Gaussian Processes for Bayesian Filtering

Dieter Fox, Jonathan Ko, Brian Ferris
University of Washington
Department of Computer Science & Engineering
Bayesian Filtering

- Estimate state of a dynamical system from sensor data

- Key problems in robotics
  - Localization, mapping, people tracking, activity recognition, POMDPs, ...

- Various instantiations / approximations
  - Kalman filter, EKF, UKF, particle filters, grid filters, ...

## Bayes Filters

- **Dynamics model:** \[ p(s(k) \mid s(k-1), u(k-1)) \]
- **Observation model:** \[ p(z(k) \mid s(k)) \]
Observation and Dynamics Models

- Typically parametric function describing underlying physical process with additive noise
- Parameters tuned or learned from data
- **Problems**
  - Parametric models might miss certain aspects
  - Very hard for high dimensional features
Use Gaussian Process regression to learn dynamics and observation models
Replace noise parameters with uncertainty from GP
Gaussian Process Models

- Wireless signal strength [Ferris et al: RSS-06]
- Failure detection [Plagemann et al: IJCAI-07]
- Gas distribution [Plagemann et al: RSS-08]
- Robotic arm dynamics [Nguyen-Tuong et al: NIPS-08]
GP Setting

- Outputs are noisy function of inputs:
  \[ y_i = f(x_i) + \varepsilon \]
- GP prior: Outputs jointly zero-mean Gaussian:
  \[ p(y) = \mathcal{N}(y; 0, K + \sigma_n^2 I) \]
- Covariance given by kernel matrix over inputs:
  \[
  K = \begin{pmatrix}
  k(x_1, x_1) & \ldots & k(x_1, x_n) \\
  k(x_2, x_1) & \ddots & \vdots \\
  \vdots & \ddots & k(x_i, x_i) \\
  k(x_n, x_1) & \ldots & k(x_n, x_n)
  \end{pmatrix}
  \]
  \[ k(x, x') = \sigma_f^2 e^{-\frac{1}{2}(x-x')W(x-x')^T} \]
Functions Sampled from Priors
GP Prediction

- Training data:
  \[ D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} = (X, y) \]

- Prediction given training samples:
  \[ p(y^* \mid x^*, y, X) = N\left( y^* ; \mu, \sigma^2 \right) \]

  \[ \mu = k^* \begin{pmatrix} \sigma_n^{-1} I \end{pmatrix} k \]
  \[ \sigma^2 = k(x^*, x^*) - k^* k \begin{pmatrix} \sigma_n^{-1} I \end{pmatrix}^{-1} \]
1D Gaussian Process Example
Kernel Width

Kernel width: 0.2
Local likelihood: -0.566
Hyperparameter Estimation

- Maximize data log likelihood:
  \[ \theta_* = \arg \max_{\theta} p(y \mid X, \theta) \]
  \[ \log p(y \mid X, \theta) = -\frac{1}{2} y^T (K + \sigma_n^2 I)^{-1} y - \frac{1}{2} \log (K + \sigma_n^2 I) - \frac{n}{2} \log 2\pi \]

- Compute derivatives wrt. params \[ \theta = \langle \sigma_n^2, l, \sigma_f^2 \rangle \]
- Optimize using conjugate gradient descent
WiFi Sensor Model

Mean

Variance
Location Model
Tracking Example
GP-BayesFilters

- Learn GP:
  - Input: Sequence of ground truth states along with controls and observations: <s, u, z>
  - Learn GPs for dynamics and observation models

- Filters
  - Particle filter: sample from dynamics GP, weigh by GP observation function
  - EKF: GP for mean state, GP derivative for linearization
  - UKF: GP for sigma points
Blimp Tracking
GP-UKF Tracking Example
## Blimp Results

