State Space Expectation Propagation

Efficient Inference Schemes for Temporal Gaussian Processes

William Wilkinson[*], Paul Chang[*], Michael Riis Andersen[†], Arno Solin[*]

Aalto University[*], Technical University of Denmark[†]

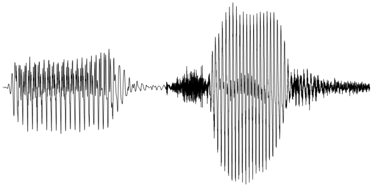
ICML 2020

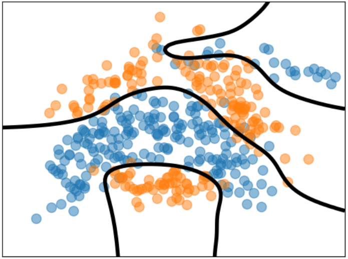
Aalto University

DTU

Motivation

- We're interested in long temporal and spatio-temporal data with interesting non-conjugate GP models (e.g. classification, log-Gaussian Cox processes).
- Idea: We should treat the temporal dimension in a fundamentally different manner to other dimensions.





Approximate Inference in Temporal GPs

There exists a dual kernel / SDE form for most popular Gaussian process (GP) models

$$
\begin{aligned}
f(t) &\sim \mathcal{GP}\big(0,\, K_\theta(t,t')\big), & \mathbf{f}_k &= \mathbf{A}_{\theta,k}\mathbf{f}_{k-1} + \mathbf{q}_k, & \mathbf{q}_k &\sim \mathrm{N}(\mathbf{0}, \mathbf{Q}_k) \\
y_k &\sim p(y_k \mid f(t_k)) & y_k &= h(\mathbf{f}_k, \boldsymbol{\sigma}_k), & \boldsymbol{\sigma}_k &\sim \mathrm{N}(0, \boldsymbol{\Sigma}_k)
\end{aligned}
$$

Approximate Inference in Temporal GPs

There exists a dual kernel / SDE form for most popular Gaussian process (GP) models

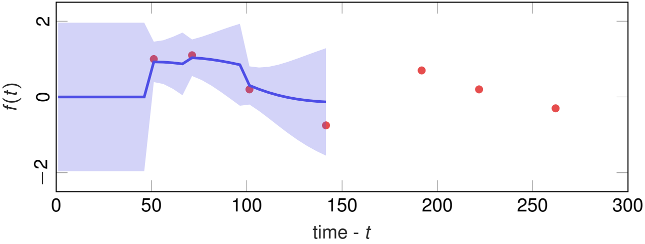
$$
\begin{aligned}
f(t) &\sim \mathcal{GP}\big(0,\, K_\theta(t,t')\big), & \mathbf{f}_k &= \mathbf{A}_{\theta,k}\mathbf{f}_{k-1} + \mathbf{q}_k, & \mathbf{q}_k &\sim \mathrm{N}(\mathbf{0}, \mathbf{Q}_k) \\
y_k &\sim p(y_k \mid f(t_k)) & y_k &= h(\mathbf{f}_k, \boldsymbol{\sigma}_k), & \boldsymbol{\sigma}_k &\sim \mathrm{N}(0, \boldsymbol{\Sigma}_k)
\end{aligned}
$$

inference in $\mathcal{O}(n)$ via Kalman filtering and smoothing

Approximate Inference

Kalman filter update step:

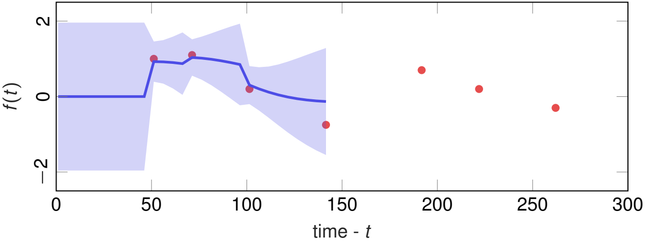
$$p(\mathbf{f}_k|y_{1:k}) \propto \mathrm{N}(\mathbf{m}_k^{\text{predict}}, \mathbf{P}_k^{\text{predict}})\, p(y_k \mid f(t_k))$$



