Gaussian Processes for Bayesian Filtering

Dieter Fox
University of Washington

Joint work with
Jonathan Ko, Brian Ferris, Marc Deisenroth, Ashish Deshpande, Neil Lawrence

Funded by NSF and ONR
Bayes Filters

• Task: Estimate state of a dynamical system from sensor data and control information

• Key problems in robotics
  – Localization, mapping, people and object tracking, activity recognition, POMDPs, ...

• Various instantiations / approximations
  – Kalman filter, EKF, UKF, ADF, particle filters, grid filters
• **Typical Bayesian filtering**
  – Parametric dynamics and observation models
  – Approximate posterior via sampling (PF), sigma points (UKF), linearization (EKF), moment matching (ADF)

• **GP-BayesFilters**
  – GP dynamics and observation models
  – Noise derived from GP prediction uncertainty
  – Can be integrated seamlessly into Bayes filters: EKF, UKF, PF, ADF
Learning GP Dynamics and Observation Models

• Ground truth training sequence:
  \[ S = [s_1, s_2, ..., s_n], Z = [z_1, z_2, ..., z_n], U = [u_1, u_2, ..., u_n] \]

• Learn observation and dynamics GPs:

  \[ s_k \rightarrow \text{GP observation model} \rightarrow z_k \]

  \[ [s_k, u_k] \rightarrow \text{GP dynamics model} \rightarrow \Delta s_k = s_{k+1} - s_k \]

  \[ [s_k, u_k] \rightarrow \text{EGP dynamics model} \rightarrow r_k = \Delta s_k - f(s_k, u_k) \]

• Learn separate GP for each output dimension

• Diagonal noise matrix

[Deisenroth-et al] introduced GP-ADFs and EP for smoothing in GP dynamical systems
GP-PF Propagation

- Propagate each particle using GP prediction
- Sample from GP uncertainty
- One GP mean and variance prediction per particle

\[
s^m_{k+1} = \text{GP}_\mu(s^m_k, u) + \text{sample}\left[\text{GP}_\Sigma(s^m_k)\right]
\]
GP-EKF Propagation

\[ \langle \mu_k, \Sigma_k \rangle \xrightarrow{\text{Propagate mean using GP prediction}} \langle \mu_{k+1}, \Sigma_{k+1} \rangle \]

\[ \mu_{k+1} = GP_\mu(\mu_k) \]

\[ G = \frac{\partial GP_\mu(\mu_k)}{\partial s} \]

\[ \Sigma_{k+1} = G \Sigma_k G^T + GP_\Sigma(\mu_k) \]

- Propagate mean using GP prediction
- Use gradient of GP to propagate covariance
GP-UKF Propagation

\[ \langle \mu_k, \Sigma_k \rangle \quad \langle \mu_{k+1}, \Sigma_{k+1} \rangle \]

\[ \chi_k = \left( \mu_k, \mu_k + \gamma \sqrt{\Sigma_k}, \mu_k - \gamma \sqrt{\Sigma_k} \right) \]

for \( i = 0 \ldots 2n \): \( \chi_{k+1} = GP_\mu(\chi_k) \)

\[ \mu_{k+1} = \sum_{i=0}^{2n} \omega_m^i \chi_{k+1}^i \]

\[ \Sigma_{k+1} = \sum_{i=0}^{2n} \omega_c^i (\chi_{k+1}^i - \mu_{k+1})(\chi_{k+1}^i - \mu_{k+1})^T + GP_\Sigma(\mu_k) \]

- Propagate each sigma point using GP prediction
- 2d+1 sigma points -> 2d+1 GP mean predictions
Algorithm Extended Kalman Filter ($\mu_{t-1}, \Sigma_{t-1}$):

1. $\hat{\mu}_t = g(\mu_{t-1}, u_{t-1})$
2. $G_t = g'(\mu_t, u_{t-1})$
3. $\hat{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + Q_t$
4. $\hat{z}_t = h(\hat{\mu}_t)$
5. $H_t = h'(\hat{\mu}_t)$
6. $K_t = \hat{\Sigma}_t H_t^T (H_t \hat{\Sigma}_t H_t^T + R_t)^{-1}$
7. $\mu_t = \hat{\mu}_t + K_t(z_t - \hat{z}_t)$
8. $\Sigma_t = (I - K_t H_t) \hat{\Sigma}_t$
9. return $\mu_t, \Sigma_t$

Algorithm GP-EKF ($\mu_{t-1}, \Sigma_{t-1}, u_{t-1}, z_t$):

