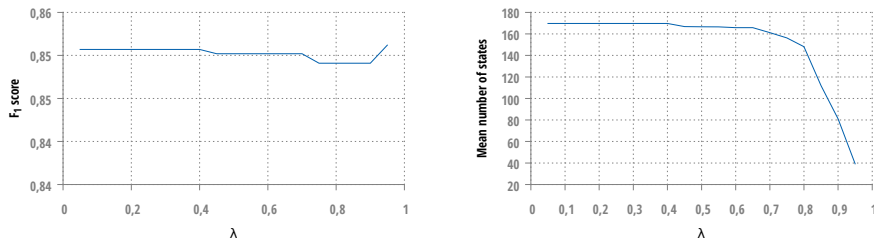


Fig. 4. Sensitivity to the λ parameter of the incremental approach.

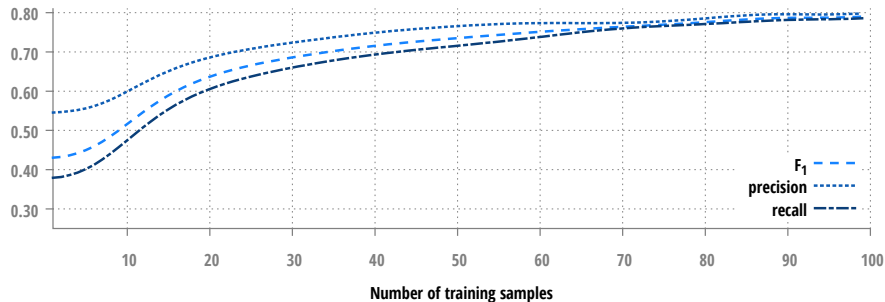


Fig. 5. Early learning behavior of the incremental model. The model performs notably well even with a single training example.

the whole range of values for the λ parameter (equation 2). While the F_1 score is stable with respect to λ , the number of hidden states in the model decreases drastically from 170 to 39 (at $\lambda = 0.95$). Setting $\lambda = 1$ and thus evaluating the model quality only by the prior favoring simpler models with less states leads to poorly performing models with just a single hidden state.

3.4 Early learning behavior

In this experiment we evaluate the one-shot learning capability of the incremental model. We prepared training sets with only 1 to 100 examples per action class and evaluated the classification performance as in section 3.2. As can be seen in figure 5 the precision is with 54% already notably good after the classifier had seen only one training example per action class.

3.5 Early classification behavior

To evaluate how well the system performs when it is presented with incomplete sequences, we trained the system as in section 3.2 but truncated the test sequences. As can be seen in figure 3.5, the sequences for the push action are rather distinctive. After the classifier has seen only 5% of the sequence, over 70% of the respective sequences are correctly classified as belonging to that action class. The “circles around” and “jumps over” actions get classified rather late in the sequence because a jumping trajectory could easily look like the start of a circles around action.

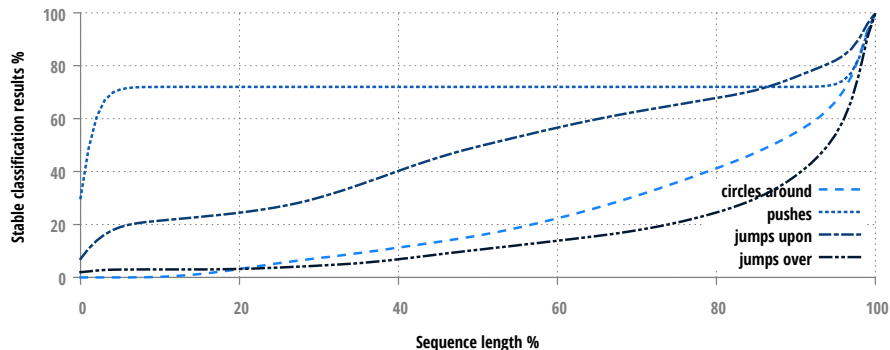


Fig. 6. Early sequence classification behavior. The x-axis represents the length of the sequence presented to the classifier and the y-axis the percentage of sequences for which their classification result will not change if more of the sequence was presented.

References

1. Komei Sugiura, Naoto Iwahashi, Hideki Kashioka, and Satoshi Nakamura. Learning, generation and recognition of motions by reference-point-dependent probabilistic models. *Advanced Robotics*, 25(6-7):825–848, 2011.
2. Moritz Tenorth and Michael Beetz. KnowRob – Knowledge Processing for Autonomous Personal Robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4261–4266, 2009.
3. Andreas Stolcke and Stephen Omohundro. Hidden markov model induction by bayesian model merging. *Advances in neural information processing systems*, pages 11–11, 1993.
4. Maximilian Panzner, Judith Gaspers, and Philipp Cimiano. Learning linguistic constructions grounded in qualitative action models. *IEEE International Symposium on Robot and Human Interactive Communication*, 2015.
5. Nico Weghe, Bart Kuijpers, Peter Bogaert, and Philippe Maeyer. A Qualitative Trajectory Calculus and the Composition of Its Relations. *GeoSpatial Semantics SE - 5*, 3799(Dc):60–76, 2005.
6. Thomas Bruss and Ludger Rüschemdorf. On the perception of time. *Gerontology*, 56(4):361–370, 2010.
7. Stephen Omohundro. Best-first model merging for dynamic learning and recognition. In *Advances in Neural Information Processing Systems 4*, pages 958–965. Morgan Kaufmann, 1992.
8. Andreas Stolcke and Stephen Omohundro. Inducing probabilistic grammars by bayesian model merging. In *Grammatical inference and applications*, pages 106–118. Springer, 1994.
9. Roger N Shepard. Toward a universal law of generalization for psychological science. *Science*, 237(4820):1317–1323, 1987.

This research/work was supported by the Cluster of Excellence Cognitive Interaction Technology 'CITEC' (EXC 277) at Bielefeld University, which is funded by the German Research Foundation (DFG).

On the Applicability of Recurrent Neural Networks for Pattern Recognition in Electroencephalography Signals

Marcel Binz, Sebastian Otte, and Andreas Zell

Cognitive Systems Group
University of Tuebingen
Tuebingen, Germany

Abstract. Brain-Computer Interfaces (BCI) define a direct communication interface between brains and external devices, computers or manipulators. Most BCI require a classification of Electroencephalography (EEG) signals. This research investigates the application of Recurrent Neural Networks (RNN) in this domain, with a focus on the learning properties of Dynamic Cortex Memories (DCM), an extension of the Long Short-Term Memory (LSTM) model. Methods are evaluated on a cued motor imagery task, where subjects have to imagine different movements over a fixed period of time. While the investigated recurrent architectures do not outperform state-of-the-art methods in terms of final accuracy, they provide several important advantages, like their ability to predict just-in-time, the absence of a time window in the recall phase, and their adaptability. Conclusively, the applied methodology exhibits the promising potential of RNN based BCI systems.

Keywords: EEG, Recurrent Neural Network, Dynamic Cortex Memory, Long Short Term Memory

1 Introduction

Brain-Computer Interfaces (BCI) allow for direct communication between the brain and external devices by specifying a mutual language. This requires an interpretation of brain signals in terms of classification. Here we analyse the application of *Recurrent Neural Networks* (RNN) for such signals captured with *Electroencephalography* (EEG).

State-of-the-art BCIs use μ - and β -rhythms to derive the desired mapping. A time window is specified, over which variances of a band-pass filtered signal are computed. These are used as input for *Support Vector Machines* (SVM) or linear models, like *Linear Discriminant Analysis* (LDA). Discriminability between two classes is increased by the *Common Spatial Pattern* (CSP) algorithm, which maximizes variance in one condition, while minimizing it in the other, achieving information transfer rates of 50 bits/min [3].

While traditional supervised methods assume an independent and identical distribution of the data set, this assumption does not hold for EEG measurements, since they are clearly correlated over time. RNNs offer a way to model

these dependencies. They are, in theory, able to recognize and learn relevant temporally encoded patterns automatically. Thus, no information will be lost by averaging over time windows and information will not be restricted to the same window. The CSP algorithm assumes a jointly Gaussian-distributed source activation and known frequency bands, possibly limiting its power and restricting its use cases. In an optimal scenario RNNs would work on raw data and detect even unknown properties automatically.

EEG signals have been applied as inputs for RNNs in multiple domains, for example the prediction of epileptic seizures [14]. Gueler et al. [8] employ Lyapunov exponents for signal classification in combination with EEG measurements. Probably most relevant to our work is classification of imagined mental tasks with Elman RNNs [5]. LSTMs in particular were utilized to detect lapse and achieved high temporal resolution [2].

This research, however, focuses on *Dynamic Cortex Memory* (DCM) based RNNs [13]. The DCM model, an extension of the well-established *Long Short-Term Memory* (LSTM) [9, 6, 7], provides an architectural modification, which facilitates faster convergence and a more stable behavior during training. Especially, results presented in [12] indicating their superiority on suppressing high-frequency noise, make them interesting for the problem scenario given in this paper due to the noisy nature of EEG signals.

The remainder of this paper is structured as follows: Section 2 provides foundations on BCI systems and relevant signal properties. In Section 3 RNNs, and particularly the DCM architecture, are briefly recapitulated. Then, in Section 4, the data set is introduced and the experimental setup is outlined. The results achieved are presented and discussed in Section 5. Finally, Section 6 concludes major aspects and observations, and motivates future studies.

2 Brain Computer Interfaces and Electroencephalography

While computers are able to exchange information with each other at the speed of over one terabits/sec, information transfer rates between humans and computers reach a maximum of 50 bits/sec [16, 4]. BCI address this communication bottleneck by providing a direct communication channel between the brain and an executing device. To achieve this they establish a new mutual language between both interactors. Developing BCIs will have major impacts on the way human interaction with computers works. The current state of research is still far away to be used universally in practice, but there are target groups, which could receive immediate benefits. Restoring motor functions in locked-in patients or monitoring in high risk situations are the first use cases that come to mind. Additionally BCIs have the possibility to provide better understanding of brain pattern.

2.1 The BCI Pipeline

Typical BCIs include multiple processing steps (Figure 1), starting with capturing the signal. For this purpose EEG is most prevalent. In EEG multiple

electrodes are placed on the scalp to measure corresponding electrical activities. Due to the existence of functional maps their spatial location provides useful information. Once digitalized, preprocessing methods are applied to increase the signal-to-noise ratio. In the last step a translation algorithm generates a mapping from the transformed signal to a command.

In some BCIs these algorithms are stationary and rely on the users to learn a voluntarily regularization of their brain activity in order to archive control, while others employ machine learning algorithms to adapt themselves to already present pattern.

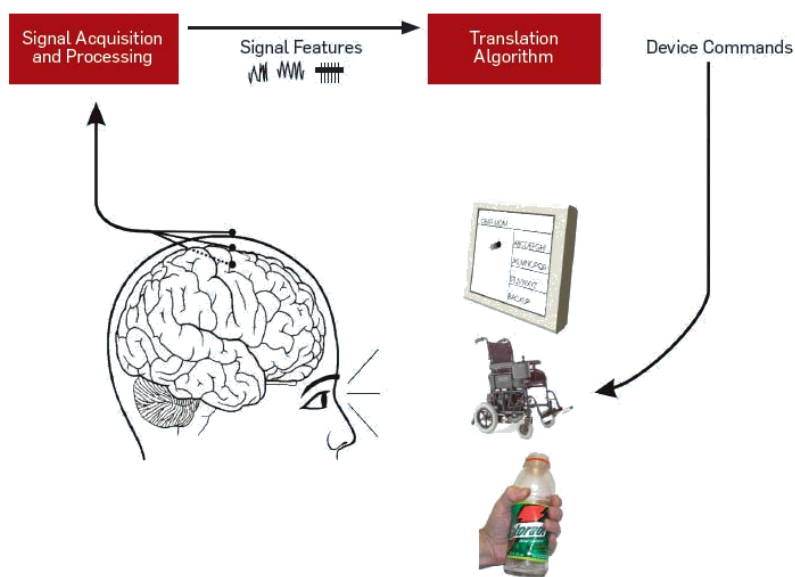


Fig. 1. Processing steps of a BCI. Captured signals are transformed with preprocessing methods. These are then passed on to a classifier, which generates the task specific mapping, leading to device commands [11].

2.2 Motor Imagery Tasks

Motor imagery tasks are part of the BCI framework. In their experimental setup a subject has to imagine a movement out of a predefined set of movements. In the synchronous version the length of the imagination is fixed and known in advance, while the asynchronous version discards this constraint. It is known, that movements or only their imagination are indicated by decreased μ - and β -rhythms (see Figure 2). This process is referred to as *Event-Related Desynchronization* (ERD).

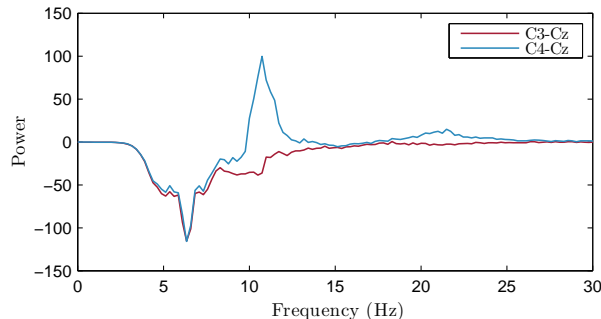


Fig. 2. Average power of the fourier transformed signal in one condition. The blue line corresponds to the difference between C3 and Cz, the red one to the difference between C4 and Cz. A decrease in μ - and β -rhythms is observed in one electrode, while it is absent in the other.

3 Dynamic Cortex Memory

With the introduction of the Long Short-Term Memory (LSTM) [9], especially Recurrent Neural Networks (RNNs), enjoy again emerging popularity. Due to their cyclic connections RNNs can learn temporal dependencies in data sequences. In contrast to standard RNNs, networks containing LSTM blocks [9], which are themselves a special kind of a recurrent network, that can be seen as differentiable memory cells, are even able to deal with long time-lag problems. LSTMs provide additional abilities. These are, for instance, that they are able to learn highly non-uniformly compressed sequences, where the position of important input events is difficult to predict or the adequate recognition of even noisy sequences.

However, in this paper recently introduced *Dynamic Cortex Memory* (DCM) networks [13] are used. A DCM block is in principle an LSTM block with *forget gates* [6] and *peep-hole connections* [7], but has several novel weighted connections within each block, i.e. connections from each gate to each other gate and a self-recurrent connection for each gate (see Figure 3). As in LSTM networks the gradient is computed using *Back Propagation Through Time* (BPTT). Nonetheless, even if there are more connections per block, DCM networks tend to require less blocks and, thus, less weights than LSTM networks to achieve similar or even better results. DCMs were shown to converge faster than LSTMs during training and behave more stably [13, 12].

4 Experimental Setup

The present data set [10] consists of a 2-class motor imagery task and was part of the BCI Competition IV [1]. Classes include a left and a right movement and

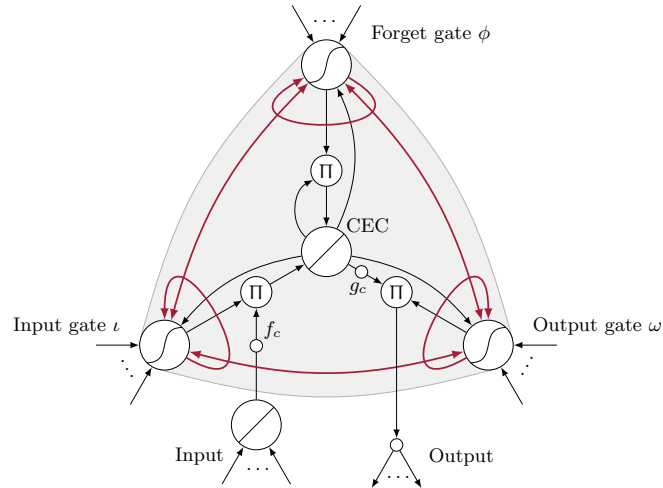


Fig. 3. Illustration of a DCM block [12]. Like LSTMs, DCMs consist of at least one memory cell that is controlled by three gates. Additionally, DCMs provide an inter-gate communication infrastructure that connects each gate with each other gate. This includes also a self-recurrent connection for each gate.

corresponding cues were presented synchronously. EEG data of nine subjects was collected. All were right-handed and had normal or corrected-to-normal vision. For each subject five sessions were recorded. The first two sessions were recorded normally, while the last three contained feedback in form of a smiley face. The last two sessions are used to evaluate submission and their true labels were not distributed until the end of the competition. To ensure a fair comparison they are also not used for training in this work.

During all sessions subjects were placed in an armchair and had to watch a monitor with a distance of approximately one meter. *Electro-oculargraphy* (EOG) activation was recorded separately prior to the actual task. Three EEG channels were measured with a sampling frequency of 250 Hz.

Before the first session each subject individually selected a movement, which they could imagine best (for example pulling a brake or squeezing a ball). Due to their correspondence with evaluation sessions, we will only use the third session for training. A session consisted of four runs with 40 trials (20 for each class). At the start of each trial a fixation cross followed by a short warning sound was presented. After the fixation cross disappeared, a visual cue was shown. Feedback started 0.5 seconds later. The subjects were instructed to move the smiley to the respective side by imagining the corresponding movement. Positive feedback was provided by a green smiley, negative by a red one. The cue vanished at 7.5 seconds, which results in a feedback period of four seconds. A break between one and two seconds was added after each trial.

5 Results and Discussion

Entries from the competition were required to provide predictions for each recording step. From all these Cohen's Kappa, as given in Equation (1), is used to evaluate their performance. To ensure comparison we will stick to this measure.

$$\mathcal{K} = \frac{p_0 - p_c}{1 - p_c} \quad (1)$$

Here, p_0 is the observed chance and p_c the random chance for an agreement. Note that the presented results are average Kappa values calculated for entire output sequences.

EOG artifacts were removed for all subjects except for subject 5, because of missing EOG recordings in evaluation sessions, by a method based on regression analysis [15]. Mutual Information was used to locate subject specific μ - and β -bands. Fifth-order Butterworth filters extracted information from the two selected bands. Band power, corresponding to its squared value, of the transformed signal was obtained and used as input for the network. Resulting networks consisted of six input units, one hidden layer with eight DCM units and a softmax output layer.

80% of the data were used for training, 20% were separated from the training set and used for validation. Supervised feedback was provided in terms of a target sequence with length corresponding to the input, which does not only optimize classification for the sequence as a whole during the learning process, but also for previously unlearned parts. Training with sequential targets has the drawback of many adjacent learning iterations of the same class. Due to this, batch learning was used to derive gradients. A decay rate of 0.01 and a momentum term of 0.9 were set. A 3×4 cross-validation between the learning rate (10^{-6} , $3 \cdot 10^{-7}$ or 10^{-7}) and the selection of the training interval (0 – 4s, 0.5 – 3.5s, 0.5 – 2.5s or 1.5 – 3.5s) was performed and the best network based on the validation set was chosen. During this process subjects 2 and 3 showed unsatisfying performances, thus their networks were trained using wider frequency bands and data from all available subjects.

Figure 4 plots the average development of correct predictions over the course of a trial. Starting with guessing at chance level the networks are able to detect patterns and to transmit them over a whole trial. This process leads to steadily improved predictions. It is to mention that the winner of the competition used a two second delay for classification, which limits usage in online BCI. Compared to this, RNNs are always able to perform predictions just-in-time. It is still straight forward to incorporate a delay in the network and thus improve its performance, see Figure 4 for a realization.

Table 1 shows the final results of the competition in 2008. While not placing in the top ranks, the RNNs achieve competitive results, averaging Kappa values of 0.43 without delay and 0.53 with induced delay.

Usage of DCM networks allows to discard several constraints and requires less a priori knowledge. They do not require the fixation of a time window in advance, which provides more flexibility. Additionally their results are independent of

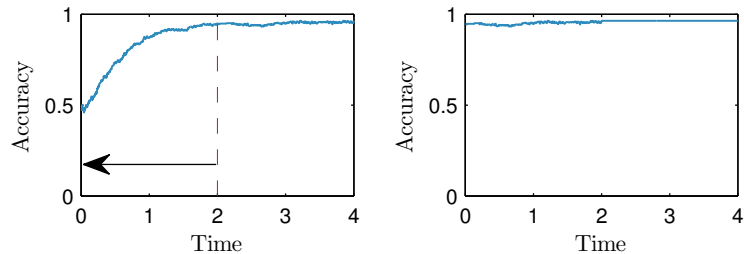


Fig. 4. Both plots display the average percentage of correct predictions for one subject over all time points. The left plot consists of un-delayed outputs, here accuracy remains at chance level at the beginning. Over time the network is able to identify pattern and accumulate them. At the end of a trial a nearly perfect prediction is reached. Delaying the prediction by a two second window, as indicated in the left plot, leads to the accuracies in the right plot.

Rank	Contributor	Kappa	Approach
1	Chin et al.	0.60	Filter Bank CSP & Naive Bayes Parzen Window Classifier
2	Gan et al.	0.58	CSSD & LDA
3	DCM with 2s delay	0.53	DCM with 2s delay
4	Coyle et al.	0.46	CSP, NTSP & SVM or LDA
5	DCM without delay	0.43	DCM without delay
6	Lodder et al.	0.43	Wavelet packet transform & LDA
7	Saa	0.37	Spectral features & ANN
8	Ping et al.	0.25	Bandpower & Bayesian LDA

Table 1. Final results of all submissions for data set 2b in the BCI Competition IV [1], including those archived by DCM networks in this work.

variants of the CSP algorithm. They are always able to work online, which extends the number of possible use-cases.

6 Conclusion

In this work we examined the applicability of RNNs in a 2-class motor imagery task measured by EEG. Presented networks proved themselves as solid classifiers for this data set, reaching nearly state-of-the-art level. RNNs do not presuppose an independent and identically distribution of the data set, allowing them to retrieve additional information in the case of temporally correlated patterns, which are clearly present in brain signals. Their property to work just-in-time without requiring any additional adaptations could turn out to be powerful, especially in asynchronous or other sophisticated setups.

Additional feature engineering and incorporating adaptive filtering directly into the networks could further increase their performance. Applying them to more complex tasks could verify outlined hypothesis regarding their flexibility.