Approximate Inference

Kalman filter update step:

$$p(\mathbf{f}_k | y_{1:k}) \propto \mathrm{N}(\mathbf{m}_k^{\text{predict}}, \mathbf{P}_k^{\text{predict}})\, p(y_k \mid f(t_k))$$
$$\approx \mathrm{N}(\mathbf{m}_k^{\text{predict}}, \mathbf{P}_k^{\text{predict}})\, \underbrace{\mathrm{N}(\mathbf{m}_k^{\text{site}}, \mathbf{P}_k^{\text{site}})}_{\text{"site"}}$$

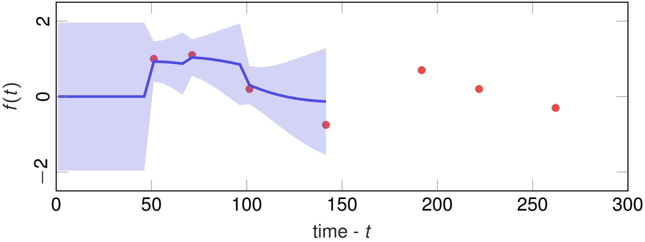


Approximate Inference

Kalman filter update step:

$$p(\mathbf{f}_k | y_{1:k}) \propto \mathrm{N}(\mathbf{m}_k^{\text{predict}}, \mathbf{P}_k^{\text{predict}})\, p(y_k \mid f(t_k))$$
$$\approx \mathrm{N}(\mathbf{m}_k^{\text{predict}}, \mathbf{P}_k^{\text{predict}})\, \mathrm{N}(\mathbf{m}_k^{\text{site}}, \mathbf{P}_k^{\text{site}})$$

Approx. Inference:
← select parameters

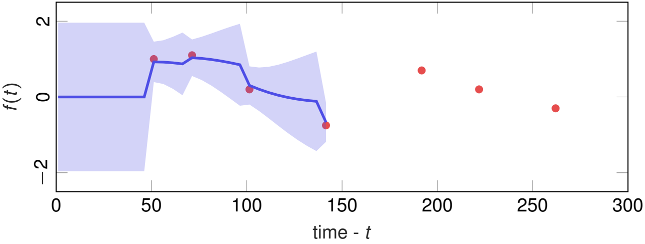


Approximate Inference

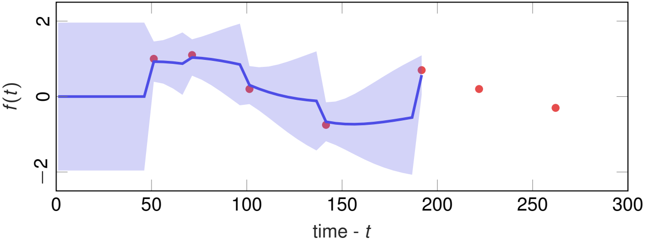
Kalman filter update step:

$$p(\mathbf{f}_k|y_{1:k}) \propto \mathrm{N}(\mathbf{m}_k^{\text{predict}}, \mathbf{P}_k^{\text{predict}})\, p(y_k \mid f(t_k))$$
$$\approx \mathrm{N}(\mathbf{m}_k^{\text{predict}}, \mathbf{P}_k^{\text{predict}})\, \mathrm{N}(\mathbf{m}_k^{\text{site}}, \mathbf{P}_k^{\text{site}})$$

