Learning Max Mixtures

Edwin Olson
elowson@umich.edu
Computer Science and Engineering
University of Michigan
The ugly
Errors

- Classical SLAM systems are very sensitive to errors
  - Natural question: How do we reduce the error rate?
Reducing Error Rates

- Neira: JCBB (2001)
- Bosse: Loop Validation (2004)
- Us: SCGP (2008)
- Us: Correlative Scan Matching (2009)
- Us: IPJC (2012)
The problem

• Each of these methods pushes error rates closer to zero. Great!

• But mapping methods can diverge with even a single error. Not Great!

• Outlier-rejection/loop-validation methods can postpone failure, but can’t eliminate it!
Let’s take a fresh perspective
Let’s take a fresh perspective

• What is an error, anyway?
Let’s take a fresh perspective

• What is an error, anyway?
  ‣ It’s an inconsistency between our probabilistic observation model and the empirical accuracy of our data association method.
Let’s take a fresh perspective

- What is an error, anyway?
  - It’s an inconsistency between our probabilistic observation model and the empirical accuracy of our data association method.
  - Maybe the problem isn’t outlier rejection, maybe the problem is that we’re using the wrong probabilistic models!
Gaussian error models

\[ p(z_i | x) = N(\mu_i, \Lambda^{-1}_i) \]
Gaussian error models

- Almost all robotics work uses Gaussian error models
  - Lead to very simple least-squares state estimation algorithms.
  - Believed to be sufficiently representative

\[
p(z_i | x) = N(\mu_i, \Lambda_i^{-1})
\]
Gaussian Errors = Easy inference

\[ p(z_i|x) = N(\mu_i, \Lambda_i^{-1}) = \frac{1}{\gamma} e^{-\frac{1}{2}(x-\mu)^T \Lambda_i(x-\mu)} \]

\[ x_{opt} = \arg\max_x \prod_i p(z_i|x) \]

\[ x_{opt} = \arg\max_x \log \left( \prod_i p(z_i|x) \right) \]

\[ x_{opt} = \arg\max_x \sum_i \log p(z_i|x) \]

\[ x_{opt} = \arg\max_x \sum_i (a_i x^2 + b_i x + c_i) \]

\[ A x_{opt} = b \quad \text{unreasonably fast methods now available!} \]
Real-world errors (maybe)

\[ p(\text{distance}) \]

Odometry *Slip or Grip*
Loop closure: null hypothesis
Sonar multi-path / surface reflections / loop closing with aliasing

\[ p(\text{range}) \]
Laser Scan Matching - Cost Function

Reference Map

15m x 15m

Cost surface for pure translations (even uglier for bad rotations!)

4m x 4m

Edwin Olson
Two basic problems

- How can we represent these more complex error models?

- How do we solve the resulting inference problems?
Sums of Gaussians

- One “obvious” way to represent more types of error functions

\[
p(z_i|x) = N(\mu_i, \Lambda_i^{-1}) \quad \Rightarrow \quad p(z_i|x) = \sum_i w_i N(\mu_i, \Lambda_i^{-1})
\]

\[
p(z_{31}|x) = 0.1 N(0, 0.25) + 0.9 N(1, 0.5)
\]
Sums of Gaussians

\[ p(z_i | x) = \sum_i w_i N(\mu_i, \Lambda_i^{-1}) \]

\[ x_{opt} = \arg \max_x \prod_i p(z_i | x) \]

\[ x_{opt} = \arg \max_x \log \left( \prod_i p(z_i | x) \right) \]

\[ x_{opt} = \arg \max_x \sum_i \log p(z_i | x) \]

\[ x_{opt} = \arg \max_x \sum_i \log \left( \sum_j w_j N(\mu_j, \Lambda_j^{-1}) \right) \]
Sums of Gaussians

\[
x_{opt} = \arg\max_x \sum_i \log \left( \sum_j w_j N(\mu_j, \Lambda_j^{-1}) \right)
\]

\[
x_{opt} = \frac{1}{\Lambda} \mathbf{1}
\]

\[
x_{opt} = \arg\max_i p(z_i | x)
\]

\[
x_{opt} = \arg\max_i \sum_j w_j N(\mu_j, \Lambda_j^{-1})\]
Challenge

• Can we find a way of representing more complex error functions?

