Gaussian Processes for Bayesian Filtering

Dieter Fox, Jonathan Ko, Brian Ferris

University of Washington
Department of Computer Science & Engineering

Intel Labs Seattle
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, RBPF, ...
Bayes Filters

\[
p(x_k \mid z_{1:k}, u_{1:k-1}) \propto \int p(z_k \mid x_k) \cdot p(x_k \mid x_{k-1}, u_{k-1}) \cdot p(x_{k-1} \mid z_{1:k-1}) \, dx_{k-1}
\]
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
Non-Parametric GP Regression

\[ p(y^* | x^*, y, X) = N\left(y^*; \mu, \sigma^2\right) \]

\[ \mu = k^*^T \left(K + \sigma_n^2 I\right)^{-1} y; \quad \sigma^2 = k(x^*, x^*) - k^*^T \left(K + \sigma_n^2 I\right)^{-1} k^* \]
Use ground truth state to extract:

- **Dynamics GP training data**
  \[ D_x = \langle [x_1, u_1], \Delta x_1 \rangle, \langle [x_2, u_2], \Delta x_2 \rangle \]

- **Observation GP training data**
  \[ D_z = \langle x_2, z_2 \rangle, \langle x_3, z_3 \rangle \]
GP-BayesFilters

- Learn GP:
  - Input: Sequence of ground truth states along with controls and observations: <x, u, z>
  - Learn GPs for dynamics and observation models
- Filters
  - GP-PF: sample from dynamics GP, weigh by GP observation function
  - GP-UKF: GPs for sigma points, additive noise
  - GP-EKF: apply GP to mean state, GP derivative for linearization, additive noise
  - GP-ADF: assumed density filtering [Deisenroth et al, ICML-09]
WiFi Sensor Model

Mean

Variance

Probabilistic Approaches for Robotics and Control

Fox, Ko: GP-BayesFilters
Tracking Example
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+/-1.1</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+/-1.1</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+/-1.1</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

![Graph showing Yaw vs Time with shaded region indicating sparsity]

Probabilistic Approaches for Robotics and Control  Fox, Ko: GP-BayesFilters  13
Lack of Ground Truth Data

- Noisy states
- Sparse states
- No states

- Weak labels denoted by $\hat{X}$
Going Latent: GPLVM / GPDMs

- **GPLVM**: optimize over latent states $X$ as well

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

[Lawrence: NIPS-03, JMLR-05]

- **GPDM**: add dynamics GP

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

[Wang et al: PAMI-08]
Extension to GP-BayesFilters

\[ p(X, \Theta_Z, \Theta_X \mid Z, U, \hat{X}) \propto \]

\[ p(Z \mid X, \Theta_Z) p(X \mid \Theta_X, U) p(X \mid \hat{X}) \]

\[ p(\Theta_Z) p(\Theta_X) \]

- Components
  - Observation model
  - Dynamics model with controls
  - Weak labels
  - Hyperparameter priors

[Probabilistic Approaches for Robotics and Control]
**GPBF Learning Algorithm**

- Optimize latent states
  - Initialize latent states
    - Weak labels if available
    - N4SID otherwise
  - Optimize $\arg \max_{X, \Theta_Z, \Theta_X} \log p(X, \Theta_Z, \Theta_X | Z, U, \hat{X})$

- Extract GP-BayesFilter from data
  - Can take advantage of sparse GP representation
Slotcar Tracking
Data

Controls

IMU: angle change rate
Tracking from Noisy State Labels

- Track with GP-UKF
- Averaged over 10 runs
- Similar results for sparsity
Simple Trajectory Replay

- Record demo traces
- Learn latent state
- Extract GP-BayesFilter for tracking
- Learn GP mapping from $X$ to $U$

- Track car in latent state and send learned control
Learned Latent Space

Recovered Latent States

-0.16
-0.1
-0.06
-0.02
0
0.02
0.06
0.08

GPBF-LF
N4SID
GPs provide flexible modeling framework
Take data noise and uncertainty due to data sparsity into account
Combination with parametric models increases accuracy and reduces training data
Computational complexity is a key problem
First steps toward trajectory replay
Also: heteroscedastic, sparse, discriminative

Should only be used if necessary
Some additional Remarks

- Combination with parametric model works best

- 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]
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 manipulation [Nguyen-Tuong et al: NIPS-08; Deshpande et al: ICRA-09]