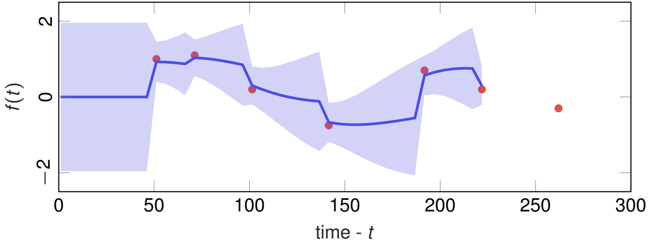
Approx. Inference:
select parameters



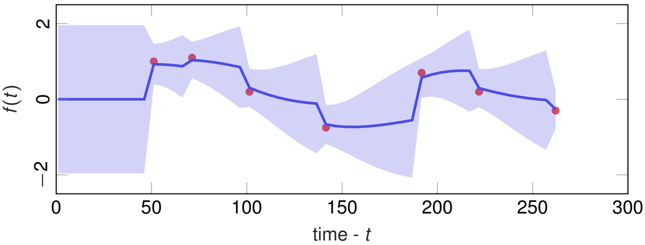
Approximate Inference



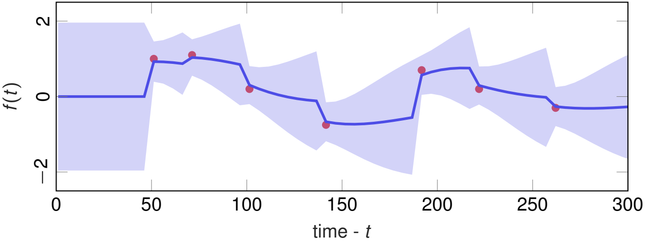
Approximate Inference



Approximate Inference



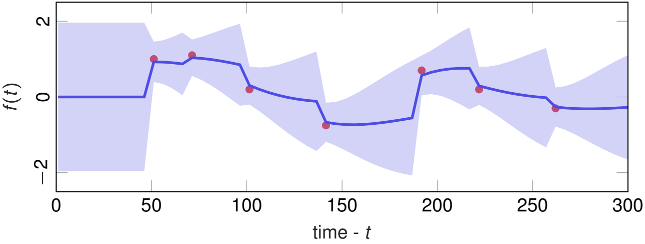
Approximate Inference



Approximate Inference

Smoothing:

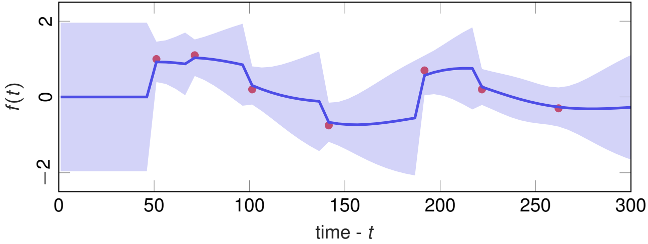
- update posterior with future observations, $p(\mathbf{f}_k \mid y_{1:N}) = \mathrm{N}(\mathbf{m}_k^{\text{post.}}, \mathbf{P}_k^{\text{post.}})$



Approximate Inference

Smoothing:

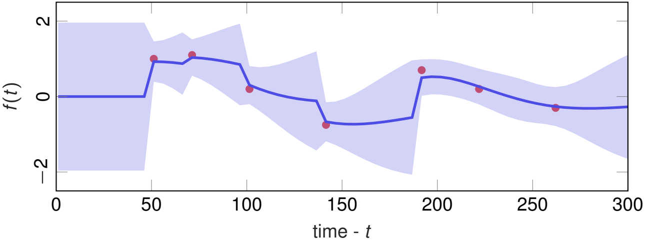
- update posterior with future observations, $p(\mathbf{f}_k \mid y_{1:N}) = \mathrm{N}(\mathbf{m}_k^{\text{post.}}, \mathbf{P}_k^{\text{post.}})$



Approximate Inference

Smoothing:

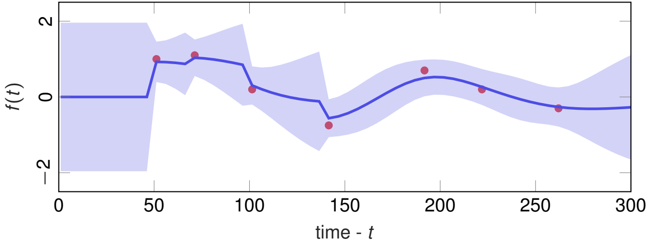
- update posterior with future observations, $p(\mathbf{f}_k \mid y_{1:N}) = \mathrm{N}(\mathbf{m}_k^{\text{post.}}, \mathbf{P}_k^{\text{post.}})$



Approximate Inference

Smoothing:

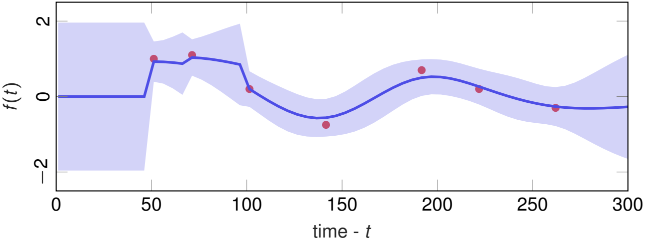
- update posterior with future observations, $p(\mathbf{f}_k \mid y_{1:N}) = \mathrm{N}(\mathbf{m}_k^{\text{post.}}, \mathbf{P}_k^{\text{post.}})$



Approximate Inference

Smoothing:

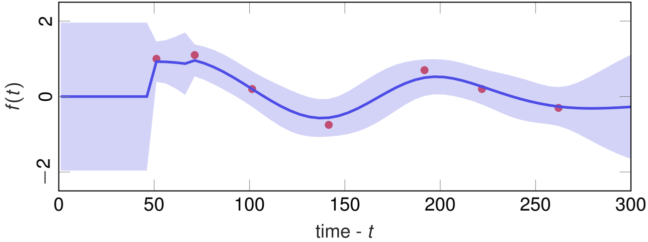
- update posterior with future observations, $p(\mathbf{f}_k \mid y_{1:N}) = \mathrm{N}(\mathbf{m}_k^{\text{post.}}, \mathbf{P}_k^{\text{post.}})$



Approximate Inference

Smoothing:

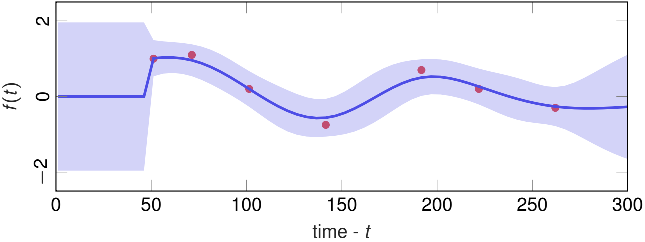
- update posterior with future observations,
  $p(\mathbf{f}_k \mid y_{1:N}) = \mathrm{N}(\mathbf{m}_k^{\text{post.}}, \mathbf{P}_k^{\text{post.}})$



Approximate Inference

Smoothing:

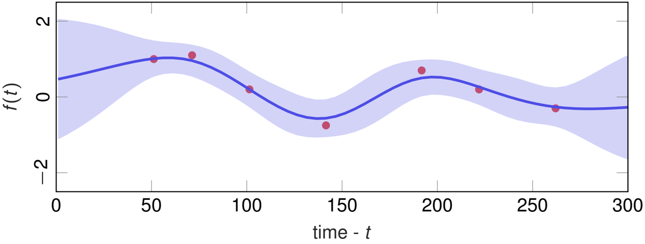
- update posterior with future observations,
  $p(\mathbf{f}_k \mid y_{1:N}) = \mathrm{N}(\mathbf{m}_k^{\text{post.}}, \mathbf{P}_k^{\text{post.}})$



Approximate Inference

Smoothing:

- update posterior with future observations, $p(\mathbf{f}_k \mid y_{1:N}) = \mathrm{N}(\mathbf{m}_k^{\text{post.}}, \mathbf{P}_k^{\text{post.}})$



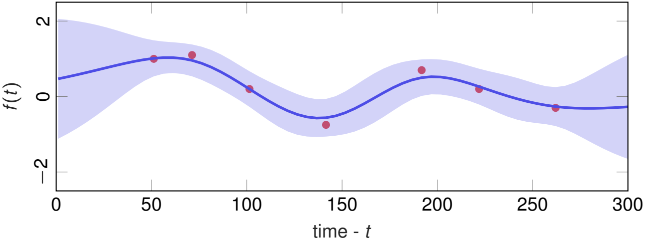
Approximate Inference

Smoothing:

- update posterior with future observations, $p(\mathbf{f}_k \mid y_{1:N}) = \mathrm{N}(\mathbf{m}_k^{\text{post.}}, \mathbf{P}_k^{\text{post.}})$

Our Contribution:

Given marginal posterior $\mathrm{N}(\mathbf{m}_k^{\text{post.}}, \mathbf{P}_k^{\text{post.}})$, we show how approximate inference amounts to a simple site parameter update rule during smoothing.



Approximate Inference

Smoothing:

- update posterior with future observations, $p(\mathbf{f}_k \mid y_{1:N}) = \mathrm{N}(\mathbf{m}_k^{\text{post.}}, \mathbf{P}_k^{\text{post.}})$

Our Contribution:

Given marginal posterior $\mathrm{N}(\mathbf{m}_k^{\text{post.}}, \mathbf{P}_k^{\text{post.}})$, we show how approximate inference amounts to a simple site parameter update rule during smoothing.

This encompasses:

- Power Expectation Propagation
- Variational Inference (with natural gradients)
- Extended Kalman Smoothing
- Unscented / Gauss-Hermite Kalman Smoothing
- Posterior Linearisation

Parameter Update Rules

for $\nabla \mathcal{L}_k = \frac{\mathrm{d}\mathcal{L}_k}{\mathrm{d}\mathbf{m}_k}$

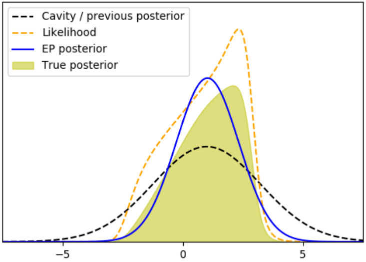
Power Expectation Propagation:

$$q_{\text{cavity}}(\mathbf{f}_k) = q_{\text{post.}}(\mathbf{f}_k) / q^{\alpha}_{\text{site}}(\mathbf{f}_k)$$

$$\mathcal{L}_k = \log \mathbb{E}_{q_{\text{cavity}}}\left[p^{\alpha}(\mathbf{y}_k \mid \mathbf{f}_k)\right]$$

$$\mathbf{P}^{\text{site}}_k = -\alpha\left(\mathbf{P}^{\text{cavity}}_k + \left(\nabla^2 \mathcal{L}_k\right)^{-1}\right)$$

$$\mathbf{m}^{\text{site}}_k = \mathbf{m}^{\text{cavity}}_k - \left(\nabla^2 \mathcal{L}_k\right)^{-1} \nabla \mathcal{L}_k$$



Parameter Update Rules

for $\nabla\mathcal{L}_k = \frac{\mathrm{d}\mathcal{L}_k}{\mathrm{d}\mathbf{m}_k}$