References

1. Blankertz, B.: BCI Competition IV, <http://www.bbci.de/competition/iv/>
2. Davidson, P.R., Jones, R.D., Peiris, M.T.R.: Eeg-based lapse detection with high temporal resolution. *Biomedical Engineering, IEEE Transactions on* 54(5), 832–839 (2007)
3. Dornhege, G.: *Toward Brain-computer Interfacing*. A Bradford book, MIT Press (2007)
4. Fitts, P.M.: The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology* 47(6), 381–391 (Jun 1954)
5. Forney, E.M., Anderson, C.W.: Classification of eeg during imagined mental tasks by forecasting with elman recurrent neural networks. In: *Neural Networks (IJCNN), The 2011 International Joint Conference on*. pp. 2749–2755. IEEE (2011)
6. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: Continual prediction with LSTM. *Neural Computation* 12, 2451–2471 (1999)
7. Gers, F.A., Schraudolph, N.N., Schmidhuber, J.: Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research* 3, 115–143 (2002)
8. Güler, N.F., Übeyli, E.D., Güler, İ.: Recurrent neural networks employing lyapunov exponents for eeg signals classification. *Expert Systems with Applications* 29(3), 506–514 (2005)
9. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. *Neural computation* 9(8), 1735–1780 (1997)
10. Leeb, R., Lee, F., Keinrath, C., Scherer, R., Bischof, H., Pfurtscheller, G.: Brain-computer communication: motivation, aim, and impact of exploring a virtual apartment. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on* 15(4), 473–482 (2007)
11. McFarland, D.J., Wolpaw, J.R.: Brain-computer interfaces for communication and control. *Commun. ACM* 54(5), 60–66 (May 2011), <http://doi.acm.org/10.1145/1941487.1941506>
12. Otte, S., Liwicki, M., Zell, A.: An analysis of Dynamic Cortex Memory Networks (2015)
13. Otte, S., Liwicki, M., Zell, A.: Dynamic Cortex Memory: Enhancing Recurrent Neural Networks for Gradient-Based Sequence Learning. In: Wermter et al., S. (ed.) *Artificial Neural Networks and Machine Learning ICANN 2014*, pp. 1–8. No. 8681 in *Lecture Notes in Computer Science*, Springer International Publishing (Sep 2014)
14. Petrosian, A., Prokhorov, D., Homan, R., Dasheiff, R., Wunsch, D.: Recurrent neural network based prediction of epileptic seizures in intra-and extracranial eeg. *Neurocomputing* 30(1), 201–218 (2000)
15. Schlögl, A., Keinrath, C., Zimmermann, D., Scherer, R., Leeb, R., Pfurtscheller, G.: A fully automated correction method of eeg artifacts in eeg recordings. *Clinical neurophysiology* 118(1), 98–104 (2007)
16. Shannon, C.E.: Prediction and entropy of printed english. *Bell System Technical Journal* 30, 50–64 (Jan 1951)

Population Monte Carlo Meets Contrastive Divergence Learning

Oswin Krause¹, Asja Fischer², and Christian Igel¹

¹ Department of Computer Science, University of Copenhagen, Denmark

² Department of Computer Science and Operations Research, Université de Montréal, Canada

1 Background

Estimating the log-likelihood gradient with respect to the parameters of an undirected graphical model, such as a Restricted Boltzmann Machine (RBM) [1–3], is a challenging task. As the analytic calculation of the gradient is computationally unfeasible, it is often approximated using Markov Chain Monte Carlo (MCMC) techniques. In k -step Contrastive Divergence (CD- k) learning [2], the algorithm most often used for RBM training in practice, a Markov chain is initialized with a sample from the training dataset, and only a small number of k steps of Gibbs-sampling are performed. This results in a considerable bias of the gradient approximation [4, 5], which is difficult to detect. To reduce the bias, the use of persistent Markov chains or alternative sampling techniques such as parallel tempering were proposed (see the review [3] and referencers therein). However, these techniques can not exploit parallelization in the same way and can not be implemented as efficiently as CD- k .

2 Population Monte Carlo and CD

We introduce an Importance Sampling (IS) based approach for bias reduction inspired by the Population Monte Carlo method [6]. Assume our goal is to estimate the expectation $E_{p(\mathbf{x})} \{f(\mathbf{x})\}$, where $p(x) = \frac{1}{Z} \tilde{p}(x)$ is the intractable distribution of the model (e.g., the RBM) with unknown normalisation constant Z and $f(x)$ is an arbitrary function using the samples x (e.g., the log-likelihood gradient). Let $q(x)$ be the distribution of the samples given by starting the Markov chain from a random sample from the training dataset and running it for k steps. Using samples of q directly (as in CD- k) will lead to the biased estimate $E_{q(\mathbf{x})} \{f(\mathbf{x})\}$. Let us now consider a known proposal distribution $\kappa(\mathbf{x}'|\mathbf{x}) > 0$ which allows to sample some \mathbf{x}' conditioned on a sample \mathbf{x} from q . We can now perform IS using the equality

$$E_{p(\mathbf{x})} \{f(\mathbf{x})\} = E_{q(\mathbf{x})} \left\{ E_{\kappa(\mathbf{x}'|\mathbf{x})} \left\{ \frac{p(\mathbf{x}')}{\kappa(\mathbf{x}'|\mathbf{x})} f(\mathbf{x}') \right\} \right\} . \quad (1)$$

As the normalization constant Z from $p(x)$ is unknown, we calculate $\tilde{p}(x)$ and estimate Z . This leads to the *self-normalized* IS estimator

$$E_{p(\mathbf{x})} \{f(\mathbf{x})\} \approx \sum_{i=1}^N \frac{\omega_i f(\mathbf{x}_i)}{\sum_{j=1}^N \omega_j}, \quad (2)$$

based on N samples, where $\mathbf{x}_i \sim q(\cdot)$, $\mathbf{x}'_i \sim \kappa(\cdot|\mathbf{x}_i)$, and $\omega_i = \tilde{p}(\mathbf{x}'_i)/\kappa(\mathbf{x}'_i|\mathbf{x}_i)$. For RBMs a natural choice of κ are the tractable conditional distributions used as transition operator in Gibbs-sampling.

The main problems of the proposed approach are the well-known issues of IS. First, if the proposal distribution κ is too dissimilar from p , the estimator will exhibit a large variance. Second, the estimation of Z by $\frac{1}{N} \sum_i \omega_i$ leads to a bias in the estimator. Furthermore, if the number of samples is small compared to the complexity of the RBM, samples will not represent the distribution well enough, which makes this approach more suited for RBMs with a small number of neurons.

3 Preliminary Results

Our preliminary experiments suggest that the proposed technique works very well for RBMs with a small number of hidden neurons, outperforming CD- k and even parallel tempering. However, when the number of hidden units is large we see no advantage of the new method which may be explained by the fact that the proposal distribution gets too dissimilar to the RBM distribution and, thus, the IS-based estimate becomes too unreliable. Therefore, future work has to focus on finding a good proposal distribution.

References

1. Smolensky, P.: Information processing in dynamical systems: Foundations of harmony theory. In Rumelhart, D.E., McClelland, J.L., eds.: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1: Foundations. MIT Press (1986) 194–281
2. Hinton, G.E.: Training products of experts by minimizing contrastive divergence. *Neural Computation* **14** (2002) 1771–1800
3. Fischer, A., Igel, C.: Training restricted Boltzmann machines: An introduction. *Pattern Recognition* **47** (2014) 25–39
4. Fischer, A., Igel, C.: Empirical analysis of the divergence of Gibbs sampling based learning algorithms for Restricted Boltzmann Machines. In Diamantaras, K., Duch, W., Iliadis, L.S., eds.: *International Conference on Artificial Neural Networks (ICANN 2010)*. Volume 6354 of LNCS., Springer-Verlag (2010) 208–217
5. Fischer, A., Igel, C.: Bounding the bias of contrastive divergence learning. *Neural Computation* **23** (2011) 664–673
6. Cappé, O., Guillin, A., Marin, J.M., Robert, C.P.: Population Monte Carlo. *Journal of Computational and Graphical Statistics* **13** (2004)

CAPTCHA Recognition with Active Deep Learning

Fabian Stark, Caner Hazırbaşı, Rudolph Triebel, and Daniel Cremers

Technical University of Munich, Germany

{fabian.stark,c.hazirbas,rudolph.triebel,cremers}@in.tum.de

Abstract. CAPTCHAs are automated tests to tell computers and humans apart. They are designed to be easily solvable by humans, but unsolvable by machines. With Convolutional Neural Networks these tests can also be solved automatically. However, the strength of CNNs relies on the training data that the classifier is learnt on and especially on the size of the training set. Hence, it is intractable to solve the problem with CNNs in case of insufficient training data. We propose an Active Deep Learning strategy that makes use of the ability to gain new training data for free without any human intervention which is possible in the special case of CAPTCHAs. We discuss how to choose the new samples to re-train the network and present results on an auto-generated CAPTCHA dataset. Our approach dramatically improves the performance of the network if we initially have only few labeled training data.

1 Introduction

A CAPTCHA [1] (**C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part) is an automated test to identify whether the user is a human or not. CAPTCHAs are often used on the internet to prevent automated programs from abusing online services. Nowadays, most service providers such as email or online shopping sites require users to solve CAPTCHAs, which most often consist of some distorted text that must be read and typed in correctly. For humans this is a comparably simple task, but computers still have difficulties here. Useful CAPTCHAs should be solvable by humans at least 80% of the times while programs using reasonable resources should succeed in less than 0.01% of the cases [4]. An example of a CAPTCHA is shown in Fig. 1.

Recently, researchers have started investigating automated methods to solve CAPTCHAs. Many of these existing solutions first perform character segmentation and then recognition. However, they can not solve newer, more challenging CAPTCHAs, where the letters are skewed so that they can not be separated by vertical lines. Thus, rectangular windows can not be used for segmentation, and more powerful classification methods are needed. One successful approach proposed recently by Goodfellow *et al.* [9] uses a deep Convolutional Neural Network (CNN), a framework that is also used in many other tasks such as object classification [5, 6], automatic speech recognition [8, 10] or natural language processing [3, 7]. However, a major requirement for training a deep CNN is a very large training data set (for example the ImageNet [15] data for image classification),



Fig. 1. Google’s reCAPTCHAs [2] consist of a distorted word and a word scanned from a text book. The user must type both words, thereby helping to digitize printed books.

and for CAPTCHA recognition, there is usually only a small annotated training data set available. Furthermore, the appearance of CAPTCHAs can, in contrast to objects in natural images, often change significantly from those in the training data, e.g. due to major changes in the distortion applied to the text.

To address these problems, we propose in this paper an approach that is based on Active Learning. The idea here is to start learning with a comparably small training set and to add new training samples in every subsequent learning round. The decision whether to add a sample to the training data is based on the uncertainty that the classifier associates with a given prediction. Under the assumption that this uncertainty estimation is well calibrated, the algorithm selects the most informative samples to learn from, resulting in less training samples required than in standard passive learning. As a further advantage, the algorithm can adapt to new input data that differs in appearance from the current training data. We note however that our problem is different to other Active Learning settings in that we do not need a human supervisor to acquire the ground truth labels for training. Instead, we use the return value that we obtain automatically when solving a CAPTCHA. Thus, if the classifier is able to solve a CAPTCHA correctly we can use that information for re-training, because then the ground truth label is known. Of course, if the CAPTCHA is not solved we don’t have that information, but we will show how learning can be done from the correctly predicted samples only. In summary, we present three novel contributions: First, we show how to compute uncertainty from a deep CNN and how this relates to correct classification. Second, we perform Active Learning with a deep CNN. And third, we show that already the correct, but uncertain classified samples are enough for efficient learning, with the effect that we need only little training data, and this is obtained without any human intervention.

2 Related Work

Conventional methods aim at detecting the text within natural images in two disjoint steps [11]: localizing the regions of words or single characters within the image, segmenting [17] and then recognizing them [19]. In addition, a dictionary can be used to dismiss unlikely words. For example, Mori and Malik [18] proposed a method to solve CAPTCHAs using a dictionary with all 411 words that the considered CAPTCHAs contain. Chellapilla and Simard [4] also solve CAPTCHAs

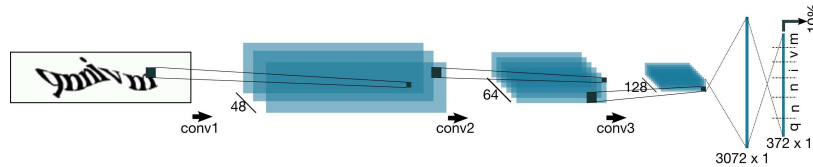


Fig. 2. Convolutional Neural Network for CAPTCHA Recognition. Our CNN is composed of three convolution, three pooling and two fully-connected layers. The last layer outputs the probability distributions for all digits for which we can compute the prediction and uncertainty of the prediction.

by segmenting single characters and recognizing them, but without a dictionary. However, in modern CAPTCHAs, single characters can not be segmented easily with rectangular windows, as the characters can overlap each other (see Fig. 1). These CAPTCHAs are more similar to hand-written text, and LeCun *et al.* [16] proposed to use Convolutional Neural Networks (CNN) for recognition of hand-written digits. These CNNs are designed to construct the hierarchy of the objects layer by layer and perform classification. In 2014, Goodfellow *et al.* [9] proposed to combine localization, segmentation and recognition of multi-character text using deep CNNs. Training is done on millions of images using a cluster of several computers. Jaderberg *et al.* [12] proposed a CNN for text recognition on natural scene images. However, for training they artificially create a very large set of text images. In contrast, we use a much smaller training set. By exploiting Active Learning, we fine-tune the network during runtime, and our network is fed with correctly classified but highly uncertain test samples.

3 A Deep CNN for CAPTCHA Recognition

We propose a deep CNN to solve the whole sequence of a CAPTCHA. Our purpose is to recognize the full sequence without pre-segmentation. We use the network structure shown in Fig. 2. We focus on CAPTCHAs with 6 digits. Each digit is represented by 62 neurons in the output layer. We define a bijection $\Theta(x)$ that maps a character $x \in \{‘0’, \dots, ‘9’, ‘A’, \dots, ‘Z’, ‘a’, \dots, ‘z’\}$ to an integer $l \in \{0, \dots, 61\}$:

$$\Theta(x) = \begin{cases} 0 \dots 9, & \text{if } x = ‘0’ \dots ‘9’ \\ 10 \dots 35, & \text{if } x = ‘A’ \dots ‘Z’ \\ 36 \dots 61, & \text{if } x = ‘a’ \dots ‘z’ \end{cases} \quad (1)$$

We assign the first 62 output neurons to the first digit of the sequence, the second 62 neurons to the second digit and so on. Thus, for a digit x_i the neuron index n is computed as $n = i \cdot 62 + \Theta(x_i)$, where $i \in \{0, \dots, 5\}$ is the index of the digit, i.e. the output layer has $6 \cdot 62 = 372$ neurons. To predict a digit, we consider the corresponding 62 neurons and normalize their sum to 1. Fig. 4 shows an example of a network output. Here, the predicted character index for the first digit is $c_0 = 52$ and the predicted label is $x = \Theta^{-1}(c_0) = ‘q’$.

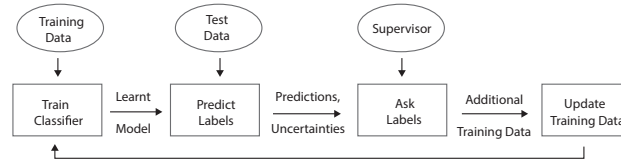


Fig. 3. Flow chart of Active Learning. We start with training on a small data set. Then, the classifier is applied to some new data, resulting in a label prediction and an associated uncertainty. From this uncertainty, the classifier decides whether to ask for a ground truth label or not. In our case, this query is done by solving the given CAPTCHA with the prediction and using it if it was correct. Then, the training data is increased and learning is performed again. In our method, we use a deep CNN, which can be efficiently re-trained using the newly added training samples.

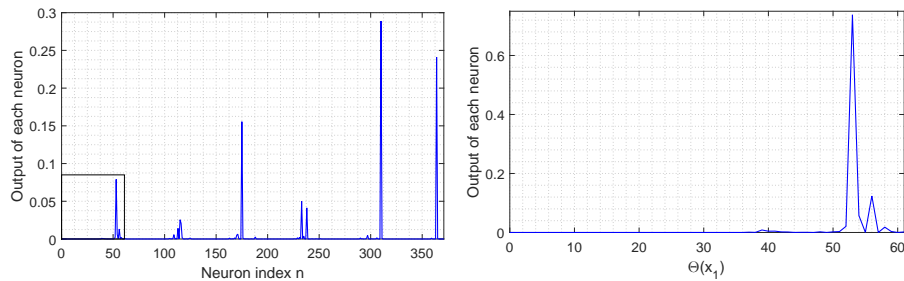


Fig. 4. Example output of the network for the CAPTCHA “qnnivm” in Fig. 2. **Left:** There are 62 outputs for each digit. The black box shows the output for the first digit. **Right:** Probability distribution for the first digit. The sum is normalized to 1.

4 Active Learning to Reduce the Required Training Data

For a good classification accuracy, CNNs usually require a very large training set. However, collecting millions of hand-labeled CAPTCHAs is infeasible. Therefore, we propose to use Active Learning (see Fig. 3). The main idea of this is to add new training data only if necessary, i.e. if the sample is informative enough for re-learning. This is decided based on the uncertainty of the prediction, which we compute using the best-versus-second-best strategy [14], as described next.

4.1 Obtaining the uncertainty

As mentioned above, we estimate the predictive distribution of each digit by normalizing the sum of the corresponding network outputs to 1. From this we compute the overall uncertainty η using “best-vs-second-best” as

$$\eta = \frac{1}{d} \cdot \sum_{i=1}^d \frac{\arg \max \{ \mathcal{P}(x_i) \setminus \arg \max \mathcal{P}(x_i) \}}{\arg \max \mathcal{P}(x_i)}, \quad (2)$$

where $\mathcal{P}(x_i)$ is the set of all network outputs for digit d_i . Thus we divide the second best by the best prediction for every digit.

4.2 Querying Ground Truth Information

Our CAPTCHA recognition problem is unique in the sense that we can perform learning without human intervention. We achieve this by only using those data samples for re-training, for which the classifier already provided a correct label. For these, the CAPTCHA can be solved and we know what the correct text is. However, simply using all these correctly classified samples for re-training would be very inefficient. In fact, training would be done more and more often, because the classifier will be better over time and therefore classify more samples correctly. Thus, with every new correctly classified sample a retraining would be necessary. To avoid this, we use the uncertainty values presented above: We sort the correctly classified test samples in each learning round by prediction uncertainty and use only the most uncertain ones for re-training. This results in a lower number of required training samples, but as we will show in the experiments, the most uncertain samples are also the most informative for learning.

5 Experimental Evaluation

We present the results of our approach on auto-generated CAPTCHAs. All experiments have been executed using the Caffe [13] deep learning framework on an NVIDIA GeForce[®] GTX[™] 750 Ti GPU.

5.1 Dataset Generation

As there is no hand-labeled CAPTCHA dataset, we use our own scripts to generate CAPTCHAs. During the auto-generation, we ensure that there is no duplication in the dataset.

We use the **Cool PHP CAPTCHA** framework to generate CAPTCHAs. They are composed of distorted text with a fixed length of 6 similar to Google's reCAPTCHA. They have a size of 180×50 . We have modified the framework to generate black and white images. Furthermore we have disabled shadows and the line through the text. We also do not use dictionary words, but random characters. Therefore we have removed the rule that every second character has to be a vowel. We fix the font to "AntykwaBold". Fig. 5 shows some examples of our auto-generated CAPTCHAs.

5.2 Network Design

We use the network illustrated in Fig. 2. The convolutional layers have a size of 48, 64 and 128. They all have a kernel size of 5×5 and a padding size of 2. The pooling layers have a window size of 2×2 . The first and third pooling layers and also the first convolutional layer have a stride of 2. Then the network has




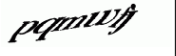






				
qnnivm	oigjppj	ijcvyl	pqmwfj	eilrqi
				
cvodvt	njbzpz	pzcqee	hepfpf	ijlmqw

Fig. 5. Example CAPTCHAs used in the experiments.

one fully connected layer with a size of 3072 and a second fully connected layer (classifier) that has an output size of 372. We also add rectified linear units and Dropout after every convolutional and the first fully connected layer. The batch size for every iteration is 64.

5.3 Qualitative Evaluation

We train the network with the SGD algorithm. However, in contrast to other methods we train the network for all digits independently. The learning rate changes by the rule $\alpha = \alpha_0 \cdot (1 + \gamma \cdot t)^{-\beta}$ where the base learning rate $\alpha_0 = 10^{-2}$, $\beta = 0.75$, $\gamma = 10^{-4}$ and t is the current number of iteration. We set *momentum* $\mu = 0.9$ and *regularization parameter* λ is $5 \cdot 10^{-4}$.

As the most expensive part is to get the training samples, our approach aims at decreasing the required size of the initial training set. So we first of all train our network with a very small initial training set of 10^4 images for $5 \cdot 10^4$ iterations. We only achieve an accuracy of 9.6% which even decreases with more iterations. Because of that, we want to make use of Active Learning.

First of all, we again train our network with 10^4 training images for $5 \cdot 10^4$ iterations. Afterwards, we classify $5 \cdot 10^4$ test images. Then, we pick new training samples from the correctly classified ones. We can take all of them, or we only pick $5 \cdot 10^3$ samples based on their uncertainty: Either with the highest uncertainty, the lowest uncertainty, or randomly. Uncertainty is computed as described in Section 4.1. Once the new selected samples are added to the training set, we re-train the network for $5 \cdot 10^4$ iterations. Subsequently we follow the same procedure. We apply this algorithm for in total 20 Active Learning rounds (*epochs*). The accuracy is computed after every $5 \cdot 10^3$ iterations on a fixed validation set. We get the best performance with the correct but uncertain predictions (see top plot in Fig. 6). All results are the average out of two runs.

However increasing the number of samples in the training set requires more storage. Moreover, one should increase the number of iterations to benefit more from the cumulated set which will cause longer training time. For all these reasons, we suggest to use only the selected samples at each iteration to re-train the network. Therefore we again train with 10^4 initial training images for $5 \cdot 10^4$ iterations. Then we classify 10^5 test images and replace the training set with 10^4 of the correct classified ones and train for $2.5 \cdot 10^5$ iterations again. Subsequently we follow the same procedure and decrease the number of iterations after each epoch according to the following rule: $2.5 \cdot 10^4$ iterations until epoch 6, $2 \cdot 10^4$ until

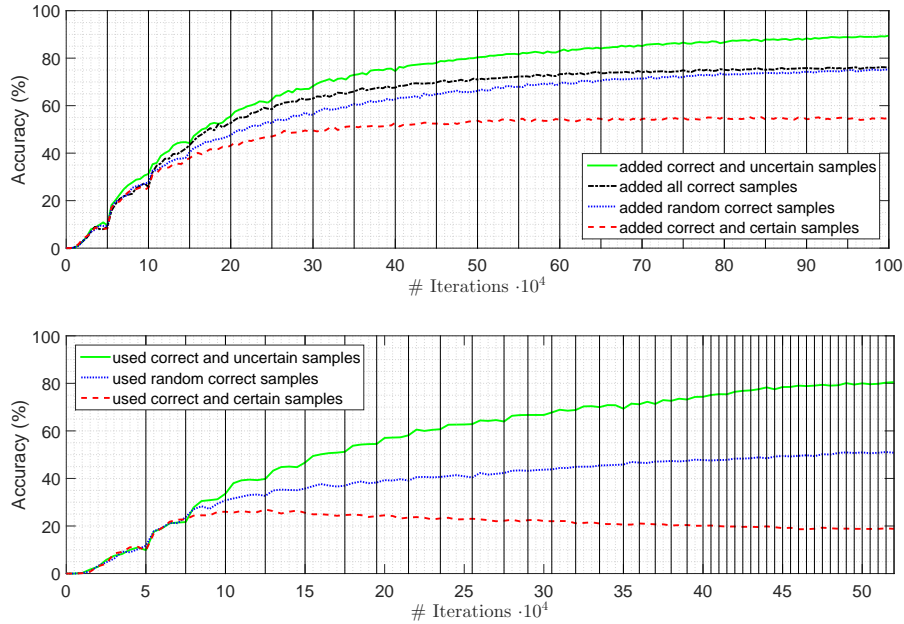


Fig. 6. Learning curves for Active Deep Learning. **Top:** The training set is increased with the selected samples after each iteration. When using all the correct ones (black curve), we stop adding new images to the training set after $50 \cdot 10^4$ iterations, because the size of the training set already exceeds $3 \cdot 10^6$. **Bottom:** Network is retrained only on the new samples. Vertical black lines denote the end of every Active Learning epoch.

epoch 11, $1.5 \cdot 10^4$ until epoch 16, $1 \cdot 10^4$ until epoch 21 and $5 \cdot 10^3$ until epoch 40. We again get the best performance with the correct but uncertain predictions (see bottom plot in Fig. 6). This is reasonable as the network in fact classifies the images correctly, but still is very uncertain about the prediction. Hence it can learn from the fact that it was indeed right with its classification. One can argue that learning with misclassified samples should yield better results. This is indeed the case, however not possible in practice.