<table>
<thead>
<tr>
<th>Tracking algorithm</th>
<th>pos(mm)</th>
<th>rot(deg)</th>
<th>vel(mm/s)</th>
<th>rotvel(deg/s)</th>
<th>MLL</th>
<th>time(sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GP-PF</td>
<td>91 +/- 7</td>
<td>6.4 +/- 1.6</td>
<td>52 +/- 3.7</td>
<td>5.0 +/- .2</td>
<td>9.4 +/- 1.9</td>
<td>449.4 +/- 21</td>
</tr>
<tr>
<td>GP-EKF</td>
<td>93 +/- 1</td>
<td>5.2 +/- .1</td>
<td>52 +/- .5</td>
<td>4.6 +/- .1</td>
<td>13.0 +/- 2</td>
<td>.29 +/- .1</td>
</tr>
<tr>
<td>GP-UKF</td>
<td>89 +/- 1</td>
<td>4.7 +/- .2</td>
<td>50 +/- .4</td>
<td>4.5 +/- .1</td>
<td>14.9 +/- .5</td>
<td>1.28 +/- .3</td>
</tr>
<tr>
<td>ParaPF</td>
<td>115 +/- 5</td>
<td>7.9 +/- .1</td>
<td>64 +/- 1.2</td>
<td>7.6 +/- .1</td>
<td>-4.5 +/- 4.2</td>
<td>30.7 +/- 5.8</td>
</tr>
<tr>
<td>ParaEKF</td>
<td>112 +/- 4</td>
<td>8.0 +/- .2</td>
<td>65 +/- 2</td>
<td>7.5 +/- .2</td>
<td>8.4 +/- 1</td>
<td>.21 +/- .1</td>
</tr>
<tr>
<td>ParaUKF</td>
<td>111 +/- 4</td>
<td>7.9 +/- .1</td>
<td>64 +/- 1</td>
<td>7.6 +/- .1</td>
<td>10.1 +/- 1</td>
<td>.33 +/- .1</td>
</tr>
</tbody>
</table>

- Blimp tracking using multiple cameras
- Ground truth obtained via Vicon motion tracking system
- Parametric model takes drag, thrust, gravity, etc, into account
- Can (and should) incorporate parametric model!
Dealing with Training Data Sparsity
Going Latent

- Sometimes ground truth states are not or only partially available.
- Instead of optimizing over hyperparameters only
  \[ \theta^* = \arg \max_{\theta} p(y \mid X, \theta) \]
  optimize over latent states \( X \) as well
  \[ \langle X^*, \theta^* \rangle = \arg \max_{X, \theta} p(y \mid X, \theta) \]
- **GPLVM**: non-linear probabilistic dimensionality reduction
WiFi-SLAM: Mapping without Ground Truth

[Fierris-Fox-Lawrence: IJCAI-07]
GPDM: Latent Variable Models for Dynamical Systems

- GPLVM with additional constraints on latent variables to model dynamical system

\[ p(y, X, \theta) = p(y | X, \theta)p(X | \theta)p(\theta) \]

\[ \langle X_*, \theta_* \rangle = \arg\max_{X, \theta} p(y, X, \theta) \]

- Dynamics are modeled via another GP
GPBF-Learning

- Extend GPDMs to
  - incorporate control
  - incorporate sparse labels on latent states

- Steps:
  - Learn GPLVM
  - Extract GP-BayesFilter for tracking

- See Jonathan’s talk on Tuesday!
Slotcar Tracking
Data

Controls

IMU: angle change rate
Learned 2D Embedding

N4SID

GPBF-Learn
Time Alignment

Raw  N4SID  GPBF-Learn

RSS Workshop on Regression in Robotics  Fox, Ko: GP-BayesFilters
Beyond Plain GP Regression

- **Heteroscedastic** (state dependent) noise
  [Quoc-etal: ICML-05; Kersting-etal: IMCL-07]

- **Efficiency**: **Sparse** GPs
  [Snelson and Ghahramani: NIPS-06; Smola and Bartlett: NIPS-01]

- **Discrete** state components: GP classification
  [Plagemann etal: IJCAI-07]
Summary

- GPs provide **flexible modeling framework**
- Take **data noise and uncertainty due to data sparsity** into account
- Combination with parametric models increases accuracy and reduces need for training data
- Computational **complexity** is a key problem
- Should only be used if necessary