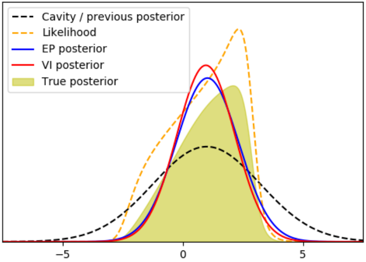
Power Expectation Propagation:

$$q_{\text{cavity}}(\mathbf{f}_k) = q_{\text{post.}}(\mathbf{f}_k)/q^{\alpha}_{\text{site}}(\mathbf{f}_k)$$

$$\begin{aligned}
\mathcal{L}_k &= \log \mathbb{E}_{q_{\text{cavity}}}\left[p^{\alpha}(\mathbf{y}_k \mid \mathbf{f}_k)\right] \\
\mathbf{P}^{\text{site}}_k &= -\alpha\left(\mathbf{P}^{\text{cavity}}_k + (\nabla^2\mathcal{L}_k)^{-1}\right) \\
\mathbf{m}^{\text{site}}_k &= \mathbf{m}^{\text{cavity}}_k - (\nabla^2\mathcal{L}_k)^{-1}\nabla\mathcal{L}_k
\end{aligned}$$

Variational Inference:

$$\begin{aligned}
\mathcal{L}_k &= \mathbb{E}_{q_{\text{post.}}}\left[\log p(\mathbf{y}_k \mid \mathbf{f}_k)\right] \\
\mathbf{P}^{\text{site}}_k &= -(\nabla^2\mathcal{L}_k)^{-1} \\
\mathbf{m}^{\text{site}}_k &= \mathbf{m}^{\text{post.}}_k - (\nabla^2\mathcal{L}_k)^{-1}\nabla\mathcal{L}_k
\end{aligned}$$



Parameter Update Rules

for $\nabla\mathcal{L}_k = \frac{\mathrm{d}\mathcal{L}_k}{\mathrm{d}\mathbf{m}_k}$

Power Expectation Propagation:

$$q_{\text{cavity}}(\mathbf{f}_k) = q_{\text{post.}}(\mathbf{f}_k)/q^{\alpha}_{\text{site}}(\mathbf{f}_k)$$

$$\mathcal{L}_k = \log \mathbb{E}_{q_{\text{cavity}}}\left[p^{\alpha}(\mathbf{y}_k \mid \mathbf{f}_k)\right]$$

$$\mathbf{P}^{\text{site}}_k = -\alpha\left(\mathbf{P}^{\text{cavity}}_k + (\nabla^2\mathcal{L}_k)^{-1}\right)$$

$$\mathbf{m}^{\text{site}}_k = \mathbf{m}^{\text{cavity}}_k - (\nabla^2\mathcal{L}_k)^{-1}\,\nabla\mathcal{L}_k$$

Variational Inference:

$$\mathcal{L}_k = \mathbb{E}_{q_{\text{post.}}}\left[\log p(\mathbf{y}_k \mid \mathbf{f}_k)\right]$$

$$\mathbf{P}^{\text{site}}_k = -(\nabla^2\mathcal{L}_k)^{-1}$$

$$\mathbf{m}^{\text{site}}_k = \mathbf{m}^{\text{post.}}_k - (\nabla^2\mathcal{L}_k)^{-1}\,\nabla\mathcal{L}_k$$

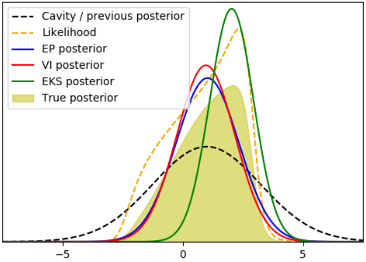
Extended Kalman Smoother:

$$\mathbf{v}_k = \mathbf{y}_k - \mathbf{h}(\mathbf{m}^{\text{post.}}_k, \mathbf{0})$$

$$\mathbf{S}_k = \mathbf{H}_{\mathbf{f}}^{\top}\mathbf{P}^{\text{post.}}_k\mathbf{H}_{\mathbf{f}} + \mathbf{H}_{\boldsymbol{\sigma}}\Sigma_k\mathbf{H}_{\boldsymbol{\sigma}}^{\top}$$