6 Conclusion

We propose a CAPTCHA solving technique that uses initially a very small set of images to train a deep CNN and then improves the classifier by exploiting the test samples. New training samples are chosen from the test set based on their uncertainty. Our results show that the performance of the network can be significantly improved with the correctly classified but uncertain test samples.

Acknowledgments The work in this paper was partly funded by the EU project SPENCER (ICT-2011-600877).

References

1. von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: Captcha: Using hard ai problems for security. In: EUROCRYPT (2003)
2. von Ahn, L., Maurer, B., McMillen, C., Abraham, D., Blum, M.: recaptcha: Human-based character recognition via web security measures. *Science* (2008)
3. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. *The Journal of Machine Learning Research* 3, 1137–1155 (2003)
4. Chellapilla, K., Simard, P.Y.: Using machine learning to break visual human interaction proofs (hips). In: NIPS (2004)
5. Claudiu Ciresan, D., Meier, U., Gambardella, L.M., Schmidhuber, J.: Deep big simple neural nets excel on handwritten digit recognition. arXiv preprint arXiv:1003.0358 (2010)
6. Coates, A., Ng, A.Y., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: *Int. Conf. on Artificial Intell. and Statistics*. pp. 215–223 (2011)
7. Collobert, R., Weston, J.: A unified architecture for natural language processing: Deep neural networks with multitask learning. In: *Proceedings of the 25th international conference on Machine learning*. pp. 160–167. ACM (2008)
8. Dahl, G.E., Yu, D., Deng, L., Acero, A.: Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on* 20(1), 30–42 (2012)
9. Goodfellow, I.J., Bulatov, Y., Ibarz, J., Arnoud, S., Shet, V.: Multi-digit number recognition from street view imagery using deep convolutional neural networks. *ICLR* (2014)
10. Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., et al.: Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE* 29(6), 82–97 (2012)
11. Jaderberg, M., Vedaldi, A., Zisserman, A.: Deep features for text spotting. In: *CVPR* (2014)
12. Jaderberg, M., Simonyan, K., Vedaldi, A., Zisserman, A.: Reading text in the wild with convolutional neural networks. *IJCV* (2015)
13. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014)
14. Joshi, A., Porikli, F., Papanikolopoulos, N.: Multi-class active learning for image classification (2009)
15. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)
16. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. *Proceedings of the IEEE* (1998)
17. Lu, Y.: Machine printed character segmentation; an overview. *Pattern Recognition* 28(1), 67–80 (1995)
18. Mori, G., Malik, J.: Recognizing objects in adversarial clutter: breaking a visual captcha (2003)
19. Simard, P.Y., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks applied to visual document analysis. In: *2013 12th International Conference on Document Analysis and Recognition*. vol. 2, pp. 958–958. IEEE Computer Society (2003)

Intrinsic Plasticity: A Simple Mechanism to Stabilize Hebbian Learning in Multilayer Neural Networks

Michael Teichmann, Fred H. Hamker

Technische Universität Chemnitz, Department of Computer Science,
Straße der Nationen 62, 09107 Chemnitz, Germany

{[michael.teichmann](mailto:michael.teichmann@informatik.tu-chemnitz.de), [fred.hamker](mailto:fred.hamker@informatik.tu-chemnitz.de)}@informatik.tu-chemnitz.de

<https://www.tu-chemnitz.de/informatik/KI/>

Abstract. Hebbian learning uses only information local to the neurons. Hence, there is no control of the encoding by a population of neurons. This can lead to an imbalanced representation particularly in deeper networks. Intrinsic plasticity, the regulation of the neurons excitability, can overcome this issue by ensuring that each neuron equally participates in the encoding. We present a simple form of this mechanism, regulating the mean and the variance of a neuron based on local information, within a fully plastic multilayer model of the early visual system. Where all excitatory as well as inhibitory network connections are learnable, using Hebbian and anti-Hebbian principles. We show that the presented intrinsic plasticity mechanism effectively regulates the mean and the variance of the neuron's activities, improving the encoding of deeper network layer.

Keywords: intrinsic plasticity, deep neural networks, Hebbian learning

1 Introduction

One of the most important criteria of biologically plausible learning principles is their locality, i.e. a neuron has only access its own state and incoming signals. Common formulations of Hebbian learning use only information local to synapse or neuron. Indeed, such formulations can not ensure that a population of so defined neurons will learn a codebook representing the inputs manifold. By extending such populations by plausible mechanisms of inter neuron interaction, as inhibition, it can be shown that an adequate representation of the input can be learned [2]. A common plausible principle to learn such connections is anti-Hebbian learning [2, 13, 8]. However, this principle aims only to reduce correlations between neurons based on their coactivity. It has no objective ensuring an adequate input encoding by a neuron population. Further, Hebbian learning depends on the association between pre- and postsynaptic activity. If a presynaptic neuron shows more activity, it's connection will be more strengthen. Hence, it is very likely for more complex input data or in higher layers of abstractions

in deep networks, that some neurons will show higher activity for the pattern they encode. Which in turn will enforce the imbalance in subsequent layers.

The human nervous system has various mechanisms to stabilize its functioning [12, 11]. Beside homo- and hetero-synaptic regulations of the weight development, as synaptic scaling, it has been found that neurons preserve their average firing rate over time [14]. If a neuron is highly stimulated, respectively weakly, over a long period of time it can be observed that its sensitivity to this stimulation decreases, respectively increases, and it returns to its previous activity regime. This means, its intrinsic excitability is adapted, called intrinsic plasticity [14, 11].

For first computational implementations of intrinsic plasticity it has been speculated that neurons try to approach an exponential firing regime [10], being efficient from the perspective of information theory. To transfer any input distribution into an exponential output distribution, a nonlinear transfer function, as a sigmoidal, is needed [10]. Indeed, it is still unclear whether the objective of cortical neurons is an exponential regime as well as how they achieve it. Beside sigmoidal activation functions, rectified linear activation functions have been very common in the past and recently in deep neural networks. There is also biological evidence that this function type is an adequate description for cortical neurons [7]. However, this function is mainly linear, except its rectification. Hence, mathematically it can not transfer any input distribution into an exponential one.

We believe that the aspect of the exponential regime is not the objective of cortical neurons and is a byproduct of the neural circuit developing sparse representations. Instead, the intrinsic plasticity mechanism aims to stabilize the operating point of a neuron so that the brain is not wasting resources for non responding cells or hyperactive ones. Thus, we aim to control the two most important moments of neural activity, the mean and the variance, by adapting the slope and the threshold of the neurons rectified linear activation function. We implemented this form of intrinsic plasticity into a model of the primary visual cortex (V1) to demonstrate its effectiveness for stabilizing the neural response properties.

2 Model

Based on our previous work [13, 8], we developed a fully plastic network model of V1. The network consists of two layers with two populations of excitatory and inhibitory neurons, respectively. The connectivity between both layers relies on neuroscientific foundations. Following we will introduce the basic structure and the main mechanisms of the network.

Architecture After preprocessing the gray scaled input images, using a whitening procedure [5, 13], we set these images on the input layer LGN (lateral geniculate nucleus). LGN provides input to all neurons in V1-layer 4, which in turn is projecting to layer 2/3. Beyond the standard feedforward view on V1, we implemented the connectivity based on neuroscientific data [6, 1, 9], resulting in a

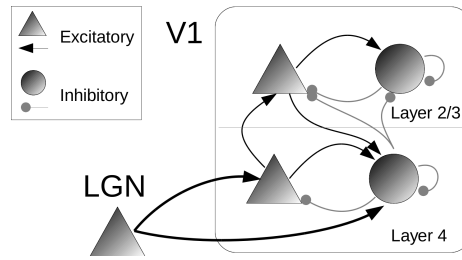


Fig. 1. Model architecture. Two layer V1 model, using dedicated excitatory and inhibitory neurons. The connectivity base on neuroscientific foundations.

richer connectivity structure. Figure 1 provides an illustration of the connections between all populations of neurons in the network. The neurons in the network are retinotop organized (for geometry see Table 1). Each neuron has a limited receptive field and is only connected to its neighbors. For instance, the connections of the first V1-layer 4 neuron are defined so that it receives input from the first 12 by 12 patch of the LGN population. Dependent on the position in the layer grid the next neuron receives a shifted patch (1px shifts). Further, neurons having overlapping connections are all connected with the same inhibitory neuron. All neurons having the same x and y position also have the same connectivity. If neurons are connected to a certain x-y position then they have a connection to all neurons at this position, i.e. to all in the z dimension. Due to this organization the model is scalable to any input size. Indeed to reduce the computational costs, we chose a 24 by 24 size for the input.

Neuron activation function The model neurons are described via differential equations calculating the firing rate. The membrane potential m_j of a neuron j is calculated via Eqn. 1,

$$\tau_m \frac{\partial m_j}{\partial t} = a_j \cdot \left(\sum_i w_{ij} r_i - \sum_{k, k \neq j} c_{kj} r_k - \theta_j \right) - m_j \quad (1)$$

where r_i denotes the firing rate of an afferent neuron i and w_{ij} is the excitatory weight between these two neurons and c_{kj} is the inhibitory weight from

Table 1. Layer geometry and amount of neurons. The x and y dimension denotes the position in the grid and the z dimension the amount of cells at this position.

Layer	Type	Geometry	Neurons
LGN	Input	24x24x2	1152
V1-L4	Excitatory	13x13x4	676
	Inhibitory	13x13x1	169
V1-L2/3	Excitatory	7x7x12	588
	Inhibitory	7x7x3	147

another neuron k . The influence of these currents on the neuron can be modulated by a threshold θ_j and a slope a_j . The time constant τ_m is set to $10ms$. The neurons output is defined as rectified linear function with a saturation term for high values (Eqn. 2).

$$r_j = \begin{cases} 0.5 + \frac{1}{1+e^{-3.5(m_j-1)}} & \text{if } m_j > 1 \\ (m_j)^+ & \text{else} \end{cases} \quad (2)$$

Intrinsic plasticity The fraction neurons participating in the encoding is controlled by their first activity moments [10]. This is, the mean and variance of activity. We adapt the parameters θ_j and a_j for each individual neuron, so that a target activity θ^{target} and squared activity a^{target} are approached. This is done by increasing the threshold when the neuron is more active than the target value (Eqn. 3) and decreasing the slope when the neurons squared activity is above the target value (Eqn. 4). Hence, over time (τ_θ and τ_a are set to $10000ms$) all neurons are pushed into the same regime.

$$\tau_\theta \frac{\partial \theta_j}{\partial t} = (r_j - \theta^{target}) - \delta(\theta_j) \quad (3)$$

$$\tau_a \frac{\partial a_j}{\partial t} = (a^{target} - r_j^2) - \delta(a_j - 1) \quad (4)$$

We add a small constant ($\epsilon = \frac{1}{100}$) drift $\delta(x)$ in the direction of the initial values $\theta = 0$ and $a = 1$ (Eqn. 5), giving the neurons a small bias to prefer a minimal modified activation function. Further, this prevents neurons in very deep layers, learning long periods from very noisy inputs, from developing extreme values.

$$\delta(x) = \epsilon \cdot \text{sgn}(x) \quad (5)$$

Neuron calcium level Unlike the standard Hebbian learning, we rely on the postsynaptic calcium concentration instead of directly using the neurons firing rate (for details see [8]). This calcium concentration Ca_j follows the postsynaptic firing rate with a layer specific time constant $\tau_{Ca,layer}$ (Eqn. 6). This allows us to learn from the activity trace, facilitating the learning of invariance properties by using a slow time constant [8]. We set the time constant for layer 4 neurons to $10ms$ and layer 2/3 neurons to $500ms$.

$$\tau_{Ca,layer} \frac{\partial Ca_j}{\partial t} = r_j - Ca_j \quad (6)$$

Calcium dependent Hebbian learning For learning the excitatory connections, we use Hebbian learning with a covariance term on the presynaptic site, where \bar{r}^{pre} denotes the mean activity of the presynaptic layer (Eqn. 7). Multiplied with the postsynaptic calcium concentration and discounted by a normalization term [3], constraining the weight vector length. The parameter α_j is adapted based on the maximum activity of a neuron (see [8]).

$$\tau_{Learn,j} \frac{\partial w_{ij}}{\partial t} = (r_i - \bar{r}^{pre}) \cdot Ca_j - \alpha_j (Ca_j)^2 w_{ij} \quad (7)$$

Anti-Hebbian learning To decorrelate the neurons, so that each becomes selective to another pattern, we use anti-Hebbian learning on inhibitory connections (cf. [13]). An inhibitory weight c_{kj} is strengthened when both neurons, the pre- and the postsynaptic, are coactive and is decreased when only the postsynaptic neuron shows activity (Eqn. 8). To prevent the total suppression of a neuron the weight is also decreased when the postsynaptic activity drops below $\theta_c = 0.05$.

$$\tau_c \frac{\partial c_{kj}}{\partial t} = r_k \cdot (r_j - \theta_c)^+ - \alpha_c \cdot r_j \cdot c_{kj} \quad (8)$$

3 Methods

Network training To learn receptive fields, being later comparable to findings in the primate brain, we trained the model on natural scenes [4, 13, 8]. Therefore, we present a sequence of 500000 image patches for 100ms to the network. Each 10 presentations a new image source and new patch coordinates are randomly selected. After this selection the patch position underlies a random walk for 10 presentations, simulating microsaccadic eye movements (for details see [8]).

Evaluation We want to investigate the effectiveness of the proposed intrinsic plasticity mechanism to regulate the mean and variance of the neuron activity. To obtain the response statistic for each network neuron after learning, we turn off all plasticity mechanisms and present 100000 randomly selected image patches to the network. At the end of a presentation period (100ms), we record the responses of the excitatory neurons and calculate the mean and variance for each from its 100000 responses. For comparison, we further repeat the same analysis for three model variations, where we: 1) turned off the intrinsic plasticity, 2) used only the regulation of the mean, and 3) use only the regulation of the variance. Further, we investigate the parameter development of θ , respectively a , and show their distribution for the model with full intrinsic plasticity.

4 Results

The model with intrinsic plasticity shows normally distributed means (Fig. 2a,b) and variances (Fig. 2e,f) for all layers. Whereas, the neurons of the model without intrinsic plasticity have only in layer 4 normally distributed means and variances, with a few neurons being much more active than the average (Fig. 2c,g). The neurons in layer 2/3 have an exponential like distribution of their means and variances (Fig. 2d,h). Here, a huge fraction of neurons shows nearly no activity or variance and some have very high averages, indicating a strongly imbalanced population encoding. The control experiments 2 and 3, with θ respectively a regulation only, show that regulating each variable is effective and the distribution of the means, respectively variances, is comparable with the “full” intrinsic plasticity model (Fig. 3). The other unregulated variable behaves as in the model without intrinsic plasticity (Fig. 3b,c). However, the results are slightly better

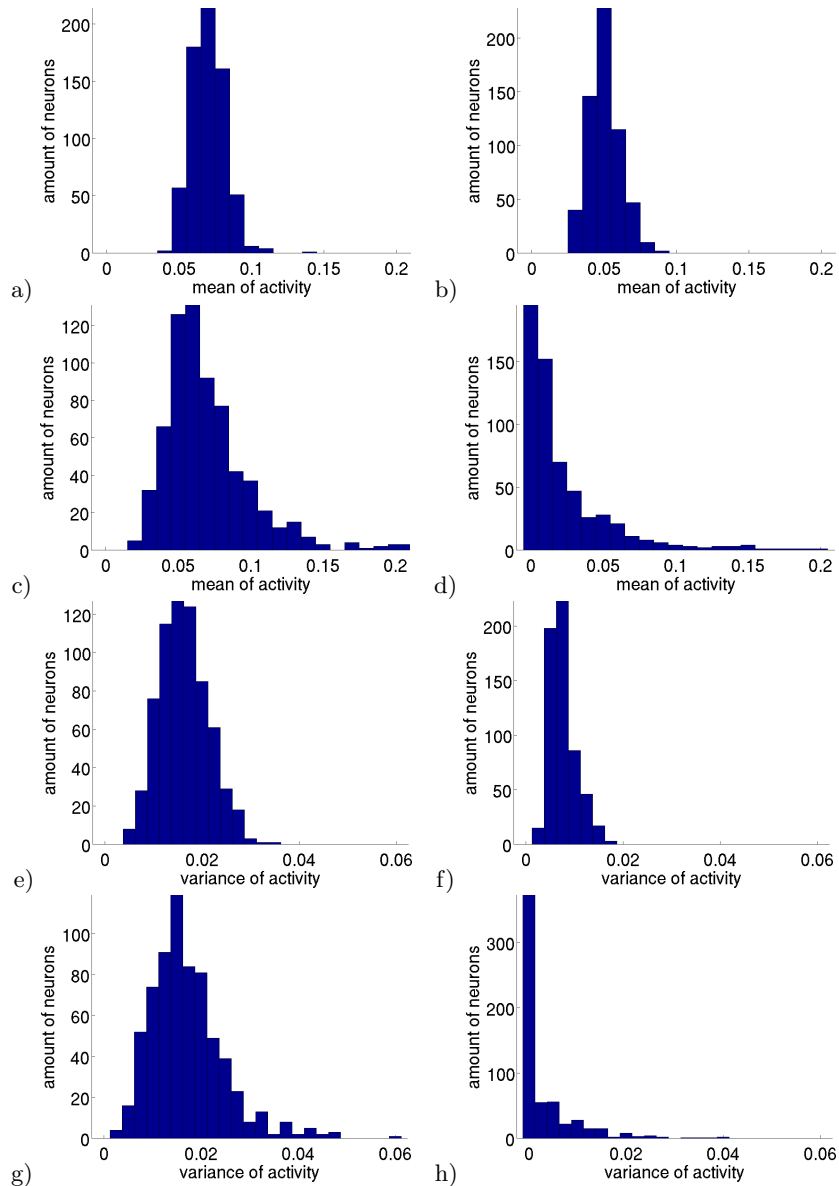


Fig. 2. Histograms of the mean and variance of the neurons activity. The left column (a, c, e, g) shows results from layer 4 neurons and the right from layer 2/3 neurons. The mean of activity (a, b) and the variance of activity (e, f) are obtained with full intrinsic plasticity. Whereas the subsequent row (c, d; g, h) shows the results obtained without intrinsic plasticity.

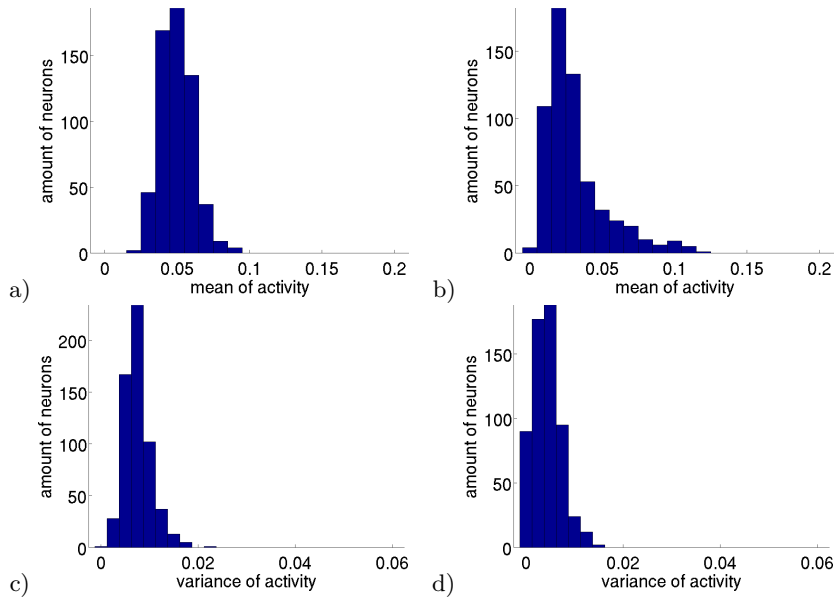


Fig. 3. Histograms of the mean and variance of the neurons activity, regulating a single variable. The left column (a, c) shows the results for layer 2/3 neurons regulating θ and right shows the results regulating a .

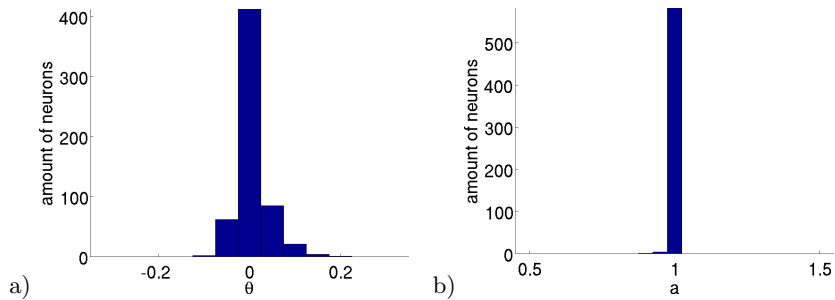


Fig. 4. Distribution of a) the threshold θ and b) the slope a for layer 4 neurons in the model with full intrinsic plasticity.

as fewer inactive neurons are obtained. Which in turn indicates that regulating the mean, respectively variance, is enough to preserve the neurons activity level so that they still participate in the encoding. Furthermore, no neurons with very high values are found for the unregulated variable. Hence, regulating a single activity moment can effect other moments.

We investigated the stability of the intrinsic plasticity mechanism by visualizing the two parameters θ and a (Fig. 4). The threshold of the neurons is

normally distributed around its origin in an adequate range with no extreme values. The variable a shows a narrow normally distribution around its origin too.

5 Conclusion

We argued that deep networks, based on unsupervised Hebbian learning, can suffer from an imbalanced encoding of their neurons, amplifying in deeper layers. We demonstrated this effect in a multilayer network of the early visual cortex (V1). To overcome this issue we proposed a simple form of intrinsic plasticity, regulating the first moments of activity, the mean and variance of a neuron. We showed that this regulation leads to populations of neurons having similar means and variances. Further, we showed that deeper layer benefit from this regulation and developed a balanced representation of the input, where each neuron is participating in the encoding. Besides, we demonstrated that the regulation mechanisms are effective in regulating their target value. However, the regulation of one moment also influence the results of the other moment. An important criteria for the regulation mechanism is its stability. The proposed intrinsic plasticity mechanism is demonstrated to develop values in a suitable range.