1. $\hat{\mu}_t = \mu_{t-1} + GP_{\mu}([\mu_{t-1}, u_{t-1}], D_p)$
2. $Q_t = GP_{\Sigma}([\mu_{t-1}, u_{t-1}], D_p)$
3. $G_t = I + \frac{\partial GP_{\mu}([\mu_{t-1}, u_{t-1}], D_p)}{\partial x_{t-1}}$
4. $\hat{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + Q_t$
5. $\hat{z}_t = GP_{\mu}(\hat{\mu}_t, D_o)$
6. $R_t = GP_{\Sigma}(\hat{\mu}_t, D_o)$
7. $H_t = \frac{\partial GP_{\mu}(\hat{\mu}_t, D_o)}{\partial x_t}$
8. $K_t = \hat{\Sigma}_t H_t^T (H_t \hat{\Sigma}_t H_t^T + R_t)^{-1}$
9. $\mu_t = \hat{\mu}_t + K_t(z_t - \hat{z}_t)$
10. $\Sigma_t = (I - K_t H_t) \hat{\Sigma}_t$
11. return $\mu_t, \Sigma_t$
WiFi-Based Location Estimation

[Similar to [Schwaighofer-et al: NIPS-03]]

[Ferris-Haehnel-Fox: RSS-06]
Building Model
Tracking Example
WiFi-SLAM: Mapping without Ground Truth Using GPLVMs
Blimp Testbed

- Task: Track a blimp with two webcams
- Baseline: Parametric model that takes drag, thrust, gravity, etc, into account
- GP-BayesFilters and parametric model trained on ground truth data obtained with Vicon motion capture system
GP-UKF Tracking Example

- Blue ellipses: sigma points projected into observation space
- Green ellipse: Mean state estimate
## Tracking Results

<table>
<thead>
<tr>
<th>Method</th>
<th>GP</th>
<th>EGP</th>
<th>hetGP</th>
<th>sparseGP</th>
</tr>
</thead>
<tbody>
<tr>
<td>UKF</td>
<td>30.75 ± 1.41</td>
<td>34.10 ± 1.76</td>
<td>35.76 ± 1.61</td>
<td>32.05 ± 2.02</td>
</tr>
<tr>
<td>EKF</td>
<td>27.66 ± 1.04</td>
<td>31.44 ± 2.43</td>
<td>33.70 ± 2.09</td>
<td>29.72 ± 1.90</td>
</tr>
<tr>
<td>PF</td>
<td>33.93 ± 7.24</td>
<td>35.95 ± 6.91</td>
<td>na</td>
<td>38.92 ± 2.17</td>
</tr>
</tbody>
</table>

Percentage reduction in RMS over parametric baseline

- Cross validation with 900 timesteps for training
- sparseGP: sparsified to 50 active points [Snelson-Ghahramani: NIPS-06]
Dealing with Training Data Sparsity

- Training data for right turns removed

![Graphs showing full process model tracking and no right turn process model tracking.](image)
Heteroscedastic GP

- Grey shading indicates tail motor power
GP-UKF Issue

- Simulated robot moving in circuit while observing landmarks
- Sigma points in region of low training sample density
- Poor GP prediction leads to large UKF tracking error

<table>
<thead>
<tr>
<th></th>
<th>posErr</th>
<th>velErr</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPUKF</td>
<td>74.6</td>
<td>8.0</td>
</tr>
<tr>
<td>GPEKF</td>
<td>48.2</td>
<td>7.2</td>
</tr>
<tr>
<td>GPPF</td>
<td>43.8</td>
<td>6.0</td>
</tr>
</tbody>
</table>
Going Latent

• Sometimes ground truth states are not or only partially available

• Instead of optimizing over GP hyperparameters only, optimize over latent states $S$ as well
GP Latent Variable Models

- Latent variable models [Lawrence: NIPS-03, Wang-etal: PAMI-08]
- Learn latent states and GPs in one optimization

\[
\arg\max_{S, \Theta_Z, \Theta_S} \log p(S, \Theta_Z, \Theta_S | Z, U, \hat{S})
\]

\[
= \log p(Z | S, \Theta_Z) + \log p(S | U, \Theta_S) + \log p(S | \hat{S}) + \log p(\Theta_Z) + \log p(\Theta_S) + \text{const}
\]

- Can take noisy labels into account
Slotcar Testbed

- Track contains banked curves, elevation changes
- Custom IMU with gyros and accelerometers built by Intel Research Seattle
- Observations very noisy, perceptual aliasing
Predictive Capability