$$\mathbf{P}^{\text{site}}_k = \left(\mathbf{H}_{\mathbf{f}}^{\top}\left(\mathbf{H}_{\boldsymbol{\sigma}}\Sigma_k\mathbf{H}_{\boldsymbol{\sigma}}^{\top}\right)^{-1}\mathbf{H}_{\mathbf{f}}\right)^{-1}$$

$$\mathbf{m}^{\text{site}}_k = \mathbf{m}^{\text{post.}}_k + (\mathbf{P}^{\text{site}}_k + \mathbf{P}^{\text{post.}}_k)\mathbf{H}_{\mathbf{f}}^{\top}\mathbf{S}_k^{-1}\mathbf{v}_k$$



for $\mathbf{H}_{\mathbf{f}} = \frac{\mathrm{d}\mathbf{h}}{\mathrm{d}\mathbf{f}}$ and $\mathbf{H}_{\boldsymbol{\sigma}} = \frac{\mathrm{d}\mathbf{h}}{\mathrm{d}\boldsymbol{\sigma}}$, $\boldsymbol{\sigma}_k \sim \mathrm{N}(0, \Sigma_k)$

A Unifying Perspective

- For sequential data, the EKF / UKF / GHKF are equivalent to single-sweep EP where the moment matching is solved via linearisation.

A Unifying Perspective

- For sequential data, the EKF / UKF / GHKF are equivalent to single-sweep EP where the moment matching is solved via linearisation.
- The iterated Kalman smoothers (EKS / UKS / GHKS) can also be recovered under certain parameter choices. But note that they optimise a different objective to EP (see paper for details).

A Unifying Perspective

- For sequential data, the EKF / UKF / GHKF are equivalent to single-sweep EP where the moment matching is solved via linearisation.
- The iterated Kalman smoothers (EKS / UKS / GHKS) can also be recovered under certain parameter choices. But note that they optimise a different objective to EP (see paper for details).
- We show how natural gradient VI updates are surprisingly similar to the EP updates (when using a similar parametrisation).

New Algorithms

- We propose to mix the beneficial properties of EP with the efficiency of classical smoothers.

New Algorithms

- We propose to mix the beneficial properties of EP with the efficiency of classical smoothers.

- For example, using linearisation to speed up the updates, whilst also introducing the EP cavity and fractional updates.

New Algorithms

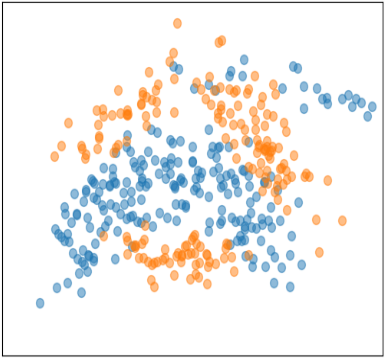
- We propose to mix the beneficial properties of EP with the efficiency of classical smoothers.
- For example, using linearisation to speed up the updates, whilst also introducing the EP cavity and fractional updates.
- We call this Extended Kalman Expectation Propagation (EK-EP).

New Algorithms

- We propose to mix the beneficial properties of EP with the efficiency of classical smoothers.
- For example, using linearisation to speed up the updates, whilst also introducing the EP cavity and fractional updates.
- We call this Extended Kalman Expectation Propagation (EK-EP).
- It has clear computational benefits when the parameter updates are high-dimensional, *e.g.*, in spatio-temporal problems.

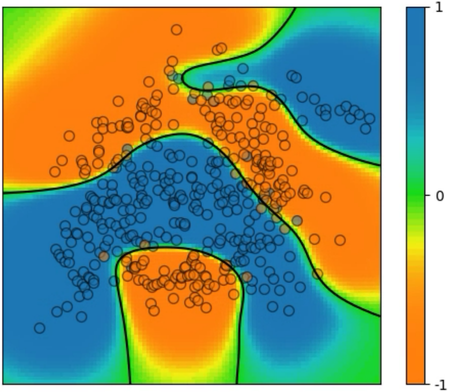
Spatio-Temporal Classification

- We show that our smoothing methods can be applied to tasks with more than one input dimension.