Acknowledgments. The work has been supported by the German Research Foundation (DFG GRK1780/1) and by the US-German collaboration on computational neuroscience (BMBF 01GQ1409).

References

1. R. J. Douglas and K. A. C. Martin. Neuronal circuits of the neocortex. *Annu. Rev. Neurosci.*, 27:419–51, Jan. 2004.
2. P. Földiák. Forming sparse representations by local anti-Hebbian learning. *Biol. Cybern.*, 237(5349):55–56, May 1990.
3. E. Oja. A simplified neuron model as a principal component analyzer. *J. Math. Biol.*, 15(3):267–273, 1982.
4. B. Olshausen and D. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–9, 1996.
5. B. a. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: a strategy employed by V1? *Vision Res.*, 37(23):3311–25, Dec. 1997.
6. T. C. Potjans and M. Diesmann. The Cell-Type Specific Cortical Microcircuit: Relating Structure and Activity in a Full-Scale Spiking Network Model. *Cereb. Cortex*, Dec. 2012.
7. D. L. Ringach and B. J. Malone. The operating point of the cortex: neurons as large deviation detectors. *J. Neurosci.*, 27(29):7673–83, July 2007.
8. M. Teichmann, J. Wiltchut, and F. H. Hamker. Learning invariance from natural images inspired by observations in the primary visual cortex. *Neural Comput.*, 24(5):1271–96, May 2012.

9. A. M. Thomson and A. P. Bannister. Interlaminar connections in the neocortex. *Cereb. Cortex*, 13(1):5–14, 2003.
10. J. Triesch. Synergies between Intrinsic and Synaptic Plasticity in Individual Model Neurons. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Adv. Neural Inf. Process. Syst. 17*, pages 1417–1424. MIT Press, 2005.
11. G. Turrigiano. Too many cooks? Intrinsic and synaptic homeostatic mechanisms in cortical circuit refinement. *Annu. Rev. Neurosci.*, 34:89–103, Jan. 2011.
12. G. G. Turrigiano and S. B. Nelson. Homeostatic plasticity in the developing nervous system. *Nat. Rev. Neurosci.*, 5(2):97–107, Mar. 2004.
13. J. Wiltschut and F. H. Hamker. Efficient coding correlates with spatial frequency tuning in a model of V1 receptive field organization. *Vis. Neurosci.*, 26(1):21–34, 2009.
14. W. Zhang and D. J. Linden. The other side of the engram: experience-driven changes in neuronal intrinsic excitability. *Nat. Rev. Neurosci.*, 4(11):885–900, Nov. 2003.

Identifying bank stress by deep learning of news

Samuel Rönqvist¹ and Peter Sarlin²

¹ Turku Centre for Computer Science / Åbo Akademi University

² Hanken School of Economics / RiskLab Finland, Arcada Univ. of Applied Sciences
sronnqvi@abo.fi, peter@risklab.fi

In the aftermath of the global financial crisis, machine-learning-based analysis of financial stability gained considerable traction. A particular interest has been the modeling of banks, to identify or forecast risks. Yet, problems pertaining to data availability and quality persist. We explore the use of financial news text as a novel data source in the study of bank distress events. This paper discusses how deep learning may be used to take advantage of this new data source, by modeling discussion related to banks in order to predict coinciding distress events. We propose to employ unsupervised pre-training based on distributional semantics to learn semantic vector representations of news articles as practical low-dimensional, fixed-length features. Compared to conventional data sources and methods, a particular advantage of using deep learning on text data for the task is the qualitative information and explanation of events that it may provide.

To predict bank distress, most earlier works focus on accounting data and the identification of early build-up of risk and imbalances (e.g., [1]), but they suffer from data issues such as restricted access and infrequent and lagging reporting. By contrast, market data constitute a more accessible and timely source of information, as a proxy for imbalances and stress (e.g., [2]). In this context, text data like financial news represent an alternative source of information that is readily accessible, abundant, timely and descriptive. The qualitative content of text is not limited to its use as predictive input, but can provide descriptions as a way of making models more interpretable, while market data does not in itself offer descriptive information regarding events.

Although text data has not previously been used to predict bank stress, some recent work use the source for related tasks. Dictionary-based sentiment analysis has been applied to study the tone in news as a sign of growing market imbalances ([8, 5]). Likewise, data-driven approaches may provide more flexible means of analyzing risk (e.g., [9]), although its results are not necessarily as easy to interpret. We advocate a fully data-driven setup that is flexible, in the sense that the distress event data alone define the type of risk to be modeled. Yet, the semantic modeling involved should provide a good basis for interpretation of results. While the dictionaries reflect elementary types of sentiment, our approach can be thought of as risk sentiment analysis targeted by event data selection. Hence, this method may be applicable beyond the analysis of bank distress events.

A common principle in deep learning is to learn abstractions of input data, as a replacement for classical feature engineering [6]. Unsupervised pre-training of abstraction layers in the deep neural network offers flexibility and can support supervised learning against signals (e.g., distress events). As [7] point out,

this flexibility is particularly useful in text analysis. Manually constructed features used to introduce structure and generalizations into text tend to be over-specified, incomplete and laborious to develop for specific tasks and languages.

Abstractions in our case are semantic representations of text, based on distributional semantics. Modeling of word-level semantics is performed based on word contexts, and yields distributed representations as continuous word vectors, providing a semantic vector space embedding that makes the symbolic text input comparable. Mikolov et al. [4] have proposed a neural method able to learn accurate word vectors that scales to massive data sets (billions of words). This was extended by [3] to represent compositional semantics of sequences of words of arbitrary length, which is claimed to provide state-of-the-art performance on sentiment analysis of movie reviews. Similarly, their distributed memory method (a 3-layer feed-forward network) may be applied on news articles to learn document vectors as features in event prediction (by another 3-layer feed-forward network). Thus, the semantic vector learning and predictive modeling constitute two separate steps in training of the combined, deep neural network.

The semantic modeling reduces the dimensionality from the size of the vocabulary (millions of words) to the vector length (e.g., 400), while also providing a compositional representation of a sequence of words. In tentative experiments, the features have shown positive results in prediction, when pre-trained on 260k documents and then supervised against only 243 events. The approach holds promise as a means of harnessing text data for identification – and description – of bank distress, with little effort required for task-specific adaptations.

References

1. F. Betz, S. Opricǎ, T. A. Peltonen, and P. Sarlin. Predicting distress in european banks. *Journal of Banking & Finance*, 45:225–241, 2014.
2. R. Gropp, J. Vesala, and G. Vulpes. Equity and bond market signals as leading indicators of bank fragility. *Journal of Money, Credit & Banking*, 38:399–428, 2006.
3. Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196, 2014.
4. T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at International Conference on Learning Representations*, 2013.
5. R. Nyman, D. Gregory, K. Kapadia, P. Ormerod, D. Tuckett, and R. Smith. News and narratives in financial systems: exploiting big data for systemic risk assessment. BoE, mimeo, 2015.
6. J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.
7. R. Socher and C. Manning. Deep learning for natural language processing (without magic). Keynote at NAACL2013: Human Language Technologies.
8. C. K. Soo. Quantifying animal spirits: news media and sentiment in the housing market. *Ross School of Business Paper No. 1200*, 2013.
9. W. Y. Wang and Z. Hua. A semiparametric gaussian copula regression model for predicting financial risks from earnings calls. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2014.

Visualisation of heterogeneous data with simultaneous feature saliency using Generalised Generative Topographic Mapping

Shahzad Mumtaz, Michel F. Randrianandrasana, Gurjinder Bassi and Ian T. Nabney

System Analytics Research Institute (SARI),
Aston University, Birmingham, B4 7ET, United Kingdom.
Email: mumtazs;randrimf;t-bassig;i.t.nabney@aston.ac.uk

Abstract. Most machine-learning algorithms are designed for datasets with features of a single type whereas very little attention has been given to datasets with mixed-type features. We recently proposed a model to handle mixed types with a probabilistic latent variable formalism. This proposed model describes the data by type-specific distributions that are conditionally independent given the latent space and is called generalised generative topographic mapping (GGTM). It has often been observed that visualisations of high-dimensional datasets can be poor in the presence of noisy features. In this paper we therefore propose to extend the GGTM to estimate feature saliency values (GGTMFS) as an integrated part of the parameter learning process with an *expectation-maximisation* (EM) algorithm. The efficacy of the proposed GGTMFS model is demonstrated both for synthetic and real datasets.

1 Introduction

Type-specific data analysis has been well studied in the machine learning community [6]. In the recent couple of decades, the need to analyse mixed-type data is gaining a lot of attention from machine learning experts because of the fact that real world processes often generate a data of mixed-type. An example of such a mixed-type data could be a hospital's patients' dataset where typical fields include age (discrete or continuous), gender (binary), test results (binary or continuous), height (continuous) etc. In practice a number of ad-hoc solutions are used to handle mixed-type data [6]. However, the ideal general solution for analysing such heterogeneous data is to specify a model that builds a joint distribution with the assumption of an appropriate noise distribution for each type of feature set (for example a Bernoulli for modelling binary, a multinomial for modelling multi-category features and a Gaussian for modelling continuous features) and then fitting the model to data where the parameter estimates are used to draw inferences [6].

In the literature there is no multivariate distribution that can model random variables of different types. However, one possible way of jointly modelling discrete and continuous features is using a latent variable approach to understand the correlation between features of different types in combination. Type-specific latent variable models have already been proposed such as a generative topographic mapping (GTM) appropriate for continuous features and a latent trait model (LTM) appropriate for discrete

type features [7] (an LTM was proposed as a generalisation of GTM model). This has encouraged us to recently propose to combine GTM and LTM [13] in a probabilistic non-linear latent variable model in a principled way to visualise mixed-type data on a single continuous latent space under a unified proposed framework of conditional independence criteria: we called this model a generalised-GTM (GGTM).

In principle, the machine learning algorithms assume to perform well in cases where we have more information about data instances. This suggests that the use of more features is important for the learning algorithms. However, in practice it is observed that not all the features are important. It is therefore useful to select a subset of features which are relevant thereby ignoring the irrelevant (noisy) features which compromise performance of the learning algorithm [8,9]. An understanding of which features are relevant is valuable in its own right. In the exploratory phases of analysis (which is when data visualisation is most used) it is usual to measure as many variables as is feasible, since it is not known which features are relevant to the task. Feature selection then plays an important role in simplifying the task and making data collection cheaper and faster.

Feature selection (FS) has been widely used in supervised learning problems where the search is guided by the known target values. FS methods can be categorized into four classes [1,14]: filters, wrappers, hybrid and embedded. Details of each of them are given in [10,14]. FS for unsupervised learning algorithms is a challenging task as there are no target values to guide the search. Very few attempts have been made to estimate the importance of features in the unsupervised learning algorithms. A brief review of feature selection in a clustering perspective is given in [1,4,8,14,18] and details of some previous attempts in the latent variable formalism are given in [5,9,11,15,16,17]. To the best of our knowledge, there is no similar approach in the literature for estimating feature saliency when modelling mixed-type data, though [4] did discuss this as a possible extension in the clustering perspective.

Our focus in this paper is to demonstrate an extension of the GGTM model to estimate feature saliency values not only for discrete type features but also for mixed-type features in a dataset, as an integrated part of the parameter learning process, under the latent variable formalism. The structure of the remainder of the paper is as follows. In Section 2, we explain our proposed GGTMFS model and derive the EM parameter learning process. Section 3 describes our experiments to demonstrate the effectiveness of the proposed approach. We conclude the paper in Section 4.

2 A GGTM with Simultaneous Feature Saliency (GGTMFS)

The main goal of a latent variable model is to find an M -dimensional manifold, \mathcal{H} , (usually $M = 2$) for the distribution $p(\mathbf{x})$ in a high-dimensional data space, \mathcal{D} , with D -dimensions. We write each observation vector, \mathbf{x}_n in terms of sub-vectors $\mathbf{x}_n^{\mathcal{R}}$, $\mathbf{x}_n^{\mathcal{B}}$ and $\mathbf{x}_n^{\mathcal{C}}$ for continuous, binary and multi-category features respectively. In the rest of this paper we use superscript \mathcal{R} for continuous features, superscript \mathcal{B} for binary features and superscript \mathcal{C} for categorical features. The symbol $|\cdot|$ is used to indicate the number of features in each type of data space. We also use \mathcal{M} to indicate either \mathcal{R} or \mathcal{B} or \mathcal{C} .

In this paper, we now propose an extension of the GGTM visualisation model described in [10] to simultaneously estimate feature saliencies (we call this extension GGTMFS) and learn the model parameters. To estimate feature saliency values, we assume that each feature is independent of the component label under the appropriate noise model distribution. As a special case for the Gaussian noise model, the feature independence assumption is modelled by adopting diagonal covariance matrices (as used in [8,9]) instead of spherical covariance (as used in [3] and GGTM). Now the probability density function of the GGTMFS model takes the form

$$p(\mathbf{x}_n|\pi, \Theta) = \sum_{k=1}^K \pi_k \left[\prod_{\mathcal{M}} \left[\prod_{d=1}^{|\mathcal{M}|} p(x_{nd}^{\mathcal{M}}|\Theta_{kd}^{\mathcal{M}}) \right] \right], \quad (1)$$

where $p(\cdot|\Theta_{kd}^{\mathcal{M}})$ is the probability density functions of the d th feature for the k th component and π_k is the mixing coefficient of the k th component and is taken to be fixed to $\frac{1}{K}$ for all the components in the mixture model and $\mathcal{M} \in \{\mathcal{R}, \mathcal{B}, \mathcal{C}\}$ indicates type of data space (and the corresponding distributional assumption).

We make the definition that $\Psi^{\mathcal{M}} = (\psi_1^{\mathcal{M}}, \dots, \psi_{|\mathcal{M}|}^{\mathcal{M}})$ (where $\mathcal{M} \in \{\mathcal{R}, \mathcal{B}, \mathcal{C}\}$), is the set of binary indicators $\psi_d^{\mathcal{M}} = 1$ for a relevant feature and $\psi_d^{\mathcal{M}} = 0$ otherwise. Combining $\psi_d^{\mathcal{M}}$ for each type of variable, we obtain $\Psi = \{\Psi^{\mathcal{R}}, \Psi^{\mathcal{B}}, \Psi^{\mathcal{C}}\}$. Now the probability density of our mixture model takes the form

$$p(\mathbf{x}_n|\pi, \Theta, \lambda, \Psi) = \sum_{k=1}^K \pi_k \left[\prod_{\mathcal{M}} \left[\prod_{d=1}^{|\mathcal{M}|} [p(x_{nd}^{\mathcal{M}}|\Theta_{kd}^{\mathcal{M}})]^{\psi_d^{\mathcal{M}}} [q(x_{nd}^{\mathcal{M}}|\lambda_d^{\mathcal{M}})]^{(1-\psi_d^{\mathcal{M}})} \right] \right]. \quad (2)$$

The common distribution $q(x_{nd}^{\mathcal{M}}|\lambda_d^{\mathcal{M}})$ is designed to explain all the data that is poorly explained by the GGTM model. The notion of feature saliency is modelled as follows: we first treat $\psi_d^{\mathcal{M}}$ as a missing variable in the EM algorithm and as a second step we estimate the feature saliency, $\rho_d^{\mathcal{M}} = p(\psi_d^{\mathcal{M}} = 1)$, which is the probability that the d th feature is relevant. The resulting model now takes the form,

$$p(\mathbf{x}_n|\Omega) = \sum_{k=1}^K \pi_k \left[\prod_{\mathcal{M}} \left[\prod_{d=1}^{|\mathcal{M}|} [\rho_d^{\mathcal{M}} p(x_{nd}^{\mathcal{M}}|\Theta_{kd}^{\mathcal{M}})] + [(1 - \rho_d^{\mathcal{M}})q(x_{nd}^{\mathcal{M}}|\lambda_d^{\mathcal{M}})] \right] \right], \quad (3)$$

where $\Omega = \{\pi_k, \{\Theta_{kd}^{\mathcal{M}}\}, \{\lambda_d^{\mathcal{M}}\}, \{\rho_d^{\mathcal{M}}\}\}$ is the set of all the parameters of the model.

A simple way to understand how Equation (3) is obtained is to observe that $[p(x_{nd}^{\mathcal{M}}|\Theta_{kd}^{\mathcal{M}})]^{\psi_d^{\mathcal{M}}} [q(x_{nd}^{\mathcal{M}}|\lambda_d^{\mathcal{M}})]^{1-\psi_d^{\mathcal{M}}}$ can be re-written as $\psi_d^{\mathcal{M}} [p(x_{nd}^{\mathcal{M}}|\Theta_{kd}^{\mathcal{M}})] + (1 - \psi_d^{\mathcal{M}}) [q(x_{nd}^{\mathcal{M}}|\lambda_d^{\mathcal{M}})]$ given that $\psi_d^{\mathcal{M}}$ is a binary indicator variable (see the proof in [10,12]). The log-likelihood now takes the form

$$\mathcal{L}(\Omega) = \sum_{n=1}^N \ln p(\mathbf{x}_n|\Omega). \quad (4)$$

2.1 An EM algorithm for GGTMFS

The latent structure of the GGTM model can be exploited to estimate feature saliencies, in a similar way as previously exploited for the standard GTM [9]. For this purpose, we consider flipping of a biased coin with probability ρ_d^M ; if the coin is a head then the feature is generated from the mixture component, $p(\cdot|\Theta_{kd}^M)$, otherwise the ‘background component’, $q(\cdot|\lambda_d^M)$, is responsible.

We treat \mathbf{Y} (i.e. component labels) and Ψ as missing variables and we can derive an EM algorithm for estimating model parameters (see details in [10,12]). In the **E-step**, we use the current set of parameters, Ω , to compute the posterior probability (i.e. responsibility) $r_{nk} = p(y_n = k|\mathbf{x}_n)$ using Bayes’ theorem,

$$\pi_k \frac{\left[\prod_{\mathcal{M}} \left[\prod_{d=1}^{|\mathcal{M}|} [\rho_d^M p(x_{nd}^M|\Theta_{kd}^M)] + [(1 - \rho_d^M)q(x_{nd}^M|\lambda_d^M)] \right] \right]}{\sum_{k=1}^K \pi_k \left[\prod_{\mathcal{M}} \left[\prod_{d=1}^{|\mathcal{M}|} [\rho_d^M p(x_{nd}^M|\Theta_{kd}^M)] + [(1 - \rho_d^M)q(x_{nd}^M|\lambda_d^M)] \right] \right]}. \quad (5)$$

The responsibility matrix, \mathbf{R} , is used to compute $u_{nk}^M = p(\psi_d^M = 1, y_n = k|\mathbf{x}_n^M)$, which is a measure of the importance of the n th data point relating to the k th component using the d th feature of the \mathcal{M} type observation space and $v_{nk}^M = p(\psi_d^M = 0, y_n = k|\mathbf{x}_n^M)$.

$$u_{nk}^M = \frac{\rho_d^M p(x_{nd}^M|\Theta_{kd}^M)}{\rho_d^M p(x_{nd}^M|\Theta_{kd}^M) + [(1 - \rho_d^M)q(x_{nd}^M|\lambda_d^M)]} r_{nk}, \quad (6)$$

$$v_{nk}^M = r_{nk} - u_{nk}^M. \quad (7)$$

M-step: We can use \mathbf{U}^M to re-estimate the weight matrix \mathbf{W}^M (i.e. \mathcal{M} indicate type of observation space) using a set of linear equations. Both for binary and multinomial cases we use gradient-based approach as used in [7]. The weight vector \mathbf{w}_d^M of each d th feature can be updated using

$$\widehat{\mathbf{w}}_d^R = (\Phi^T \mathbf{E}_d^R \Phi)^{-1} \Phi^T \mathbf{U}_d^R \mathbf{x}_d^R, \quad (8) \quad \Delta \mathbf{w}_d^B \propto \Phi^T \left[\mathbf{U}_d^B \mathbf{x}_d^B - \mathbf{E}_d^B g^B(\Phi \mathbf{w}_d^B) \right], \quad (9)$$

$$\Delta \mathbf{W}_{S_d}^C \propto \Phi^T \left[\mathbf{U}_d^C \mathbf{X}_{S_d}^C - \mathbf{E}_d^C g^C(\Phi \mathbf{W}_{S_d}^C) \right], \quad (10)$$

where Φ is a $K \times L$ matrix, \mathbf{U}_d^M is a $K \times N$ matrix calculated using Equation (6), \mathbf{x}_d^M is an $N \times 1$ data vector of real/binary values (the $\mathbf{X}_{S_d}^C$ is binary encoded matrix of d th multi-category feature) and the diagonal matrix \mathbf{E}_d^M has values $e_{kkd}^M = \sum_{n=1}^N u_{nk}^M$.