• AND make sure that we can actually solve the resulting problem?
Our approach

\[ p(z_i | x) = \sum_i w_i N(\mu_i, \Lambda_i^{-1}) \]

\[ p(x | z) \propto \prod_i p(z_i | x) \]
Our approach

- Use mixture models for more realistic probability distributions

  - Change SUM to MAX

\[
p(z_i | x) = \sum_i w_i N(\mu_i, \Lambda_i^{-1})
\]

\[
p(x | z) \propto \prod_i p(z_i | x)
\]
Our approach

• Use mixture models for more realistic probability distributions

  ‣ Change SUM to MAX

\[
\begin{align*}
    p(z_i|x) &= \sum_i w_i N(\mu_i, \Lambda_i^{-1}) \\
    p(x|z) &\propto \prod_i p(z_i|x)
\end{align*}
\]

\[
\begin{align*}
    p(z_i|x) &= \max_i w_i N(\mu_i, \Lambda_i^{-1}) \\
    p(x|z) &\propto \prod_i p(z_i|x)
\end{align*}
\]

• Can “push” the log past the MAX...
Our approach

- Use mixture models for more realistic probability distributions
  
  - Change SUM to MAX

  \[
p(z_i | x) = \sum_i w_i N(\mu_i, \Lambda_i^{-1}) \quad \Rightarrow \quad p(z_i | x) = \max_i w_i N(\mu_i, \Lambda_i^{-1})
\]

  \[
p(x | z) \propto \prod_i p(z_i | x)
\]

  - Can “push” the log past the MAX...

  - Results in Ax=b, just like with simple Gaussian error models
Max Mixtures: Examples

Figure 2: Mixture Overview. Given two mixture components (top left), the max- and sum-mixtures produce different distributions. In both cases, arbitrary distributions can be approximated. A robustified cost function (in this case a corrupted Gaussian, bottom) can be constructed from two Gaussian components with equal means but different variances.

Cost function interpretation (Corrupted Gaussian)
Inference on Max Mixtures

- Max mixture formulation is highly suggestive of an optimization strategy:

  - Inference on Gaussians:
    - For each observation $p(z_i|x)$
      - Compute contribution to $A$ matrix and $b$ vector from the Gaussian.
    - Solve $Ax = b$

  - Inference on Max Mixtures
    - For each observation $p(z_i|x)$
      - Find mixture component $j$ that is most likely given $x$.
      - Compute contribution to $A$ matrix and $b$ vector from Gaussian mixture component $j$.
    - Solve $Ax = b$
The $20$ question

• Least-Squares regression (inference on Gaussians) is convex*:
  ‣ A single minimum
  ‣ All starting points lead to the minimum
  ‣ Well behaved optimization problem!

• This is not true of max mixtures:
  ‣ Exponentially many local minima!
  ‣ Does this scheme robustly find the global minimum?
Slip or Grip (Toy Problem)

Devising a stochastic method for the max-mixture case is even more challenging than the uni-modal case. The Cholesky-MM approach is not generalizable: for the uni-modal case, it results in an implausible maximum likelihood solution. Since the uni-modal case is a special instance of the max-mixture case, the Cholesky-MM solution would be over-prescriptive for the uni-modal case as well.