- Latent space dimensionality: GPBFL 3D, HSE-HMM 20D
- HSE-HMM [Song-etal: ICML-10] much more efficient
- GPBFL optimization can incorporate noisy labels
Simple Trajectory Replay

- **Learning**
  - Human demonstrates control
  - Learn latent states using GPBF-Learn
  - Learn mapping from state to control

- **Replay**
  - Track state using GP-BayesFilter
  - Use control given by control GP
Trajectory Replay
Time Alignment

- 1d latent position vs. ground truth track position
- Blue indicates GPBFL alignment (multiple laps)
Learning from Noisy State Labels

- 1d latent space indicates car position on track
- Learn position of car when initialized with noisy weak labels
- Shading indicates control values (darker is stronger)
Comparison with Subspace Identification

- Simulated system with 1D observation and 1D control input
- Prediction errors
  - N4SID [Overschee-94]: 8.08
  - KCCA subspace identification method [Kawahara-etal: NIPS-07]: 6.42
  - GPBF-Learn: 2.09
ACT Hand Control

1. Investigation of muscle-joint kinematical relationship
2. How to control joints with muscles?
ACT Hand Tendon Arrangements

- Tendon hood structure for extensors
  - Critical for preserving hand functionality
  - Slides over the bones and joints

- We have non-linear, non-constant relationships between muscles and joints
Determine Muscle-Joint Kinematics

- Move finger in its ranges of motion
  - Record joint angle and muscle excursion data
- Determine mappings using joint and muscle data
- Determine moment arm matrix using mapping functions

\[
l_j = f_j(\theta)
\]
\[
\dot{l} = R(\theta)\dot{\theta}
\]

\[
R_{ij} = \frac{\partial l_{mi}}{\partial \theta_j} = \frac{\partial f_i}{\partial \theta_j}
\]

[Deshpande et al, BioRob 08, J Biomch 09]
Force Optimized Joint Control

- Determine the desired joint torques
  - Desired joint positions
  - Finger dynamics
- Determine muscle forces

\[ \tau_{\text{joint}} = -R^T \cdot F_{\text{muscle}} \]

\[
\begin{align*}
\text{determine} & \quad \text{with} & \quad \text{min}(\sum |\tau_{\text{des}} + R^T \cdot F_m|) \\
\text{with constraints} & \quad 0 \leq F_m(i) \leq F_{\text{max}}(i)
\end{align*}
\]
Force Optimized Joint Control

- Better position tracking than with polynomial fit
- Motions are not smooth...
GP-Based Control
RL with GP Dynamics Models

• So far GPs for
  – filtering and prediction
  – subspace id
  – trajectory replay
  – hand control

• Now: Incorporation of GP models into RL
PILCO: Probabilistic Inference for Learning Control

- Model-based policy search to minimize given cost function
- Policy: mapping from state to control
- Rollout: plan using current policy and GP dynamics model
- Policy parameter update via CG/BFGS
- Highly data efficient

[Deisenroth-etal, ICML-11, RSS-11]
Model Learning and Approximate Inference

Gaussian Process Forward Model

Approximate Inference for Policy Learning

- Probabilistic GP model consistently describes model uncertainties
- Long-term planning requires approximate inference: moment matching
- Model uncertainties are integrated out analytically (opposed to MC [Bagnell-00])

Deisenroth-etal also introduced GP-ADFs and EP for smoothing in GP dynamical systems
Controlling a Low-Cost Robotic Manipulator

- **Low-cost** system ($500 for robot arm and Kinect)
- **Very noisy**
- No sensor information about robot’s joint configuration used
- **Goal:** Learn to stack tower of 5 blocks from scratch
- Kinect camera for tracking block in end-effector
- State: coordinates (3D) of block center (from Kinect camera)
- 4 controlled DoF
- 20 learning trials for stacking 5 blocks (5 seconds long each)
- Account for **system noise**, e.g.,
  - Robot arm
  - Image processing
Collision Avoidance

- Use valuable prior information about obstacles if available
- Incorporation into planning → penalize in cost function
Collision Avoidance Results

Experimental Setup

Training runs (during learning) with collisions

• Cautious learning and exploration (rather safe than risky-successful)
• Learning slightly slower, but with significantly fewer collisions during training
• Average collision reduction (during training): 32.5% → 0.5%
Summary

• GPs provide flexible modeling framework
• Take data noise and uncertainty due to data sparsity into account
• Seamless integration into Bayes filters
• Combination with parametric models increases accuracy and reduces amount of training data
• Subspace identification via extended GPLVMs
• Data efficient RL
• Computational complexity is a key problem
• Advances in GPs and GPLVMs can be leveraged