Now we can straightforwardly re-estimate parameters of the mixture model using the re-estimated weight matrix of each type, $\widehat{\mathbf{W}}^M$: first we re-estimate the centres (for each type features) of the mixture model in the data space (see Equations (11), (12) and (13))

$$\widehat{Mean}\Theta_k^R = \widehat{\mathbf{m}}_k^R = \Phi(\mathbf{z}_k) \widehat{W}^R, \quad (11)$$

$$\widehat{\mathbf{m}}_k^{\mathcal{B}} = g^{\mathcal{B}}(\Phi(\mathbf{z}_k)\widehat{\mathbf{W}}^{\mathcal{B}}), \quad (12) \quad \widehat{\mathbf{m}}_{k,S_d}^{\mathcal{C}} = g^{\mathcal{C}}(\Phi(\mathbf{z}_k)\mathbf{w}_{S_d}^{\mathcal{C}}), \quad (13)$$

where $\widehat{\mathbf{m}}_k^{\mathcal{M}}$ is a $1 \times |\mathcal{M}|$ vector, $g^{\mathcal{B}}(\cdot)$ is a logistic sigmoid and $g^{\mathcal{C}}(\cdot)$ is a *softmax* function. We use re-estimated centre to update the diagonal Gaussian width in each direction (for each continuous feature): see Equation (14) (similar to standard GTMFS [9])

$$\frac{1}{\widehat{\beta}_d^{\mathcal{R}}} = \frac{\sum_k \sum_n u_{nk d}^{\mathcal{R}} (x_{nd}^{\mathcal{R}} - \widehat{m}_{kd}^{\mathcal{R}})^2}{\sum_k \sum_n u_{nk d}^{\mathcal{R}}}. \quad (14)$$

Common density parameters, $\lambda_d^{\mathcal{R}}$, can be updated using

$$\widehat{Mean}\lambda_d^{\mathcal{R}} = \frac{\sum_n (\sum_k v_{nk d}^{\mathcal{R}}) x_{nd}^{\mathcal{R}}}{\sum_{nk} v_{nk d}^{\mathcal{R}}}. \quad (15)$$

$$\widehat{Mean}\lambda_d^{\mathcal{B}} = \frac{\sum_n (\sum_k v_{nk d}^{\mathcal{B}}) x_{nd}^{\mathcal{B}}}{\sum_{nk} v_{nk d}^{\mathcal{B}}}. \quad (16) \quad \widehat{Mean}\lambda_{S_d}^{\mathcal{C}} = \frac{\sum_n (\sum_k v_{nk d}^{\mathcal{C}}) x_{nS_d}^{\mathcal{C}}}{\sum_{nk} v_{nk d}^{\mathcal{C}}}. \quad (17)$$

$$\widehat{Var}\lambda_d^{\mathcal{R}} = \frac{\sum_n (\sum_k v_{nk d}^{\mathcal{R}} (x_{nd}^{\mathcal{R}} - \widehat{Mean}\lambda_d^{\mathcal{R}})^2)}{\sum_{nk} v_{nk d}^{\mathcal{R}}}. \quad (18)$$

For the feature saliency parameter update, we use prior distributions for each type of variable separately as explained in [12]. The resultant feature saliency updates are

$$\widehat{\rho}_d^{\mathcal{R}} = \frac{\max(\sum_{nk} u_{nk d}^{\mathcal{R}} - \frac{KP}{2}, 0)}{\max(\sum_{nk} u_{nk d}^{\mathcal{R}} - \frac{KP}{2}, 0) + \max(\sum_{nk} v_{nk d}^{\mathcal{R}} - \frac{T}{2}, 0)}, \quad (19)$$

where P and T are the number of parameters in $\Theta_{kd}^{\mathcal{R}}$ and $\lambda_d^{\mathcal{R}}$ respectively.

$$\widehat{\rho}_d^{\mathcal{B}} = \frac{\max(\sum_{nk} u_{nk d}^{\mathcal{B}} + \alpha_d - 1, 0)}{\max(\sum_{nk} u_{nk d}^{\mathcal{B}} + \alpha_d - 1, 0) + \max(\sum_{nk} v_{nk d}^{\mathcal{B}} + \beta_d - 1, 0)}. \quad (20)$$

$$\widehat{\rho}_d^{\mathcal{C}} = \frac{\max(\sum_{nk} u_{nk d}^{\mathcal{C}} - \frac{K(c_d-1)}{2}, 0)}{\max(\sum_{nk} u_{nk d}^{\mathcal{C}} - \frac{K(c_d-1)}{2}, 0) + \max(\sum_{nk} v_{nk d}^{\mathcal{C}} - \frac{(c_d-1)}{2}, 0)}, \quad (21)$$

where c_d represents number of categories for the d th feature. We also extend GGTMFS by deriving an *expectation-maximisation* (EM) variant to incorporate missing values (for details see the technical report [12]).

3 Experiments

A series of experiments was performed to demonstrate the effectiveness of the proposed GGTMFS model for both synthetic and real datasets. Each weight sub-matrix (i.e. $\mathbf{W}^{\mathcal{R}}$, $\mathbf{W}^{\mathcal{B}}$ and $\mathbf{W}^{\mathcal{C}}$) was initialised using principal component analysis (PCA). On average, 500 iterations of EM were sufficient for convergence. We used a latent grid of size 8×8 and an RBF grid of size 4×4 .

3.1 Synthetic data

A synthetic data was used to assess the GGTMFS model: a combination of continuous and binary features. A comparison of the resulting projections with those given by the GGTM model is also shown on both complete and incomplete data where 10% of the data was removed at random for each observation space. We first generated 2 feature dataset with 2000 data points from an equiprobable mixture of four Gaussians (for details see technical report [12]) and then generated 8 noisy features (where each feature was sampled independently from $\mathcal{N}(0, I)$ distribution) and combined them yielding a 10-feature dataset. We then generated a binary dataset of 100 features where the first 40 features were drawn from four equiprobable clusters and the remaining 60 features are noisy (with random distribution of 1s). A small amount of noise (5%) was added by inserting random 0s in the informative features. For the uninformative features, we added a random distribution of 1s with different percentages by 20%, 40%, 60%, 80% and also with no or all 1s in the uninformative features (and we report here results of binary uninformative features with no 1s). We then combined both continuous and binary features yielding a dataset with 110 features.

Visualisation results for GGTM and GGTMFS and saliency values estimated from the GGTMFS are presented in Figure 1 for both complete and incomplete data.

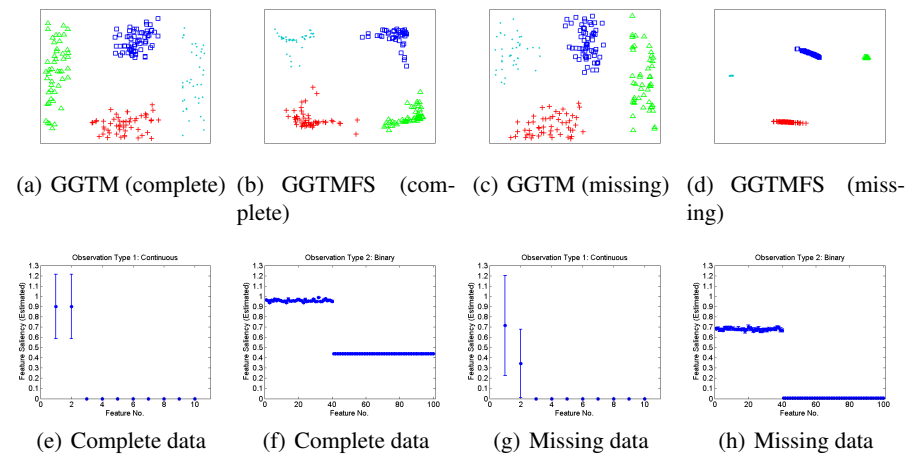


Fig. 1. GGTM and GGTMFS visualisations and saliencies of the synthetic mixed-type complete and missing datasets with 10 continuous and 100 binary features. 10% of the continuous and binary complete data have been randomly removed to produce the missing data. GGTMFS visualisations quite often show more compact clusters compared to GGTM visualisations. Saliencies plots show results as error bars from our cross-validation results. (e) and (g) show FS values of continuous features whereas (f) and (h) show FS values of binary features.

3.2 Real-world data: oil exploration

We applied the GGTMFS to two sets of oil exploration data from the Barents Sea: oil maturity and environmental parameters. The maturity data consists of 17 continu-

ous features with a large fraction (34%) of missing values and the environment data contains 13 continuous features with 16% of missing values. Both datasets consist of 168 samples. All the variables in the maturity dataset are important except feature 7, which has an environment influence that might make it behave differently, and feature 17, which is an environmental parameter. The features in the environment data are a combination of geochemical properties with variable importance. Features 6, 7, 11 and 12 are more influenced by maturity than environment and hence their saliency values should be low.

The resulting GGTMFS visualisation and saliency values plots are shown in Figure 2. The GGTMFS plots are superimposed with magnification factor plots which enable the user to observe the amount of stretching of the data-space manifold at different parts of the latent space [2]. This is useful to understand how the data is embedded in the data space, detect outliers and separate clusters. The magnification factors are represented by colour shading in the projection manifold: the lighter the colour, the more stretch in the projection manifold. The GGTMFS visualisations on the oil data show

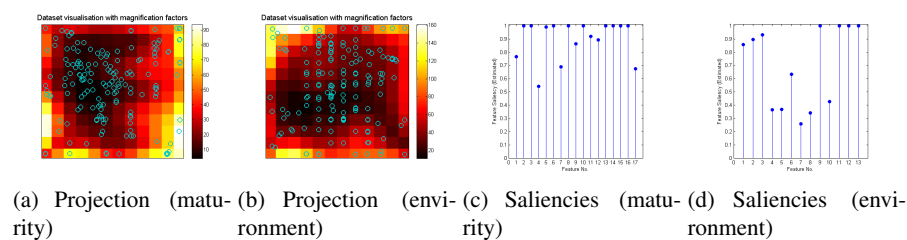


Fig. 2. GGTMFS plot of the maturity (see (a)) and environment data (see (b)) and estimated feature saliency values of continuous features (see (c) and (d)).

that there is relatively little discrete structure (no clear clusters) in the data. The model was able to give a sensible saliency value (0.675) to the feature 17 in the maturity data as this variable is actually an environmental parameter. However, feature 1 should have a high saliency value according to the domain experts. In the environment data, the low saliency values of the features 6 and 7 make sense given that these features have a strong maturity influence. However, features 11 and 12 should also have low saliency values, and feature 10 should have a high saliency value.

4 Conclusion

We derived a non-linear model for visualising a mixed-type dataset to simultaneously estimate saliency values both for complete and incomplete datasets. We called this model a generalised GTM with simultaneous feature saliency estimation (GGTMFS). Experimental visualisation results for both synthetic and real mixed-type datasets have shown that this model, unlike GTM, provided more compact clusters especially in the presence of missing values and irrelevant features. More detailed results with other datasets are available in a technical report [12].

References

1. S. Alelyani, J. Tang, and H. Liu. Feature selection for clustering: A review. In *Data Clustering: Algorithms and Applications*, pages 29–60. Chapman and Hall/CRC, 2013.
2. C. Bishop, M. Svensén, and C. K. I. Williams. Magnification factors for the GTM algorithm. In *In Proceedings IEE Fifth International Conference on Artificial Neural Networks*, pages 64–69, 1997.
3. C. M. Bishop and M. Svensen. GTM: The generative topographic mapping. *Neural Computation*, 10(1):215–234, 1998.
4. N. Bouguila. On multivariate binary data clustering and feature weighting. *Comput. Stat. Data Anal.*, 54(1):120–134, 2010.
5. I. O. Caparrosó. *Variational Bayesian algorithms for generative topographic mapping and its extensions*. PhD thesis, Universitat Politècnica de Catalunya, 2008.
6. A. R. de Leon and K. C. Chough. *Analysis of Mixed Data: Methods & Applications*. Taylor & Francis Group. Chapman and Hall/CRC, 2013.
7. A. Kabán and M. Girolami. A combined latent class and trait model for the analysis and visualization of discrete data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(8):859–872, 2001.
8. M. H. C. Law, M. A. T. Figueiredo, and A. K. Jain. Simultaneous feature selection and clustering using mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1154–1166, 2004.
9. D. M. Maniyar and I. T. Nabney. Data visualization with simultaneous feature selection. In *Computational Intelligence and Bioinformatics and Computational Biology, 2006. CIBCB '06. 2006 IEEE Symposium on*, pages 1–8, 2006.
10. S. Mumtaz. *Visualisation of bioinformatics datasets*. PhD thesis, Aston University, 2015.
11. S. Mumtaz, I. T. Nabney, and D. R. Flower. Novel visualization methods for protein data. In *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), 2012*, pages 198–205, May 2012.
12. S. Mumtaz, M. F. Randrianandrasana, and I. T. Nabney. Mixed-type data visualisation and simultaneous feature saliency estimation using generalised generative topographic mapping. Technical report, Systems Analytics and Research Institute (SARI), Aston University, Birmingham, United Kingdom, 2015.
13. M. F. Randrianandrasana, S. Mumtaz, and I. T. Nabney. Visualisation of heterogeneous data with the generalised generative topographic mapping. In *Proceedings of the Tenth International Conference on Information Visualization Theory and Application*, pages 233–238, 2015.
14. C. M. V. Silvestre, M. M. G. Cardoso, and M. A. T. Figueiredo. Clustering and selecting categorical features. In *EPIA*, volume 8154 of *Lecture Notes in Computer Science*, pages 331–342. Springer, 2013.
15. A. Vellido. Preliminary theoretical results on a feature relevance determination method for generative topographic mapping. Technical report, Universitat Politècnica de Catalunya (UPC)LSI-05-13-R, Barcelona, Spain, 2005.
16. A. Vellido. Assessment of an unsupervised feature selection method for generative topographic mapping. In *Proceedings of the 16th International Conference on Artificial Neural Networks - Volume Part II, ICANN'06*, pages 361–370, Berlin, Heidelberg, 2006. Springer-Verlag.
17. A. Vellido, P. J. G. Lisboa, and D. Vicente. Robust analysis of MRS brain tumour data using t -GTM. *Neurocomputing*, 69(7-9):754–768, 2006.
18. X. Wang and A. Kabán. Finding uninformative features in binary data. In *Intelligent Data Engineering and Automated Learning - IDEAL 2005*, volume 3578 of *Lecture Notes in Computer Science*, pages 40–47. Springer Berlin Heidelberg, 2005.

Incremental Class Learning and Novel Class Detection of Gestures Using Ensemble

Husam Al-Behadili^{1,2}, Arne Grumpe², Christian Dopp², and Christian Wöhler²

¹ Engineering College, University of Mustansiriyah
Baghdad Al-Mustansiriyah, Box 46007, Iraq

² Image Analysis Group, TU Dortmund University
Otto-Hahn-Str. 4, D-44227 Dortmund, Germany

Abstract. We propose a novel classification system which has the ability to integrate new classes into its class structure and thus updates the classifier architecture. The proposed algorithm uses the Mahalanobis distance in conjunction with the confidence bands of a polynomial classifier to obtain class labels for new unlabeled training samples even if they do not belong to the known classes. If a new sample belongs to existing classes, it will be automatically included into the training set, otherwise a new label will be assigned to the “novel” sample, and a new class will be added to the existing classes in the training set. Samples belonging to random motion patterns rather than meaningful gestures are rejected as they do not occur repeatedly in a similar manner. Since the Mahalanobis metric depends on the sample mean and covariance matrix, we derive a relation for recursively updating the covariance of the training set with new training samples. Our experimental evaluation shows that a new class can be successfully identified and added to the classifier structure and that the overall recognition rate of the proposed semi-supervised learning approach is lower by only a minor amount than that of the supervised version of the algorithm.

Keywords: Gesture recognition; Dynamic Time Warping; semi-supervised learning; distance measure; Nearest Class Mean; Addition of classes

1 Introduction

The recognition of gestures has become an important element of human-machine interaction. This work focusses on the automatic recognition of 3D emblematic gestures performed with the arm acquired with a Kinect sensor. Commonly, gestures of the same class are performed in a different manner by different persons, such that it is usually not possible to train the system with all instances of a gesture that may in principle occur. Even if a gesture classification system is trained in a fully supervised manner on a large variety of gestures, a new user who is unknown to the system will have to adapt his/her way to perform the gestures such that it corresponds to the “knowledge” of the system.

As an alternative to such a closed-world gesture recognition system, we intend to build a classification system which adapts itself to new users without the

need for intervention of a human operator, such that each user can train the system with a set of own favored gestures and add new gesture classes over time. Hence, when the system perceives new gestures under specific conditions, it will construct a new class and add it to the training data, while otherwise, if the gesture category has already been seen in the training phase, the new sample will be added to the specific class in the training data according to the self-learning paradigm [29].

Various approaches have been developed that increment the database by new samples or new classes without the need for processing again the previously acquired training data, but they still need labelled data (e.g. [28, 6]).

Metric learning is important in a variety of unsupervised learning approaches, like k -nearest neighbor (kNN), where the distance from the test sample to each sample in the training set is determined [5], or prototype-based learning with adaptive distance metric [17]. Similarly, the Nearest Class Mean (NCM) approach described in [26] uses the Mahalanobis distance of the test sample from the training samples, obtained based on the covariance matrix of the training set, for determining similarities between samples. The Nearest Class Mean Multi-class Logistic Discrimination (NCMML) framework introduced in [12] extends the NCM method in that an optimal projection is determined that enforces the sample within the same class to be closer to its class mean than samples from other classes.

In this study, we use the class-specific Mahalanobis distance to compute the distance between a test sample or unlabeled sample and each class, similar to the NCM framework. Based on several thresholds (one for each class) which are optimized by the greedy algorithm, the sample is assigned to a certain class. If the distance does not meet any threshold criterion, it will be considered a possibly “novel” sample. If additionally the confidence band width of a polynomial classifier exceeds a given threshold, the sample will be marked as “novel”. After several samples have been marked as “novel”, they will either be assigned to a new class or considered as random movements (“outliers”) based on the mutual Mahalanobis distances between the novel samples. A new class will be constructed if the number of accordingly determined novel samples exceeds a minimum number. The new class is given a label and is included into the training set in the same way as the other newly labelled samples of the known classes, and the classifier is re-trained on this updated training set.

2 Data Acquisition and Feature Extraction

A well-known database of gestures acquired with the Kinect sensor is described in [15]. These gestures, however, are mainly performed with both hands simultaneously. The database in [10] comprises emblematic gestures of the hand and forearm performed on a plane. In order to be able to develop a classification system that copes with 3D trajectories but avoids the additional complexity of

two-arm gestures, we used the data set by [1] which comprises 3D trajectories performed with a single hand³.

The gestures in this data set were performed by ten persons. The samples belong to nine classes, where each person performed between 15 and 20 gestures of each class, half of them with the right and with the left hand, respectively. The data set contains 1662 gestures altogether. The classes of emblematic gestures are “circle”, “point”, “stop”, “come here”, “go away”, “up”, “down”, “wave”, and “wave vertically”; this structure was also adopted by [18] and may e.g. be useful for transferring commands to a mobile robot. These gestures had been already segmented by using the idea of the spotting algorithm [20]. The first three features correspond to the mean 3D position of the moving wrist in the specific gesture. The second three features are the extensions of the gesture in x, y and z direction. The Dynamic Time Warping (DTW) algorithm [13] is applied to each gesture to normalize it in the time domain. The DTW approach aims for warping a temporal sequence of data points such that it optimally corresponds with a reference sequence. The cost value corresponding to such a transformation can be viewed as a distance and depends on the sequence of the mutual assignments between the points of the sequences (“warping path”). It is minimized by Dynamic Programming (see [13] for details). Accordingly, the seventh feature is the minimum DTW distance between a gesture and the templates and the eighth feature is the class label of the most similar class according to that distance.

3 Novelty Detection

3.1 Support vector data description (SVDD)

Tax and Duin [22] proposed a kernel-based algorithm called support vector data description (SVDD) for detecting the outliers of a known class, thus representing a one-class classifier. In the SVDD approach, the novelty boundary of the class is obtained by computing the hypersphere of minimal volume enclosing all possible samples of the known class. Tax and Laskov [23] proposed an incremental SVDD method which allows for adding new samples to or deleting old samples from the trained classifier. Support vector based novelty detection has been examined in detail in [3]. Tavakkoli et al. [21] mentioned the limitations of the SVDD for large training data sets due to the optimization process. To solve this problem several extensions of SVDD have been proposed in different applications [e.g. 8, 11, 2, 14, 21].

3.2 The Proposed Semi-supervised Learning Framework with Adaptive Class Structure

The proposed algorithm is supposed to have the ability to classify new samples according to whether they belong to known classes or belong to a “novel” class

³ The dataset is available at <http://www.bv.e-technik.tu-dortmund.de>

which is unknown so far. The novel gestures are collected in order to construct a new class which is added to the database, where real gestures need to be distinguished from random movements (outliers). Independent of whether the sample belongs to a new or to a known class, it is included into the training set, which is then used for re-training the classifier. The classification itself is performed by a polynomial classifier [19, 7] and metric learning using Mahalanobis distance [26]. According to [19], the discriminant function of the polynomial classifier is

$$d(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \cdot p(\mathbf{x}). \quad (1)$$

where \mathbf{w} is the model parameters vector and the $p(\mathbf{x})$ represents the polynomial basis vector which is constructed from input feature vector \mathbf{x} depending on the polynomial degree k . As shown in [19], if e.g. the input vector $\mathbf{x} = [x_1 \ x_2]^T$ and $k = 2$ the polynomial basis vector will be

$$p(\mathbf{x}) = [1 \ x_1 \ x_2 \ x_1^2 \ x_1x_2 \ x_2^2]^T. \quad (2)$$

The model parameter vector \mathbf{w} is calculated using the training data and their labels. The polynomial classifier output (decision value) is real value between 0 and 1 which is used as the probabilities of \mathbf{x} belong to each of known classes [19]. Additionally the confidence band of the polynomial classifier is used as derived in [1] based on the linear method described in [9].

The sample \mathbf{x} is considered to belong to a specific class depending on the minimum Mahalanobis distance between the sample and the classes' centroid. Depending on a class-specific distance threshold whose optimization is described later in this section, the sample is considered as a "novel-candidate" gesture if it is found to not belong to any known class. The candidate novelty is then assigned to the known class associated with the largest decision value of the polynomial classifier only if the inequality

$$p_1 - p_2 \geq \alpha(\eta_1 + \eta_2) \quad (3)$$

is fulfilled (as proposed in [16]), where p_1 and p_2 are the largest and second largest decision value of the polynomial classifier, η_1 and η_2 the corresponding confidence band widths and α is a given constant. The remaining candidate novelties are passed to the next step to check if they belong to the novel class or are just outliers.

The covariance matrix of the above novel-candidates distribution in the feature space is computed when a sufficient number of novel samples has been identified, i.e. at least two times the number of features (here 2×8). Based on this covariance matrix, it is possible to reject a novel sample as a random movement if the Mahalanobis distance from their mean exceeds a given threshold. According to this rule, a new class is opened up only if an unknown gesture type occurs several times in a similar form. The training set is then extended by the new class and used to re-train the polynomial classifier.

The described scheme depends on number of thresholds whose number corresponds to the number of known classes plus one for the newly constructed

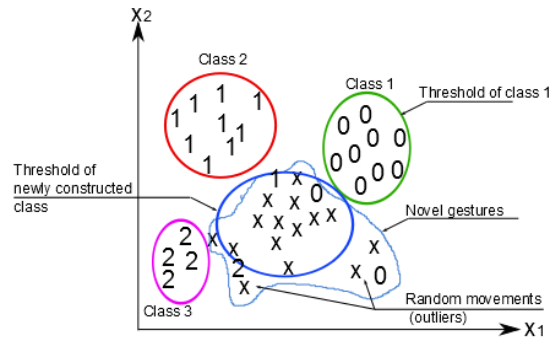


Fig. 1. Illustration of the definition of the “novel” class and random movements (outliers).

class. These thresholds are always positive as they represent distances or confidence band widths, and an upper limit of 500 has been determined based on the available data set.

Suppose the total number of samples which belong to the newly constructed class is $N_{\text{new}}^{\text{total}}$, and the number of samples in the newly constructed class but actually belonging to one of the known classes or representing random movements or gestures (outliers) is $N_{\text{new}}^{\text{false}}$. To optimize the thresholds θ_i , $N_{\text{new}}^{\text{total}}$ should be maximized while at the same time $N_{\text{new}}^{\text{false}}$ should be minimized. This approach leads to the error term

$$E = (-N_{\text{new}}^{\text{total}} + \beta \cdot N_{\text{new}}^{\text{false}}) / (1 + \beta) \quad (4)$$

where the parameter β denotes the relative importance of the maximization of the number of novel samples vs. the minimization of the number of erroneously assigned novel samples. In our experiments we assumed both criteria as equally important, resulting in $\beta = 1$. The definition of the novel class is illustrated in Fig. 1.