However, the max-mixture approach provides a solution to the "slip or grip" problem: the case where a robot’s wheels occasionally slip catastrophically, resulting in near or complete absence of proprioceptive feedback. In this case, the robot is obtaining loop closures for features not traversed by the robot's wheels, and the resulting map contains "false positives". Our method accepts some randomly-generated "false positives", but an analysis of the error of those edges indicates that they are only slightly worse than the error of true edges.
Results

Max-mixture

1 error

10 errors

100 errors

Standard Gaussian

22
Results: CPU Time

![Graph showing CPU time vs. node processed with two lines: Cholesky-MM and Normal Cholesky.](image-url)
Results: Robustness

**Standard methods**

**Proposed method**

MSE error vs. Error percentage in loop closure edges
Does it work in 6DOF?

- 3D generally much harder than 2D:
  - Rotations exacerbate local minima problems due to non-linear effects.

4.3 Extension to 6DOF

While many important domains can be described in terms of planar motion (with three-dimensional factor potentials reflecting translation in x, translation in y, and rotation), there is increasing interest in 6 degree-of-freedom problems. Rotation is a major source of non-linearity in SLAM problems, and full six degree-of-freedom problems can be particularly challenging.

To evaluate the performance of our method on a six degree-of-freedom problem, we used the benchmark Sphere2500 dataset [23]. This dataset does not contain incorrect loop closures, and so we added additional erroneous loop closures. In Fig. 7, we show the results of a standard Cholesky solver and our max mixture approach applied to corrupted Sphere2500 dataset with an additional 1, 10, and 100 erroneous edges. As in previous examples, the maps produced by a standard method quickly deteriorate. In contrast, the proposed method produces posterior maps that are essentially unaffected by the errors. In this experiment, each loop closure edge in the graph (both correct and false) was modeled as a two-component max mixture in which the second component had a large variance ($10^7$ times larger than the hypothesis itself) and a small weight ($10^{-7}$). The method is relatively insensitive to the particular values used: the critical factor is ensuring that, ...)
“Extreme SLAM”

Key concepts:
- Pose graph, nodes and edges
- Map
“Extreme SLAM”

Key concepts:
- Pose graph, nodes and edges
- Map
Eliminating the front-end

- Create a max-mixture with N+1 components
  - Best N matches (based on scan matching)
  - One “null” component.
  - (Encodes an interesting mutual-exclusion property!)

- No loop validation, no geometric constraints.
Eliminating the front-end

In the previous section, uncertain data associations were modeled as “one-in-k” mixtures, in which multiple candidate loop closures were grouped together in a single edge. Alternatively, each candidate loop closure could be encoded as a two-component mixture in a “null-hypothesis” style mixture; this approach is well-suited to the case where little is known about alternatives to a putative loop closure, while still allowing for the possibility that it is incorrect. (It is also possible that the mixture components have no obvious semantic meaning: the mixture model could simply be approximating a more complex distribution. For example, a max mixture could be fit to an empirically derived cost function from a correlation-based scan matcher [33].)

In this section, we explore the performance impact of “one-in-k” mixtures versus “null-hypothesis” mixtures. Consider a “one-in-k” mixture consisting of three candidate loop closures plus a null hypothesis:

\{L_1, L_2, L_3, \text{null}\}.

This can be transformed into three “null-hypothesis” mixtures:

\{L_1, \text{null}\}, \{L_2, \text{null}\}, and \{L_3, \text{null}\}.

These two formulations are not exactly equivalent: the “one-in-k” encodes mutual-exclusion between the hypotheses, whereas the \(k\) separate “null-hypotheses” would permit solutions in which more than one of the loop closures was accepted. In many practical situations, however, the semantic difference is relatively minor. In this section, we show that the

Figure 9: Intel without front-end loop validation. Our system can identify correct loop closures and compute a posterior map from within a single integrated Bayesian framework (right); the typical front-end loop validation has been replaced with a \(k+1\) mixture component containing the \(k\) best laser scan matches (based purely on overlap) plus a null hypothesis. In this experiment, we used \(k=5\). For reference, the open-loop trajectory of the robot is given on the left.
Performance Analysis