To find the optimal configuration of thresholds we first apply the golden section algorithm [4] to each threshold successively while keeping the other thresholds constant. This procedure is repeated two or three times. The lower limits of the resulting section intervals are used as initial values for a greedy optimization scheme [4]. In our approach we employ the greedy algorithm as described in [27]. Accordingly, each threshold θ_i is successively increased and decreased by a predefined amount $\Delta\theta$. If the error decreases after changing the threshold θ_i , the new value is kept, otherwise the value of θ_i remains unchanged. This procedure is repeated until the error stops decreasing.

Since computing the Mahalanobis distance requires knowledge about the mean and the covariance of the n -dimensional training data only but not the training data themselves, we implement our training approach as an online scheme and update the mean and covariance matrix recursively. For this purpose we derive a relation for the mean $\boldsymbol{\mu}_{\text{total}}$ and the covariance matrix $\mathbf{C}_{\text{total}}$ of a set

of $m_{\text{total}} = m_{\text{old}} + m_{\text{new}}$ samples based on the mean $\boldsymbol{\mu}_{\text{old}}$ and covariance matrix \mathbf{C}_{old} of the subset of the first m_{old} samples and the mean $\boldsymbol{\mu}_{\text{new}}$ and covariance matrix \mathbf{C}_{new} of the subset of the last m_{new} samples. For $\boldsymbol{\mu}_{\text{total}}$ one obtains

$$\begin{aligned}\boldsymbol{\mu}_{\text{total}} &= \frac{1}{m_{\text{total}}} \sum_{i=1}^{m_{\text{total}}} \mathbf{x}_i = \frac{1}{m_{\text{total}}} \left[\sum_{i=1}^{m_{\text{old}}} \mathbf{x}_i + \sum_{i=m_{\text{old}}+1}^{m_{\text{total}}} \mathbf{x}_i \right] \\ &= \frac{1}{m_{\text{total}}} (m_{\text{old}} \cdot \boldsymbol{\mu}_{\text{old}} + m_{\text{new}} \cdot \boldsymbol{\mu}_{\text{new}})\end{aligned}\quad (5)$$

as shown in [19], where \mathbf{x}_i denotes the i th sample vector in the ordered combined dataset, i.e. the last m_{new} samples are the added samples. For the covariance matrix one obtains

$$\begin{aligned}\mathbf{C}_{\text{total}} &= \frac{1}{m_{\text{total}}} \left[\sum_{i=1}^{m_{\text{old}}} (\mathbf{x}_i - \boldsymbol{\mu}_{\text{total}})(\mathbf{x}_i - \boldsymbol{\mu}_{\text{total}})^T \right. \\ &\quad \left. + \sum_{i=m_{\text{old}}+1}^{m_{\text{total}}} (\mathbf{x}_i - \boldsymbol{\mu}_{\text{total}})(\mathbf{x}_i - \boldsymbol{\mu}_{\text{total}})^T \right].\end{aligned}\quad (6)$$

Some simplifications yield

$$\mathbf{C}_{\text{total}} = \frac{m_{\text{old}}}{m_{\text{total}}} \mathbf{C}_{\text{old}} + \frac{m_{\text{new}}}{m_{\text{total}}} \mathbf{C}_{\text{new}} + \frac{m_{\text{old}} m_{\text{new}}}{m_{\text{total}}^2} (\boldsymbol{\mu}_{\text{new}} - \boldsymbol{\mu}_{\text{old}})(\boldsymbol{\mu}_{\text{new}} - \boldsymbol{\mu}_{\text{old}})^T. \quad (7)$$

When one single sample is added at a time, Eq. (7) further reduces to

$$\mathbf{C}_{\text{total}} = \frac{m_{\text{old}}}{m_{\text{old}} + 1} \mathbf{C}_{\text{old}} + \frac{m_{\text{old}}}{(m_{\text{old}} + 1)^2} (\mathbf{x} - \boldsymbol{\mu}_{\text{old}})(\mathbf{x} - \boldsymbol{\mu}_{\text{old}})^T. \quad (8)$$

The scheme of Eqs. (6)–(8) is related to but different from [19], where a direct incremental estimation of the inverse covariance matrix is derived.

4 Experiments and Results

Since it difficult to find a benchmark classifier that has the capability of incremental learning, novelty detection, and construction of new classes, we performed two experimental evaluations. In the first evaluation the ability of our proposed algorithm of constructing a new class is compared with a fully supervised classifier. The second experiment evaluates the ability of the classifier to detect all outliers and compares it with the SVDD classifier.

4.1 Evaluation of the New Class Construction

In this experiment an evaluation of the performance of the developed algorithm is performed regarding two scenarios. The first scenario corresponds to the fully supervised approach, which serves as a reference, and in the second scenario we

Table 1. Results of the proposed semi-supervised learning algorithm in comparison to the results of the fully supervised approach making use of all available manually set labels. The average values were computed over 100 training runs each, the error intervals correspond to plus/minus one standard deviation.

Excluded class	P [%]		Size ratio	
	semi-sup.	supervised	semi-supervised	supervised
class 1	97.6 ± 2.2	100.0 ± 0.0	99.7 ± 0.7	99.9 ± 0.3
class 2	89.0 ± 8.4	87.3 ± 3.7	49.9 ± 14.5	96.3 ± 1.4
class 3	93.1 ± 3.9	97.6 ± 2.1	93.1 ± 1.8	93.5 ± 2.3
class 4	95.4 ± 3.6	94.1 ± 2.0	70.2 ± 6.1	94.2 ± 4.1
class 5	99.9 ± 0.3	99.6 ± 0.5	96.0 ± 2.2	99.2 ± 0.6
class 6	68.6 ± 11.1	92.3 ± 3.6	21.0 ± 6.4	97.1 ± 1.5
class 7	98.4 ± 2.4	100.0 ± 0.0	99.3 ± 1.5	100.0 ± 0.0
class 8	99.1 ± 1.6	100 ± 0.0	99.9 ± 0.5	100.0 ± 0.2
class 9	92.4 ± 3.1	95.4 ± 1.7	85.3 ± 4.9	85.0 ± 4.4

Table 2. Overall recognition rate of the proposed semi-supervised learning algorithm compared to the recognition rate of fully supervised learning. The average values were computed over all 900 training runs, the error intervals correspond to plus/minus one standard deviation.

Overall recognition (semi-supervised)	Overall recognition (supervised)	Novel class recognition (semi-supervised)	Novel class recognition (semi-supervised)
94.5% ± 3.0	96.1% ± 0.7	79.4% ± 26.69	96.1% ± 5.11

used our developed algorithm. The available data were divided into three sets: the initial training set, the learning data set for which class labels are generated by our algorithm, and the test set. Each set comprises one third of the overall data set.

The training scheme is repeated in 9 sets of 100 runs each, where for the k th set all samples of class k are excluded from the initial training set of our algorithm while the dataset is kept complete for the supervised approach. The excluded class is considered as the novel class. In each run the data are divided into initial, learning and test data set in a different random manner. Initially, both classifiers are trained as fully supervised with their own initial training set. Labels are then generated for the learning (unlabeled) data set, and the corresponding samples are included into the training set together with the labels estimated by the classifier. The fully supervised classifier is trained on all classes. Hence, it can classify all samples in the learning set. In contrast, there are some samples in the learning set belonging to the unseen (excluded) class of our algorithm. However, it will classify them as “novel” and it will construct a new class containing some of these novel samples which fulfil the conditions of the new class, and it rejects the other novel gestures by assigning them as random movements (outliers). Let N_k the total number of samples in the learning set belonging to the excluded class k , and let $N_{\text{new}}^{\text{total}}$ and $N_{\text{new}}^{\text{true}}$ be the total amount of samples in the newly constructed class and the samples in the newly constructed class belonging to

class k , respectively. The results of the first stage of our algorithm, measured by the “purity” $P = N_{\text{new}}^{\text{true}}/N_{\text{new}}^{\text{total}}$ and the size ratio of the newly constructed class to the total size of class k ($N_{\text{new}}^{\text{total}}/N_k$), are shown in Table 1. In the case of the fully supervised classifier, the purity parameter of class k is the ratio between the correct classifier assignments to that class and its total number of samples. The size ratio of class k is computed by dividing the number of samples assigned to that class by N_k .

The obtained purity values are reasonably high (between 89% and 99.9%) for all novel classes except class 6. That class is also characterized by a low size ratio of only 21%, while for all other classes except class 2 the size ratio exceeds 70% and even reaches values larger than 99% for classes 1, 7 and 8. The purity values obtained by supervised learning, which is trained using samples of the novel class and is thus able to recognize the novel class, correspond to within a few percent to those obtained by the proposed semi-supervised method except for class 6. Regarding the size ratio, the comparison between the semi-supervised and the fully supervised learning results is quite encouraging since the recognition rate of supervised learning significantly exceeds that of the semi-supervised approach only for the apparently “difficult” classes 2, 4 and 6.

For the same 900 training runs of our algorithm, Table 2 shows the recognition rate achieved by our semi-supervised algorithm in comparison to the recognition rate of the same classifier trained in a fully supervised manner based on the same training and test data, where the average difference is only about 2.5% in favor of the fully supervised scenario. A larger difference of about 15% is observed for the recognition rate of the novel class.

This result is not unexpected since if a high purity of the novel class is desired using the employed Mahalanobis distance criterion, a considerable fraction of the possible novel samples may to be rejected as random movements (outliers). If it is desired to include more gestures into the new class, it is necessary to increase the corresponding threshold, which in turn will lead to a decreased purity. All in all, it depends on the application scenario if a large number of assigned novel samples or a high purity of the newly constructed class is desired.

4.2 Outlier Detection

The data has been divided in the same manner of the first experiment. In addition a small number of outliers has been added to the test data. The outliers samples doesn't belong to any of gesture classes and it has been created using the function “gendatout”. This function and the function “multic”, which it used together with the SVDD classifier to make it applicable in a multi-class system, are proposed by Tax [24] in the data description, outlier and novelty detection toolbox (ddtools)⁴. Both classifiers (the proposed classifier and the SVDD classifier) are trained on the training data and then they classify the test data. The proposed classifier assigns the test sample to one of three cases, which are: labelling it with the label of one of the known classes, indicate it as an outlier, or

⁴ http://prlab.tudelft.nl/david-tax/dd_tools.html [25]

Table 3. Results of the proposed semi-supervised learning algorithm in comparison to the results of the SVDD classifier. The average values were computed over 100 training runs each, the error intervals correspond to plus/minus one standard deviation.

Excluded class	Known Classes		Total Unknown Samples		Newly Con- structed Class	Outliers
	semi-sup.	SVDD	semi-sup.	SVDD	semi-sup.	semi-sup.
Class 1	93,1 ± 1,1	36,8 ± 19,9	99,6 ± 0,6	72,6 ± 41,0	100,0 ± 0,0	95,0 ± 7,1
Class 2	94,1 ± 1,1	36,7 ± 20,8	54,6 ± 11,6	55,6 ± 33,6	50,5 ± 12,7	96,4 ± 5,3
Class 3	93,8 ± 0,9	34,7 ± 23,2	97,8 ± 1,2	61,1 ± 43,1	96,1 ± 1,6	97,1 ± 5,2
Class 4	93,7 ± 1,0	41,4 ± 19,4	77,6 ± 4,5	35,5 ± 18,2	76,1 ± 4,9	92,7 ± 8,6
Class 5	93,0 ± 1,0	47,0 ± 2,3	95,8 ± 2,3	59,8 ± 6,6	95,7 ± 2,4	97,4 ± 4,6
Class 6	94,0 ± 1,0	41,6 ± 17,5	27,7 ± 5,6	48,1 ± 23,0	21,2 ± 6,1	96,0 ± 6,3
Class 7	92,7 ± 1,2	35,3 ± 18,6	99,2 ± 1,1	63,1 ± 34,8	99,5 ± 1,1	95,9 ± 6,2
Class 8	92,8 ± 1,1	34,3 ± 19,3	99,7 ± 0,5	73,4 ± 43,8	100,0 ± 0,0	96,6 ± 6,3
Class 9	93,1 ± 1,0	37,6 ± 21,9	92,2 ± 3,2	64,0 ± 39,6	92,0 ± 3,4	91,9 ± 8,1

indicate it as belonging to a new class. The SVDD classifier assigns it either as one of the known classes or as an outlier. Hence, the result is computed as the accuracy regarding the known classes and the accuracy of detecting unknown classes and outliers. These two measures are computed for both classifiers.

Another two measures are computed for the proposed classifier only, which are the accuracy of the newly constructed class and the accuracy of the rejected samples. The outliers which are added to the test data using the function “gendatout” of the ddtools toolbox [25] are considered as a reference to the outliers (rejected samples) of the proposed algorithm in computing the outlier detection accuracy. Consequently, the test samples which belong to the excluded class in the training data are considered as references to the newly constructed class.

The experiment was run 900 times. The summary of the results is shown in Table 3. The advantage of the proposed algorithm over the SVDD algorithm is clearly apparent from this table. Again both algorithms have difficulties in detecting the outliers of classes 2 and 6 as they strongly overlap with the other classes.

5 Summary and Conclusion

We have described a method for gesture recognition which has the ability to construct new classes in addition to the existing class structure by automatically extending the architecture of the classifier accordingly. Only mutually similar novel motion patterns repeatedly occurring in a similar manner may be considered as belonging to a new class, while random movements (outliers) are rejected. At each addition of an unlabeled sample to the training set, the class-specific mean and covariance matrix required to compute the corresponding Mahalanobis distance are updated in a computationally efficient recursive manner. Our experimental evaluation based on a set of gestures acquired with a Kinect sensor shows

that the new class can be identified fully autonomously with a quite reasonable average accuracy of about 80%, where the success rate of the new class may vary rather strongly for different novel classes. The overall recognition rate of our semi-supervised approach is only 2.5% lower than that of a fully supervised version of the learning algorithm.

References

- [1] Al-Behadili, H., Wöhler, C., Grumpe, A.: Semi-supervised learning of emblematic gestures. *AT-AUTOMATISIERUNGSTECHNIK* 62(10), 732–739 (2014)
- [2] Chen, S., He, H.: Towards incremental learning of nonstationary imbalanced data stream: a multiple selectively recursive approach. *Evolving Systems* 2(1), 35–50 (2011)
- [3] Clifton, L.A.: Multi-Channel Novelty Detection and Classifier Combination. Ph.D. thesis, University of Manchester, Manchester (2007)
- [4] Cormen, T.H.: Introduction to algorithms. MIT press (2009)
- [5] Cover, T., Hart, P.: Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on* 13(1), 21–27 (1967)
- [6] Ditzler, G., Muhlbaier, M.D., Polikar, R.: Incremental learning of new classes in unbalanced datasets: Learn++. udnc. In: *Multiple classifier systems*, pp. 33–42. Springer (2010)
- [7] Fukunaga, K.: Introduction to Statistical Pattern Recognition. Academic Press Professional, Inc., San Diego, CA, USA (1990)
- [8] Hua, X., Ding, S.: Incremental learning algorithm for support vector data description. *Journal of Software* 6(7), 1166–1173 (2011)
- [9] Kardaun, O.J.: Classical methods of statistics: with applications in fusion-oriented plasma physics, vol. 1. Springer Science & Business Media (2005)
- [10] Liu, L., Shao, L.: Learning discriminative representations from rgb-d video data. In: *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*. pp. 1493–1500. AAAI Press (2013)
- [11] Lütz, A., Rodner, E., Denzler, J.: I want to know more efficient multi-class incremental learning using gaussian processes. *Pattern recognition and image analysis* 23(3), 402–407 (2013)
- [12] Mensink, T., Verbeek, J., Perronnin, F., Csurka, G.: Large scale metric learning for distance-based image classification on open ended data sets. In: *Advanced Topics in Computer Vision*, pp. 243–276. Springer (2013)
- [13] Müller, M.: Dynamic time warping. *Information retrieval for music and motion* pp. 69–84 (2007)
- [14] Polikar, R., Upda, L., Upda, S.S., Honavar, V.: Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 31(4), 497–508 (2001)
- [15] Ruffieux, S., Lalanne, D., Mugellini, E., Khaled, O.A.: Gesture recognition corpora and tools: A scripted ground truthing method. *Computer Vision and Image Understanding* 131, 72–87 (2015)

- [16] Sakic, D.: Semiüberwachtes Lernen mit Ensemble-Methoden zur Erkennung von Gesten. Diplom thesis, Faculty of Computer Science, TU Dortmund University (2012)
- [17] Schneider, P., Biehl, M., Hammer, B.: Adaptive relevance matrices in learning vector quantization. *Neural Computation* 21(12), 3532–3561 (2009)
- [18] Schumacher, J., Sakič, D., Grumpe, A., Fink, G.A., Wöhler, C.: Active learning of ensemble classifiers for gesture recognition. Springer (2012)
- [19] Schürmann, J.: Pattern classification: a unified view of statistical and neural approaches. Wiley Online Library (1996)
- [20] Seki, S., Takahashi, K., Oka, R.: Gesture recognition from motion images by spotting algorithm. In: Proc. ACCV. vol. 2, pp. 759–762 (1993)
- [21] Tavakkoli, A., Nicolescu, M., Nicolescu, M., Bebis, G.: Incremental SVDD training: Improving efficiency of background modeling in videos. In: Proceedings of the 10th IASTED International Conference. vol. 623, p. 092 (2008)
- [22] Tax, D.M.J., Duin, R.P.W.: Support vector data description. *Machine learning* 54(1), 45–66 (2004)
- [23] Tax, D.M., Laskov, P.: Online SVM learning: from classification to data description and back. In: Neural Networks for Signal Processing, 2003. NNSP'03. 2003 IEEE 13th Workshop on. pp. 499–508. IEEE (2003)
- [24] Tax, D.: One-class classification: concept-learning in the absence of counter-examples. Ph.D. thesis, TU Delft, Delft University of Technology (2001)
- [25] Tax, D.: Ddtools, the data description toolbox for matlab (June 2015), version 2.1.2
- [26] Webb, A.R.: Statistical pattern recognition. John Wiley & Sons (2003)
- [27] Williams, D.J., Shah, M.: A fast algorithm for active contours and curvature estimation. *CVGIP: Image understanding* 55(1), 14–26 (1992)
- [28] Zhang, B.F., Su, J.S., Xu, X.: A class-incremental learning method for multi-class support vector machines in text classification. In: Machine Learning and Cybernetics, 2006 International Conference on. pp. 2581–2585. IEEE (2006)
- [29] Zhu, X., Goldberg, A.B.: Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning* 3(1), 1–130 (2009)

Attention as cognitive, holistic control of the visual system

Frederik Beuth, Fred H. Hamker

Technische Universität Chemnitz, Artificial Intelligence,
Strasse der Nationen 62, 09111 Chemnitz, Germany
{beuth, fhamker}@cs.tu-chemnitz.de
<http://www.tu-chemnitz.de/informatik/KI/>

Abstract Visual attention is classically seen as a selection process, picking the relevant information from the vast amount of sensory data. However, recent neuroscience studies suggest a more general role of attention as a cognitive control process that tunes the visual system for the task at hand. The process modulates the whole visual system, thus we denote its influence as holistic. This holistic control concept is not included in current computer vision systems, although it seems beneficial. We demonstrate the concept at the example of an object localization task, in which a given target has to be searched in a scene (guided visual search). State-of-the-art computer vision systems solve this task via the classical view of attention (saliency models): attention constitutes a spatial pre-selection stage for a subsequent recognition stage. Yet, this approach has the problem that selection and recognition operate in separate stages although the two processes depend on each other. The issue is solved by the holistic attention approach as it controls selection and recognition in parallel. We benchmark our model on a realistic object localization setup, using 100 different target objects, 1000 scenes, and three backgrounds. The model achieves a localization accuracy of 92% at black backgrounds. Generalization to white-noise or real-world backgrounds changes the accuracy to 71% and 42% respectively. Our results demonstrate that attention as cognitive, holistic control is able to solve realistic computer vision problems.

Keywords: Visual attention; Object localization; Guided visual search; Cognitive, holistic control; Top-down saliency model

1 Introduction

Visual attention is classically seen as a selection process to pick the relevant information among the vast amount of incoming sensory data [1]. This idea has led to the development of saliency models in the domain of object localization [2]. Object localization is here understood as the task to search a given target object in a scene (guided visual search [3]). Saliency models use attention to pre-select some regions of interest (ROIs) from the image and pass them to a subsequent, sophisticated recognition module. Top-down saliency models select

the ROIs via top-down attention to target features. Firstly, they extract from the image a number of low-level feature maps and integrate them, weighted with the top-down attention signal, to a saliency map. Afterwards, ROIs are selected around the highest activity in the saliency map. The top-down weights encode if a particular feature is part of the target object, thus the resulting saliency map contains high activity at the location of potential target objects.

However, the separation between selection and recognition does not exist in the primate brain, and attention has also not the role of a solely pre-selection stage [4]. To solve these issues, we will review recent neuroscience studies and will illustrate that attentional processing spans a top-down control network, modulating neuronal activity for the current task. The network targets the whole visual system, thus we see attention as a cognitive, holistic control process.

The attentional processing of the primate brain is already simulated by many neuro-computational models, yet only a few have been applied to realistic object localization tasks due to their ability to handle whole objects and real-world scenes [5,6,7,8,9]. We improve these studies in two points: Firstly, they have been evaluated only in relatively easy setups with a small number of object categories (2 - 16), thus we evaluate our model in realistic and large setup with 100 categories. Secondly, we address more precisely and deeply the role of attention, leading to our novel proposal of attention as holistic, cognitive control.

2 Model

We will first review the attentional processing network in the primate brain and propose our concept of attention as cognitive, holistic control. Afterwards, we will present our attention model and the implementation of the holistic control.

2.1 Attention as holistic, cognitive control of the visual system

We see attention as a cognitive and holistic control of the visual system, which tunes the whole visual processing for the task at hand. Before explaining this view, we recapitulate the primate visual system [10,13]. It consists of a feed-forward and feedback processing network (Fig. 1a), from which we model the relevant areas for object localization: ventral stream (V1, V4, IT), FEF, and PFC (Fig. 1b). The ventral stream [13] consists of the primary visual cortex (V1) encoding very local and simple features like edges, disparities, motion, or color contrasts; of the fourth visual cortex (V4) encoding complex shapes; and of the inferior temporal cortex (IT) encoding objects. The frontal eye field (FEF) encodes spatial [14] and the prefrontal cortex (PFC) task information [15].

Our novel view of attention results from the following findings. Miller & Buschman [11] illustrate that attention is transmitted via the feedback network originating from the PFC. As that area encodes task information [15], we reason that the network influences neuronal activity for the current task. The network targets the whole visual cortex, so we denote its influence as holistic.

At physiology level, attention has several effects on the neuronal responses [16]. Most prominently, it amplifies the response of neurons encoding the

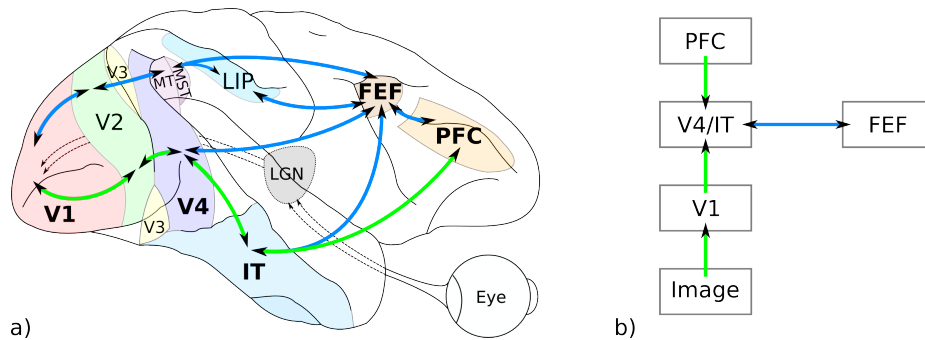


Figure 1. a) Schematic sketch of the visual attention system in primates. Adapted from [10], whereby LIP and PFC are added from [11]. The arrows denote the ventral (green) and dorsal streams (blue) of the visual cortex plus their connections to frontal cortices (FEF and PFC). Their primary functions are to process the type (green, ventral) and location of an object (blue, dorsal; [12]). Bottom-up processing is denoted by arrows from left to right, top-down processing by arrows from right to left. The top-down connections mediate attention signals [11]. b) Areas and connections simulated in the attention model. They are printed in bold in a). These relevant areas are explained in the main text, and the remaining in [10,11].

attended stimulus, and suppresses the response of neurons encoding unattended stimuli. The attended stimulus is per definition task-relevant, so attention modulates neuronal activity for the current task by increasing the response to task relevant stimuli and by suppressing irrelevant ones. We combine these findings together and argue that attention is a cognitive, holistic control, modulating neuronal activity for the current task.

2.2 Model overview

We simulate the cortical attention network in a novel system-level model of attention. The model has been developed for the upcoming doctoral thesis of the first author. An overview about the model is given in this section, whereby anatomical and physiological background can be found in the thesis, and its mathematical description in the following supplementary material: www.tu-chemnitz.de/cs/KI/supplement/BeuthHamker2015b/.

The system-level model of attention (Fig. 2) consists of the visual cortices V1 and HVA, the frontal eye field (FEF), and the prefrontal cortex (PFC). Firstly, the image is processed by a primary visual cortex model (V1), encoding oriented edges (O), red-green (L - M), and blue-yellow color contrasts (S - LM). The next stage HVA (Higher Visual Area) contains view-tuned cells representing a single view of an object. This stage is an abstraction of the cortical areas V4 and IT. HVA is simulated by a recently-developed microcircuit model of attention [16]. The model provides physiologically accurate modulation of the neuronal responses by attention as it replicates a vast amount of physiological data sets. Layer 4 of the circuit recognizes object views and layer 2/3 pools spatially over

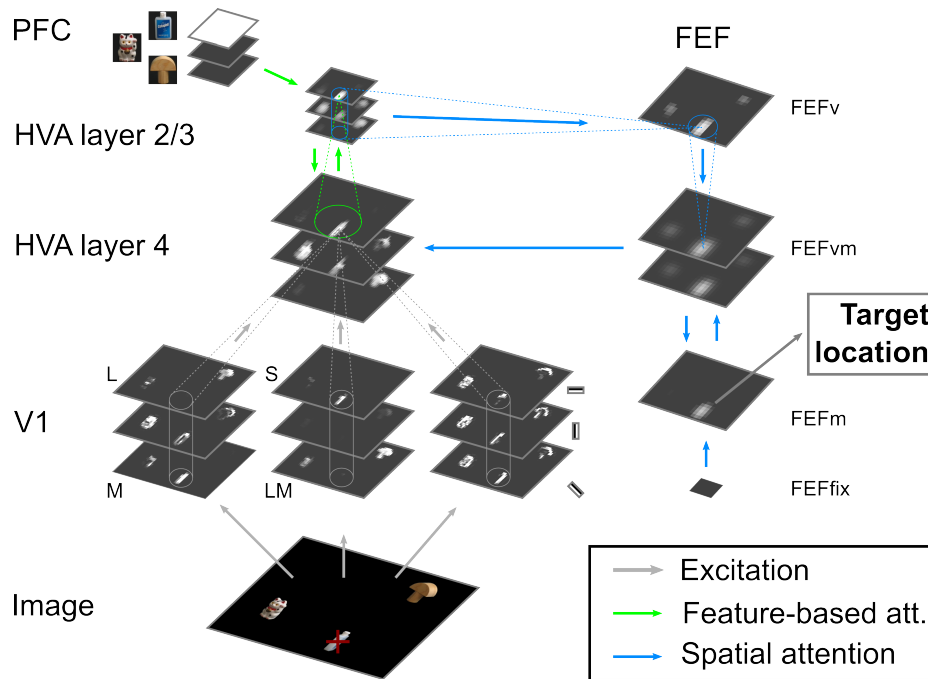


Figure 2. The novel system-level model of visual attention. The processing is illustrated at the task to localize and recognize the “bottle”, indicated by the red cross. See main text for details.

a particular region of layer 4 activities to increase spatial invariance. Each view of an object is encoded by the connection weights between V1 and HVA layer 4, so each neuron in layer 4 reacts to a specific pattern of V1 neurons. The weights are determined in an off-line training phase using trace learning [5] as explained in the supplementary material.

The object localization task implies feature-based attention towards the target object, implemented by activating a particular object neuron in the prefrontal cortex (PFC). The object neurons in PFC are connected via learned weights to view-tuned cells in HVA layer 2/3 and further to layer 4. These connections amplify all view-tuned cells encoding the target, which leads to an increased neuronal activity of all target related neurons in both HVA layers.

The frontal eye field (FEF) processes spatial information and selects the target location. It is based on the model of Zirnsak et al. [14] and is divided, according to the physiological cell types, into three layers: FEFv, FEFvm, and FEFm. Functionally, FEFv encodes possible locations of the target (white blobs in Fig. 2), whereas FEFvm and FEFm indicate the final target location. FEFvm codes the target location during the normal attentional processing, whereas FEFm represents an upcoming eye movement (saccade) towards the target. The FEFv is computed by taking the maximum activity over all the features in HVA. The

signal from FEFv to FEFvm implements a soft-competition between locations by applying a Gaussian filter to reinforce adjacent ones and by long-range inhibition to suppress remote ones. The FEFvm projects back to HVA layer 4, forming a recurrent processing loop: HVA layer 4→HVA layer 2/3→FEFv→FEFvm→HVA layer 4. The competition is continuously executed within the loop, leading to the selection of a single target location. The FEFm uses a similar competition to generate a saccade target. The FEFm competition concentrates neuronal activity to a single area over time. If this activity reaches a threshold, a saccade is triggered towards this target location which designates the final result of the localization process.

2.3 Implementation of the holistic, cognitive control

Our model implements the holistic control network via top-down attention signals from PFC to the visual cortex, i.e HVA, and further to the FEF. The connections from PFC to HVA layer 2/3 and downwards to layer 4 implement feature-based attention. It controls the visual system based on the encoded feature of a cell, whereby a feature stands for an object view in HVA and an object type in PFC. The task of object localization specifies the target object via its type. We simulate this task instruction by activating the appropriate target object cell in PFC. Afterwards, attention controls the visual system based on this cell's activity by top-down feature-based amplification: it amplifies all HVA neurons encoding a target view. This mechanism is combined with feature-based suppression, implemented via inhibitory connections within HVA, to decrease the response of HVA neurons encoding other views. This suppresses neuronal noise, originating from distractors or background clutter.

Spatial attention controls the system based on spatial information. It is implemented via the recurrent processing loop between HVA and FEF. Spatial attention has the effect to amplify the response of all HVA neurons at the target location (spatial amplification) and to suppress HVA neurons at all other locations (spatial suppression). The spatial processing relies on the HVA activity that was modulated according to the given feature of the task. It so realizes the task instruction as it determines the target location for the given feature. Spatial attention has the functions to focus neuronal activity on the target location, to inhibit the location of distractors or background clutter, and to segment the target from them. Besides these functions, spatial attention synchronizes the spatial information in FEF and the feature information in HVA [8,10].

Therefore, attention controls the system for the current task demands via the four attention mechanisms. The control process operates in all areas in parallel to realize the holistic property of the concept.

3 Results

We will first demonstrate that the approach is able to perform realistic object localization tasks, and afterwards shed light on its advantages over top-down saliency models.

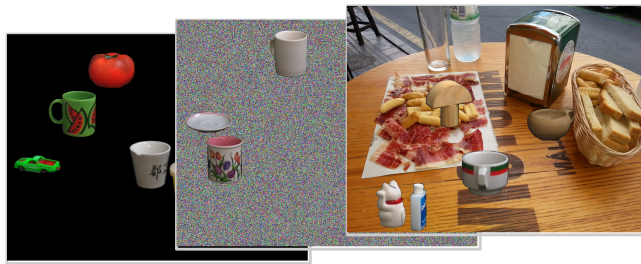


Figure 3. Exemplary test scenes with black, white-noise, and real-world backgrounds (from left to right).

3.1 Performance of the model

We evaluated the model on three large and realistic object recognition test sets, consisting of 1000 different scenes with either a) black, b) colored white-noise, or c) real-world backgrounds (Fig. 3). The sets can be downloaded via: <http://ai.informatik.tu-chemnitz.de/projects/ObjectRecAttention/>. A separate set with black backgrounds was used to train the model. Each scene contains five different objects from a set of 100 objects under 72 different rotations (COIL-100 data set [17]). The model’s task was to search for one of these five objects and to report its location. To evaluate the model, we measure how often this location was reported correctly (localization accuracy). A location was counted as correctly if the reported position was within the object borders or not more than 50 pixels Euclidean distance away from them. The value of 50 pixels corresponds roughly to the half of an object. The model achieves on the black background test set a localization accuracy of 92%. This set contains similar backgrounds as used for the training of the model. If the model has to generalize to white-noise or real-world backgrounds, the accuracy changes to 71% and 42% respectively.

Mislocalizations occur in the black background set mainly when a distractor is similar to the target and more salient than the target. These conditions are the typical problematic ones [2], even for humans [3]. They occur more often in the white-noise set as the noise reduces the neuronal representation of the target. Additionally in this set, mislocalizations take place if an object is similar to the background noise. In the real-world backgrounds, mislocalizations occur mostly if the background is similar and more salient than the target.

An existing study reported a similar performance with a saliency model. Elazary & Itti [18] benchmarked a probabilistic top-down saliency model on scenes composed from the same objects before black background. They reported a localization accuracy of 97%. The slightly higher accuracy might be attributed to the fact that their model uses supervised learning, whereby our trace learning is unsupervised. We use the trace learning approach for biological plausibility and are aware that supervised learning approaches might achieve a higher performance as they additionally exploit the object class information.

In summary, our model achieves a similar performance as a state-of-the-art approach, demonstrating that the holistic attention approach is able to solve realistic object localization tasks.

3.2 Advantages compared to top-down saliency models

The approach of attention as holistic, cognitive control has the advantage that object recognition and attention are closely intertwined, a goal already suggested by Frintrop et al. [4]. Attention operates no longer as a solely spatial pre-selection stage; instead, it controls in parallel recognition as well as spatial selection. The parallel processing solves the interdependency problem of object recognition and selection, which is unsolved in the saliency model approach. Selection needs knowledge about the properties of the target object and so depends on a successful recognition. However, recognition requires beforehand selection to segment the object from the background [5]. Spatial selection is achieved in our model via localization in the frontal eye field.

As another advantage, the holistic approach allows to select an object via high-level features instead of low-level features as in the top-down saliency models. These models deploy attention directly to the low-level features, which allows a selection only via them. This concept is only suitable in some tasks, for example Mitri et al. deploy attention to the feature ‘red’ to recognize a red ball in a soccer scenario [19]. They use the low-level feature ‘red’ as the ball color was uniquely in their scenario. However in many tasks, the target objects cannot be selected uniquely by low-level features [20]. Therefore, attention has to be deployed to more complex visual representations. These range from mid-level representation, view-tuned cells (e.g. HVA), to object-category representations (e.g. PFC) in the holistic models (ours, [5,6,7,8,9]). Selection is then based on these high-level representations. The high-level representations are also utilized for the recognition and thus are shared between recognition and selection. Walther & Koch [9] denote this concept as feature-sharing.

Summed up, the cognitive, holistic approach has the advantages of parallel recognition and selection, and of a selection via high-level feature descriptors.

4 Conclusion

In this work, we propose the view of attention as a cognitive, holistic control process. The process is mediated by a top-down network targeting the whole visual cortex. The network modulates neuronal activity for the current task and tunes so the visual system to the task demands. We simulate the network in a novel system-level model of attention and realize the neuronal modulation with a physiological-grounded microcircuit model of attention.

We demonstrate the model on a realistic object localization task and illustrate its advantages over the state-of-the-art computer vision approach of top-down saliency models. Our model achieves an accuracy of 92% at black backgrounds. Generalization to white-noise or real-world backgrounds changes the accuracy to 71% and 42%. These results demonstrate the successfulness of the holistic approach for a realistic computer vision task. Therefore, the approach might be an alternative to top-down saliency models, especially in conditions where those might fail. Such conditions enclose tasks in which the recognition and selection processes depend on each other, and tasks in which the target cannot be distinguished via simple feature from the distractors or the background.

Acknowledgments. This work has been supported by the European Project “Spatial Cognition” (No. 600785), and partly by the Research Training Group “CrossWorlds” (No. GRK1780) founded by the German Research Foundation.

References

1. Carrasco, M.: Visual attention: the past 25 years. *Vision Res* 51, 1484–525 (2011)
2. Borji, A., Itti, L.: State-of-the-art in visual attention modeling. *IEEE Trans Pattern Anal Mach Intell* 35(1), 185–207 (2013)
3. Wolfe, J.M.: Guided search 2.0 a revised model of visual search. *Psychon Bull Rev* 1(2), 202–238 (1994)
4. Frintrop, S., Rome, E., Christensen, H.: Computational visual attention systems and their cognitive foundations: A survey. *ACM TAP* 7(1), 1–39 (2010)
5. Antonelli, M., Gibaldi, A., Beuth, F., Duran, A.J., Canessa, A., Chessa, M., Hamker, F.H., Chinellato, E., Sabatini, S.P.: A hierarchical system for a distributed representation of the peripersonal space of a humanoid robot. *IEEE Trans Auton Mental Develop* 6(4), 259–273 (2014)
6. Beuth, F., Wiltshut, J., Hamker, F.H.: Attentive Stereoscopic Object Recognition. In: Villmann, T., Schleif, F.M. (eds.) *Proc Workshop New Challenges in Neural Computation 2010 - NCNC 2010, Machine Learning reports 04/2010, AG Computational Intelligence, University of Leipzig*, pp. 41–48 (2010)
7. Chikkerur, S., Serre, T., Tan, C., Poggio, T.: What and where: a Bayesian inference theory of attention. *Vision Res* 50(22), 2233–47 (2010)
8. Hamker, F.H.: The emergence of attention by population-based inference and its role in distributed processing and cognitive control of vision. *Comput Vis Image Underst* 100, 64–106 (2005)
9. Walther, D.B., Koch, C.: Attention in hierarchical models of object recognition. *Prog Brain Res* 165, 57–78 (2007)
10. Jamalian, A., Hamker, F.H.: Biologically-Inspired Models for Attentive Robot Vision: A Review. In: Pal, R. (ed.) *Innovative Research in Attention Modeling and Computer Vision Applications - In press*. IGI Global (2015)
11. Miller, E.K., Buschman, T.J.: Cortical circuits for the control of attention. *Curr Opin Neurobiol* 23(2), 216–222 (2013)
12. Ungerleider, L., Haxby, J.: ‘What’ and ‘where’ in the human brain. *Curr Opin Neurobiol* 4, 157–165 (1994)
13. Serre, T.: Learning a dictionary of shape-components in visual cortex: Comparison with neurons, humans and machines. Ph.D. thesis (2006)
14. Zirnsak, M., Beuth, F., Hamker, F.H.: Split of spatial attention as predicted by a systems-level model of visual attention. *Eur J Neurosci* 33(11), 2035–45 (2011)
15. Sakai, K.: Task set and prefrontal cortex. *Annu Rev Neurosci* 31, 219–45 (2008)
16. Beuth, F., Hamker, F.H.: A mechanistic cortical microcircuit of attention for amplification, normalization and suppression. *Vision Res* (2015), in Press
17. Nene, S.A., Nayar, S.K., Murase, H.: Columbia Object Image Library (COIL-100). Technical Report CUCS-006-96 (1996)
18. Elazary, L., Itti, L.: A Bayesian model for efficient visual search and recognition. *Vision Res* 50(14), 1338–1352 (2010)
19. Mitri, S., Frintrop, S., Pervözl, K., Surmann, H.: Robust object detection at regions of interest with an application in ball recognition. In: *Proc IEEE Int Conf Robotics and Automation 2005 - ICRA 2005*. pp. 126–131 (2005)
20. Xu, T., Chenkov, N.: Autonomous switching of top-down and bottom-up attention selection for vision guided mobile robots. In: *Proc IEEE/RSJ Conf Intelligent Robots and Systems 2009 - IROS2009*. pp. 4009–4014 (2009)

Learning Conditional Mappings between Population-Coded Modalities

Fabian Schrodt and Martin V. Butz

Cognitive Modeling, Department of Computer Science,
University of Tübingen, Germany
{tobias-fabian.schrodt,martin.butz}@uni-tuebingen.de

Abstract. It is still an open question how the brain manages to map various modalities onto each other. We introduce a tripartite neural network architecture that is able to learn non-linear mappings between topological, population-encoded modalities. The neural network gains this capability by creating sparse modal correlation maps. By applying factorization, the correlation maps serve as mutually conditional transformations onto a third modality. We show that such a combination is able to solve the locally linear task of learning forward velocity kinematics of a simple arm. In comparison to other approaches, the architecture is robust and predictable in terms of learning performance and efficient in terms of model complexity. The model is neurally plausible, mimicking coordinate transformations known to be computed in parietal cortex, and may serve as a basic building block to model non-linear mappings between population-encoded modalities, which are typically grounded in different frames of reference.

Keywords: Gain-Field Encodings, Population Coding, Factorization, Sparse Coding, Non-Linear Mappings, Neural Networks

1 Introduction

The brain is able to combine information of various modalities and frames of reference to learn predictive models about its body and its interaction with the environment. Typically, sensory input can be assumed to be encoded by populations of locally receptive cells with tunings to specific stimulus characteristics [1]. To eventually establish a body model, conditionals of multiple population-coded stimuli have to be learned to resolve non-linearities and minimize surprise about action outcomes. In parietal cortex, mappings between population-encoded modalities have been suggested to be established by means of gain-fields [2, 3].

One example of a tripartite mapping problem is learning a combined model of kinematic and dynamic bodily control: Given the proprioception of a limb, what effect does a change in its posture (or a motor command) have in terms of visually perceived motion? In this question, two modal sources of information have to be mapped onto a third one, resulting in a locally linear, conditional mapping

problem. A neurocognitively plausible model may learn such a forward model by means of supervised training because the outcomes of actions are directly observable [4]. Once a forward model has been learned, it is possible to derive an inverse mapping, which is typically ambiguous due to redundancies, that is, extra degrees of freedom, in the system. Population codes have the potential to resolve resulting uncertainties [5]. For many degrees of freedom, however, classical gain-field models are not applicable due to the resulting huge network sizes, diminishing their cognitive plausibility when no modularization is applied [6]. We asked the question how such mappings can be learned considering the conditional, tripartite nature of this problem (and others) while keeping the computational effort minimal.

In the following, we describe the Conditional Mapping Architecture (CMA) and show that it is able to learn a combined model of forward dynamics and kinematic control. The architecture relies on pair-wise multiplications in modal modules instead of tensor products and is thus easily differentiable and computationally more efficient than related approaches inspired by gain-field encodings. In subsequent experiments, we show that the model features fast and predictable learning performance that is adjustable via a single topological parameter.

2 Conditional Mapping Architecture

CMA is a tripartite architecture sketched in Fig. 1. Two of the parts serve as input to the network, each fed by data of a different modality. The third part is defined as the output modality. We assume that there is a non-linear, but distinct dependency of the output modality on the inputs. That said, the output is determined by a many-to-one mapping from the first input modality to the output modality, under the condition of the second input modality – or vice versa.

2.1 Population Coding

Each modal information is assumed to be limited to a hyperrectangular range in \mathbb{R}^{D^m} , $m \in \{1, 2, 3\}$ ¹. Each component or dimension $d^m \in \{1..D^m\}$ in a modal space is represented separately by a population of N neurons², providing regularly distributed tuning prototypes in the range of this component. Then, given a modal input sensation $s^{m,d^m} \in [s_{\max}^{m,d^m}, s_{\min}^{m,d^m}]$, $m \in \{1, 2\}$, each neuron $i \in \{1..N\}$ in an input population responds in the form of a unit Gaussian radial basis function defined by:

$$a_i^{m,d^m} = \exp \left(-1/2 \left(\frac{p_i^{m,d^m} - s^{m,d^m}}{b^{m,d^m}} \right)^2 \right), \quad m \in \{1, 2\}, \quad (1)$$

¹ Superscripts of scalars or vectors denote descriptors, while subscripts denote indices (with the exception of ‘min’ and ‘max’).

² We assume that each modal component is represented by the same number of neurons for reasons of simplicity.

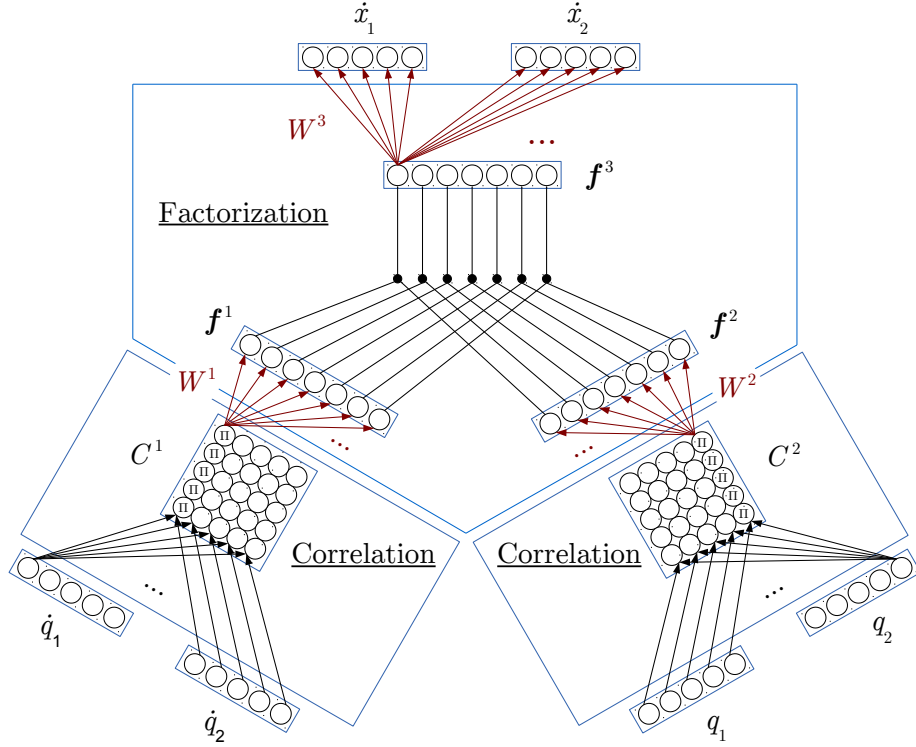


Fig. 1. Connection scheme of the tripartite mapping architecture applied to learning a forward model of endeffector motion $\dot{\mathbf{x}}$, given angles \mathbf{q} and angular velocities $\dot{\mathbf{q}}$. Free parameters are shown in red.

where p_i^{m,d^m} is the prototype in the range of the modal component and b^{m,d^m} is the breadth of the tuning of neurons in the population representing information in dimension d^m of modality m . The breadth depends on an a-priori range of the modal component:

$$b^{m,d^m} = \frac{s_{\max}^{m,d^m} - s_{\min}^{m,d^m}}{2N}, \quad m \in \{1, 2, 3\}. \quad (2)$$

For output populations ($m = 3$ in the following), each neuron $i \in \{1..N\}$ responds linearly to upstreamed signals, resulting in activities a_i^{m,d^m} (see Eq. 7). Given a target sensation or observation $s^{m,d^m} \in [s_{\max}^{m,d^m}, s_{\min}^{m,d^m}]$, an error term δ_i^{m,d^m} can be calculated suitable for backpropagation:

$$\delta_i^{m,d^m} = t_i^{m,d^m} - a_i^{m,d^m}, \quad m = 3 \quad (3)$$

where target activities t_i^{m,d^m} are determined analogously to Equation 1.