• Suppose we have N hypotheses relating a pose to earlier poses (like previous problem)
  ▶ Could use ONE max mixture with N+1 Components
  ▶ Could use N max mixtures with 2 Components

• Which one is better?
### Effect of encoding on CPU time

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Switchable constraints</th>
<th>bi-modal MM</th>
<th>k-modal MM</th>
</tr>
</thead>
<tbody>
<tr>
<td>manhattan with $k = 2$ outliers = 2099</td>
<td>iter time (s)</td>
<td>0.90 s</td>
<td>0.74 s</td>
</tr>
<tr>
<td></td>
<td>fill-in (%)</td>
<td>1.50 %</td>
<td>2.89 %</td>
</tr>
<tr>
<td></td>
<td>#loop edges</td>
<td>4198</td>
<td>4198</td>
</tr>
<tr>
<td></td>
<td>#components</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>manhattan with $k = 3$ outliers = 4198</td>
<td>iter time (s)</td>
<td>1.5 s</td>
<td>1.2 s</td>
</tr>
<tr>
<td></td>
<td>fill-in (%)</td>
<td>1.70 %</td>
<td>4.30 %</td>
</tr>
<tr>
<td></td>
<td>#loop edges</td>
<td>6277</td>
<td>6277</td>
</tr>
<tr>
<td></td>
<td>#components</td>
<td>-</td>
<td>2</td>
</tr>
</tbody>
</table>

- **N+1 component mixtures exhibit much better scaling!**
  - Max mixture selects *one* component to be dominant
  - Thus, has fewer “links” between poses...
  - Sparser $A$ matrix $\Rightarrow$ faster matrix factoring methods.
Good things lead to more good things

- A system that has accumulates errors has trouble closing new loops.
- Robustly handling errors not only avoids divergence on the loops you have, it increases the number of loops you’ll find!
Some Perspective

- Other approaches exist for multi-modal inference.
  - FastSLAM (Montemerlo)
  - Multiple Hypothesis Tracking

- Both have complexity that scales with the complexity of the posterior
  - FastSLAM: Particles
  - MHT: Hypotheses (each with an EKF)

- The complexity of the posterior grows exponentially, forcing these methods to prune. Can lead to failures (e.g. particle depletion).

- MaxMixtures is fundamentally different:
  - Memory complexity grows linearly with the size of the problem.
  - Never have to approximate the problem.
  - But, no guarantee that we find the maximum-likelihood solution!
Learning GPS Covariances

• Typically very hard to get good covariance estimates
  ▶ Multi-path / Urban canyons
  ▶ Indoor/Outdoor transitions

• Lots of interesting meta-data about sensor observations, e.g.:
  ▶ # visible satellites
  ▶ HDOP (based on geometry of satellites)
  ▶ vendor’s covariance estimate
Can we learn covariances?

- Idea:
  - Construct a feature vector $\mathbf{f}$ from this metadata
  - Learn weight vector $\mathbf{w}$ such that:
    $$\sigma = \mathbf{w}^T \mathbf{f}$$
Feature Encoding

- Constant feature:
  \[ f^T = [1] \]

- Add HDOP:
  \[ f^T = [ z_{hdop} \ 1 ] \]

- Add # satellites
  \[ f^T = [ 0 \ ... \ 0 \ 1 \ 0 \ ... \ 0 \ z_{hdop} \ 1 ] \]
Other feature types

• Idea: Generate features from other sensor modalities
  ▶ Estimate “indoorness” from LIDAR data?