2.2 Correlating Modalities

To build a processing architecture that is able to learn conditional, tripartite mappings, we at first employ a *correlation map* between all dimensions of each input modality. This correlation map is realized by all *pair-wise products* of single activations of cells in unequal populations, resulting in $N^2 \cdot D^m \cdot (D^m - 1)/2$ multiplications. Thus, the activation $c_{i,j}^{m,d^m,e^m}$ of a multiplicative unit correlating activations i and j of dimension d^m and e^m of modality m is given by

$$c_{i,j}^{m,d^m,e^m} = a_i^{m,d^m} \cdot a_j^{m,e^m}, \quad (4)$$

where $d^m \neq e^m \in \{1..D^m\}$, $i, j \in \{1..N\}$. This is similar but not equivalent to tensor products often used in gain-field models, because our approach does not require D^m -fold multiplications, which reduces the number of parameters drastically. For $D^m = 2$, this approach is equivalent to the outer product of population activity vectors. Note that only the two input modalities have to be correlated independently.

2.3 Factorization

The resulting correlation maps in our architecture can each be considered a 2D image C^m with width $N \cdot D^m \cdot (D^m - 1)/2$ and height N , reflecting all possible, pair-wise logical AND interactions. By nature, image processing methods are applicable – the aim is to find a suitable and yet sparse connectivity that is able to realize image transformations systematically, whereas one modality can be considered the input image, and the other can be considered a *warp*. Allowing all possible mappings from the two correlation maps to the output modality would result in a three-way interaction tensor with cubic number of parameters. However, it has been shown that *factorization* can reduce the number of parameters drastically, given that local regularities in the mapping exist [7].

Factorization is realized by learning a triplet of linear transformations W^m : Two of them factorize given input images each onto a modal factor vector $\mathbf{f}^m \in \mathbb{R}^F$, and the third transforms the *elementwise multiplication* (denoted \odot) of the modal factors to the output image. The factors are represented by linear hidden units. This yields the factor cell activations

$$\mathbf{f}^m = W^m \cdot C^m, m \in \{1, 2\} \quad (5)$$

$$\mathbf{f}^3 = \mathbf{f}^1 \odot \mathbf{f}^2, m = 3. \quad (6)$$

The transformation matrices W^m are the only free parameters of the architecture, resulting in $F \cdot N \cdot (N \cdot D^1 \cdot (D^1 - 1)/2 + N \cdot D^2 \cdot (D^2 - 1)/2 + D^3)$ parameters overall. Finally, \mathbf{f}^3 is transformed to the concatenated output activations, effectively including a projection of the combined correlation maps:

$$(\mathbf{a}^{m,1}, \dots, \mathbf{a}^{m,D^m}) = W^m \cdot \mathbf{f}^m, m = 3. \quad (7)$$

Training the architecture is possible by means of error gradient descent by back-propagation, since all parameters are differentiable. In the next section, we evaluate the architecture.

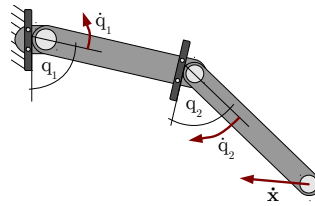


Fig. 2. Simulation of a 2-DOF arm with angular constraints.

3 Evaluation Setup

We evaluate the architecture on the task of approximating the forward velocity kinematics of a simple 2 DOF arm with two limbs with unit length and an endeffector in a 2D positional space, resulting in $D^m = 2 \forall m$. A sketch of the arm is shown in Fig. 2. The learning objective is the prediction of endeffector velocities $\dot{\mathbf{x}} \stackrel{!}{=} \mathbf{s}^3$, given angular velocities $\dot{\mathbf{q}} = \mathbf{s}^1$ under the condition of angular postures $\mathbf{q} = \mathbf{s}^2$. Given the condition, this mapping is onto and locally linear.

All modal components were represented each by $N = 10$ population neurons. For evaluation, three Conditional Mapping Architectures with different number of parameters were compared to three Multilayer Perceptrons (MLP) with three hidden layers (HL) consisting of neurons with hyperbolic tangent activation functions and biases, resulting in approximately the same number of free parameters (FP):

- **CMA-16**: $F = 16$ factor units \Rightarrow 3520 FP. Learning rate 0.001.
- **MLP-30**: 30 neurons per HL \Rightarrow 3690 FP. Learning rate 0.001.
- **CMA-49**: $F = 49$ factor units \Rightarrow 10780 FP. Learning rate 0.004.
- **MLP-60**: 60 neurons per HL \Rightarrow 10980 FP. Learning rate 0.002.
- **CMA-99**: $F = 99$ factor units \Rightarrow 21780 FP. Learning rate 0.008.
- **MLP-90**: 90 neurons per HL \Rightarrow 21870 FP. Learning rate 0.003.

Learning rates were optimized heuristically and separately for each network, all momenta were set to 0.8. Small learning rates are required for learning this prediction online to avoid catastrophic inference and forgetting of already learned, locally linear mappings [8]. Fig. 1 shows a CMA architecture example for $N = 4$ and $F = 7$ applied to the above learning task. A classical computational gain-field as originally described by Zipser and Andersen [9] would have 200k free parameters in this configuration. It would however not be possible to train it for higher-dimensional problems.

For sampling the training data, we set goal postures \mathbf{q}^g randomly distributed in posture space and moved the arm according to the normalized postural distance

$$\dot{\mathbf{q}}(t) = \frac{\mathbf{q}^g(t) - \mathbf{q}(t)}{\|\mathbf{q}^g(t) - \mathbf{q}(t)\|} \cdot \mathbb{U}^2(0.1) \quad (8)$$

where $\mathbb{U}^2(0.1)$ is a 2-dimensional, uniformly distributed random variable in the interval $[0, 0.1]$ rad. By adding noise to each angular movement, we could enforce

local variance both in the direction and velocity of the movement of the end-effector. By targeting goal postures randomly, we could ensure that the whole postural space was uniformly sampled throughout the training procedure. As soon as the Euclidean distance $\|\mathbf{q}^g(t) - \mathbf{q}(t)\|$ between the current posture vector and the goal posture vector fell below 0.2, a new goal posture was set. Goal postures were restricted by the angular constraint $q_i \in [0, \pi]$. The resulting modal ranges for angular and endeffector velocities were set accordingly.

We trained each network for 10^7 time steps in 6 independent trials, including normally distributed parameter initialization with mean 0 and variance 0.1 with different random seeds. After training, we averaged the performance in a 50k time steps evaluation phase without parameter adaptation to test for local overadaptation to the target mapping. The results are shown and interpreted in the following.

4 Results

The output components \mathbf{a}^{3,d^3} provided by a network were used to decode the predicted directional endeffector velocity $\dot{\mathbf{x}}$ by polling their prototypes and weighting them according to their activation for each modal output component d^m , yielding:

$$\dot{x}^{d^m} = \frac{\sum_{i=1}^N x_i^{m,d^m} \cdot a_i^{m,d^m}}{\sum_{i=1}^N a_i^{m,d^m}}, \quad d^m \in \{1..D^m\}, \quad m = 3 \quad (9)$$

Then, the prediction $\dot{\mathbf{x}}$ was compared to the actual observation \mathbf{s}^3 . To evaluate the networks' performances, we derived two measuring units from this comparison: First, we compared the *angular error* in the direction of the predicted endeffector motion, defined by

$$\Delta\alpha = \arccos\left(\frac{\dot{\mathbf{x}} \bullet \mathbf{s}^3}{\|\dot{\mathbf{x}}\| \cdot \|\mathbf{s}^3\|}\right) \cdot \frac{180}{\pi} \quad (10)$$

where \bullet is the scalar product. Secondly, we defined an exponential *velocity error*

$$\Delta v = \left| \log\left(\frac{\|\dot{\mathbf{x}}\|}{\|\mathbf{s}^3\|}\right) \right| \quad (11)$$

The development of these two error measures over time in comparison of the different network types is shown in Fig. 3(a) and Fig. 3(b).

At first, it has to be noted that finding a working MLP configuration for the task at hand was not trivial. Since it can be considered a highly non-linear classification problem, at least three hidden layers (resulting in 5 consecutive linear transformations) were required for decent performance. We achieved our best results using hidden layers of equal size. All MLPs had a rather slow learning progress early on: Training began to have a noticeable effect only after about 80k time steps. At this time, CMAs already reached an acceptable performance

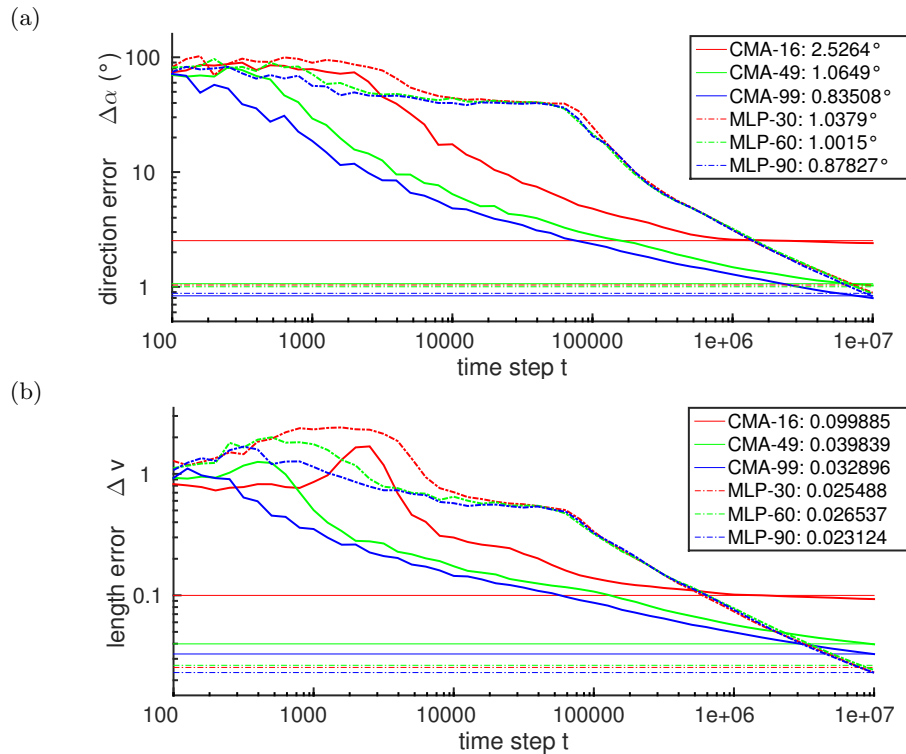


Fig. 3. Log-log plots of the deviation of (a) the predicted endeffector movement direction from the actual movement direction in degree and (b) the predicted endeffector movement speed from the actual movement direction in exponential relation. Horizontal lines represent the average error levels while testing for local overfitting, which is also indicated in the legends, respectively.

e.g. with angular errors between 2.5 and 5. However, upon reaching a specific leverage point, the errors decreased about dual-logarithmically. When increasing the learning rate, MLPs tended to strong overadaptation to the locally linear forward dynamics, resulting in worse performance after all. Interestingly, as the results show, increasing the number of hidden neurons did not improve the convergence rate nor final performance significantly. MLPs with larger hidden layers however seemed slightly less prone to overfitting.

In comparison, CMAs can especially be characterized by superior rate of learning progress during the early training period. This suggests that the solution spaces of CMAs are more convex than for MLPs, such that they were able to outperform MLPs at least for a finite time span. Also, CMAs did not tend to overfit that much when increasing the learning rate. Instead of using a hierarchy of non-linear classifiers, CMA relies on pair-wise multiplications of (linear) factors only, making them easy and robust to train. In contrast to MLP, CMA's performance increases with F as the only topological parameter of the architecture. Thus, using CMAs seems suitable to quickly find an approximate solution

without challenging topological parameters. Considering the results, though, we cannot draw the conclusion that CMAs were able to outperform MLPs in terms of the attainable error limit.

5 Conclusions

In the course of this ongoing investigation, we introduced an architecture that is able to learn online an approximation of population-coded forward dynamics in about 80k time steps with accuracy. Transforming pair-wise correlation maps of two modal input populations is sufficient to solve this problem. In contrast to conservative gain-fields, our approach is potentially able to handle higher-dimensional problems (e.g. 7 DOF forward dynamics) with a reasonable number of parameters. The applied factorization can also be considered a fusion of a pair of two-dimensional gain-field mappings onto a third modality. Just as hierarchical gain-fields can be used to reduce the number of parameters [6], applying hierarchical factorizations of correlation maps could thus result in further reduction of the number of parameters, particularly when applied to higher-dimensional problems. Specifically relating to the task of learning forward dynamics, adding recurrences from angular changes to angles might provide the network with further capabilities. In future work, also opportunities to resolve the inverse dynamics could be investigated based on this architecture. Moreover, the factorization structure should be analyzed further, in which case we expect to uncover dimensional properties of the outside environment.

References

1. Pouget, A., Dayan, P., Zemel, R.: Information processing with population codes. *Nature Reviews Neuroscience* **1** (2000) 125–132
2. Andersen, R.A., Essick, G.K., Siegel, R.M.: Encoding of spatial location by posterior parietal neurons. *Science* **230** (1985) 456–458
3. Salinas, E., Sejnowski, T.J.: Correlated neuronal activity and the flow of neural information. *Nature reviews neuroscience* **2** (2001) 539–550
4. Jordan, M.I., Rumelhart, D.E.: Forward models: Supervised learning with a distal teacher. *Cognitive science* **16** (1992) 307–354
5. Doya, K.: *Bayesian brain: Probabilistic approaches to neural coding*. MIT press (2007)
6. Kneissler, J., Butz, M.V.: Learning spatial transformations using structured gain-field networks. In: *Artificial Neural Networks and Machine Learning–ICANN 2014*. Springer (2014) 683–690
7. Memisevic, R., Hinton, G.E.: Learning to represent spatial transformations with factored higher-order boltzmann machines. *Neural Computation* **22** (2010) 1473–1492
8. McCloskey, M., Cohen, N.J.: Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of learning and motivation* **24** (1989) 109–165
9. Zipser, D., Andersen, R.A.: A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. *Nature* **331** (1988) 679–684

Nyström approximation toolbox

Andrej Gisbrecht¹ and Frank-Michael Schleich²

¹ CITEC centre of excellence, Bielefeld University, 33615 Bielefeld, Germany

² School of Computer Science, University of Birmingham, Birmingham B15 2TT, UK
 agisbrec@techfak.uni-bielefeld.de schleich@cs.bham.ac.uk

One common challenge in many research areas is the increasing complexity of the data. Such data can no longer be represented in vectorial form, but instead specifically designed (dis-)similarity measures have to be introduced. For similarities a large variety of kernel-based techniques can be used [4], but there are also techniques which can work with dissimilarities, such as e.g. [1]. Unfortunately, (dis-)similarity measures result from domain specific considerations and therefore often do not stem from Euclidean spaces. In this case, kernel methods can not be used, since they require valid positive semi-definite kernel matrices. Distance-based techniques usually still work, but no convergence guaranties can be given. There are transformation techniques which can make these data Euclidean, but they are very demanding with the computational complexity cubic in the number of data points [3]. This leads to the next problem of (dis-)similarity data: the pairwise relations between every pair of data points have to be evaluated, resulting in squared memory complexity and at least squared runtime complexity for the associated techniques. Thus, working with data sets with already as few as ten thousand points becomes infeasible in practice.

To overcome the problem of the squared complexity the Nyström approximation was proposed for the kernel based techniques [5]. However, it was limited to positive semi-definite matrices only, which made it inaccessible for dissimilarities and indefinite similarities. Later it was shown [2], that the Nyström approximation can also be applied to arbitrary symmetric matrices. Moreover, the transformations described in [3], which allow to transform the (dis-)similarities into each other, as well as to perform Euclidean correction, can be combined with this approximation and thus carried out in linear time [2]. The toolbox presented here implements these approximated transformations and can be used to design techniques capable of dealing with large (dis-)similarity data sets.

The idea of the Nyström technique is to approximate the given matrix \mathbf{S} by a low rank matrix [5]:

$$\mathbf{S} \approx \mathbf{S}_{N,m} \mathbf{S}_{m,m}^\dagger \mathbf{S}_{N,m}^\top \quad (1)$$

where $\mathbf{S}_{N,m}$ is a linear part of the full matrix, which consists of the (dis-)similarities between all N points and m randomly selected points, also called landmarks. The matrix $\mathbf{S}_{m,m}^\dagger$ denotes the pseudo-inverse matrix calculated on the (dis-)similarity matrix between the landmarks. This way only a small part of the matrix, which is linear in the number of data points, has to be computed and any algorithm in which the matrix is used only in vector-matrix products can be carried out in linear time if the matrices are multiplied in the optimal order: $\mathbf{v}\mathbf{S} \approx ((\mathbf{v}\mathbf{S}_{N,m}) \mathbf{S}_{m,m}^\dagger) \mathbf{S}_{N,m}^\top$.

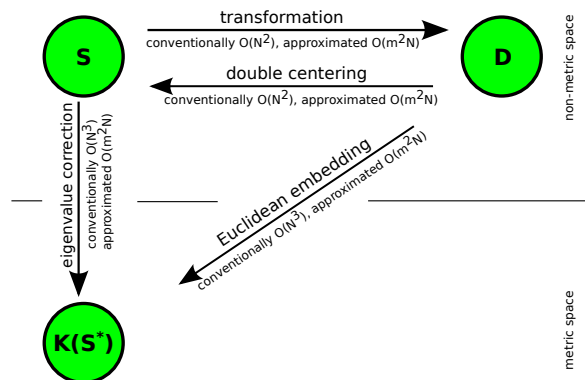


Fig. 1. Transformations between metric and non-metric (dis-)similarities. The Nyström approximation allows linear complexity.

The toolbox implements the transformations as shown schematically in Figure 1. It is possible to e.g. start with non-metric dissimilarities, convert them to similarities, perform eigenvalue correction to construct a valid kernel, and then convert the kernel back to dissimilarities, which then have properties of Euclidean distances. It is also possible to perform out-of-sample extension, i.e. if an algorithm was trained on corrected dissimilarities and should be evaluated on the uncorrected test set, it is possible to correct the new data in the same way as the training data.

The toolbox is available online at www.cit-ec.de/en/tcs/research. For detailed mathematical background on the transformations please refer to [3], for details regarding the approximation of indefinite matrices see [2]. Also if you use this toolbox please cite [2].

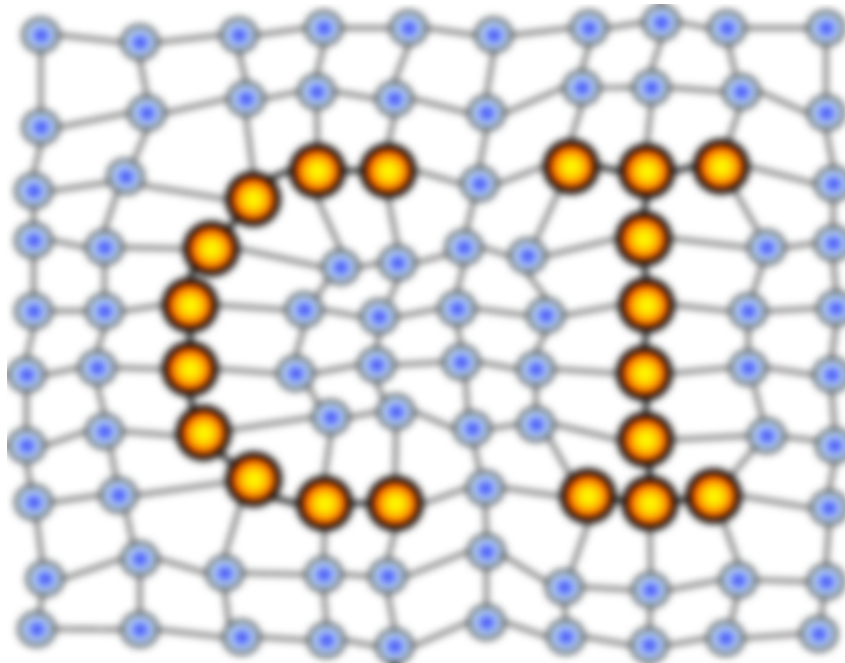
Acknowledgments: Financial support from the Cluster of Excellence 277 Cognitive Interaction Technology funded by the German Excellence Initiative is gratefully acknowledged. F.-M. Schleif was supported by a Marie Curie Intra-European Fellowship (IEF): FP7-PEOPLE-2012-IEF (FP7-327791-ProMoS).

References

1. Andrej Gisbrecht, Bassam Mokbel, and Barbara Hammer. Relational generative topographic mapping. *Neurocomputing*, 74(9):1359–1371, 2011.
2. Andrej Gisbrecht and Frank-Michael Schleif. Metric and non-metric proximity transformations at linear costs. *Neurocomputing*, 167:643–657, 2015.
3. Elzbieta Pekalska and Robert P.W. Duin. *The Dissimilarity Representation for Pattern Recognition. Foundations and Applications*. World Scientific, 2005.
4. J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis and Discovery*. Cambridge University Press, 2004.
5. Christopher Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.

MACHINE LEARNING REPORTS

Report 03/2015



Impressum

Machine Learning Reports

ISSN: 1865-3960

▽ Publisher/Editors

Prof. Dr. rer. nat. Thomas Villmann
University of Applied Sciences Mittweida
Technikumplatz 17, 09648 Mittweida, Germany
• <http://www.mni.hs-mittweida.de/>

Dr. rer. nat. Frank-Michael Schleif
University of Bielefeld
Universitätsstrasse 21-23, 33615 Bielefeld, Germany
• <http://www.cit-ec.de/tcs/about>

▽ Copyright & Licence

Copyright of the articles remains to the authors.

▽ Acknowledgments

We would like to thank the reviewers for their time and patience.