• (Not doing that here, but it’s something we’re looking at)
Extension to Max Mixture

- Learning weights $\mathbf{w}$ tells us covariance

$$\sigma = \mathbf{w}^T \mathbf{f}$$

- Extension to max mixture is easy:
  - Fit multiple sigmas. (Assume means are the same)

$$\sigma_i = \mathbf{w}_i^T \mathbf{f}$$

$$p(z|x) = \max_i \alpha_i N(u, \sigma_i)$$

- And learn mixing weights (alphas) too.
Evaluating the learned weights

• Standard approach:
  - Pick w’s that maximize the likelihood of the data

\[
\mathbf{w}^* = \max_{\mathbf{w}} \prod_j p(z_j | x)
\]

\[
\propto \max_{\mathbf{w}} \prod_j e^{-\frac{1}{2} (z_j - \mu)^T (w^T f)^{-2} (z_j - \mu)}
\]

(shown here for standard Gaussian approach)

• (i.e., what weights \( \mathbf{w} \) maximize the likelihood of all the observations?)
Non-mixture analysis

• The ML solution $w^*$ is good for two reasons:
  ‣ It’s the ML solution, and we’re all good Bayesians, right?
  ‣ It minimizes the occurrence of high $X^2$ observations
    • These have an increasing effect on the gradient in an optimization framework...
    • ...and are responsible for divergence of non-robust methods.

• I.e., in Gaussian case, low probability $\Rightarrow$ high cost function curvature $\Rightarrow$ divergence
Max Mixture Analysis

- It’s not the case that low probability $\Rightarrow$ high curvature $\Rightarrow$ divergence.
- E.g., “null hypothesis” components: low weight ($\Rightarrow$ low probability) but high variance ($\Rightarrow$ low curvature)
Evaluation

- So how do we evaluate a max mixture?
  - “Model Goodness”: Maximum Likelihood
  - Convergence: minimize gradient for bad data

- Our current thinking:
  - Still maximize the likelihood of the data, but...
  - Keep an eye on the gradients as an interesting check...
Constant model, $f = [1]$

- **Single Gaussian**

- **Max Mixture of two Gaussians**
Vendor+ model, $f = [v]$

- **Single Gaussian**

- **Max Mixture of two Gaussians**

---

TABLE I: Training and Testing Error.

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Training</th>
<th>Test</th>
</tr>
</thead>
</table>

---

The integration of GPS sensors into SLAM allows for robustness through the use of multiple covariance functions, which can decrease the effect of outliers, but has the side-effect of allowing a richer representation of the underlying model. Ideally, if the underlying Gaussian assumptions hold, the normalized histogram of movement of the worst-case arrows (both components have similar to a Max Mixture of two Gaussians) should be similar to a distribution. We have not provided quantitative uncertainty estimates that accurately reflects the true distribution, confirming the non-robustness of the unimodal models. On the right, the increased robustness of the modal models is apparent, reflecting that their ability to model high-error observations is improved. Possibly a better fit would use additional components, similar to a Max-Mixture model. This reflects the fact that the sensor performs better for the high-error observations with an overestimate of $\chi^2$ for the max-mixture models.
Results:

- More complex models ==> better predictions in an ML sense
- Two component mixture model yields higher likelihood

NB: Small numerical differences here are a big deal...
- These are average likelihoods over thousands of observations.
Gradients

- Generally see lower worst-case gradient magnitudes

- Interestingly, it's not the same observations causing “problems” in both cases
  - The high gradient observations are those just barely clinging to the “inliner” component.

<table>
<thead>
<tr>
<th></th>
<th>Non-MM max gradient</th>
<th>MM max gradient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>3.251</td>
<td>1.114</td>
</tr>
<tr>
<td>Vendor</td>
<td>2.331</td>
<td>2.024</td>
</tr>
<tr>
<td>Constant-Vendor</td>
<td>1.878</td>
<td>1.791</td>
</tr>
<tr>
<td>Constant-HDop-NSat-Vendor</td>
<td>2.127</td>
<td>1.836</td>
</tr>
</tbody>
</table>
Something Outrageous

• What do I care about?
  ▶ Robustness (to outliers) (to initial estimates)
  ▶ Where do error-models/hyper-parameters come from?

• What do I care less about?
  ▶ Inference speed
  ▶ Batch problems