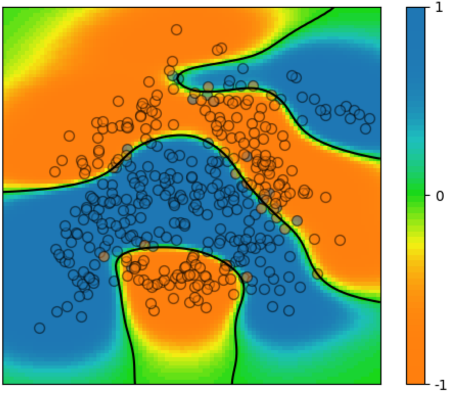
Spatio-Temporal Classification

- We show that our smoothing methods can be applied to tasks with more than one input dimension.



Spatio-Temporal Classification

- We show that our smoothing methods can be applied to tasks with more than one input dimension.



We treat the first dimension (x-axis) as time, and run iterated spatio-temporal smoothing (this demo uses EP).

Fast Learning Using JAX

- Temporal GP methods have been limited by a lack of appropriate software for hyperparameter learning:

Automatic differentiation + massive for loops

https://github.com/AaltoML/kalman-jax

Fast Learning Using JAX

- Temporal GP methods have been limited by a lack of appropriate software for hyperparameter learning:

Automatic differentiation + massive for loops

- We provide a temporal GP framework in JAX with all inference methods implemented.

https://github.com/AaltoML/kalman-jax

Fast Learning Using JAX

- Temporal GP methods have been limited by a lack of appropriate software for hyperparameter learning:

  Automatic differentiation + massive for loops

- We provide a temporal GP framework in JAX with all inference methods implemented.

  i) avoid loop "unrolling" to reduce compilation overheads

`https://github.com/AaltoML/kalman-jax`

Fast Learning Using JAX

- Temporal GP methods have been limited by a lack of appropriate software for hyperparameter learning:

  Automatic differentiation + massive for loops

- We provide a temporal GP framework in JAX with all inference methods implemented.

  i) avoid loop "unrolling" to reduce compilation overheads

  ii) JIT compilation to avoid graph retracing

`https://github.com/AaltoML/kalman-jax`

Fast Learning Using JAX

- Temporal GP methods have been limited by a lack of appropriate software for hyperparameter learning:

Automatic differentiation + massive for loops

- We provide a temporal GP framework in JAX with all inference methods implemented.

  i) avoid loop "unrolling" to reduce compilation overheads

  ii) JIT compilation to avoid graph retracing

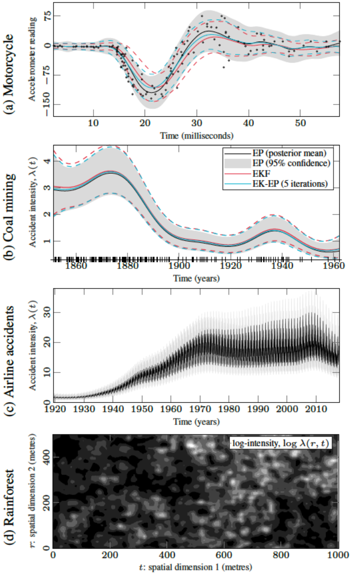
  iii) Exploits accelerated linear algebra (XLA) ops

`https://github.com/AaltoML/kalman-jax`

Results

We run extensive analysis on synthetic and real world data:

- Heteroscedastic Noise
- 1D & 2D Log Gaussian Cox Process
- 1D & 2D Classification
- Audio Amplitude Demodulation



Results

- No consistently best inference method or EP power value:
  - EK-EP the only practical method when updates are high dimensional (rainforest)
  - EP or VI needed when likelihood is highly nonlinear

Results

- No consistently best inference method or EP power value:
  - EK-EP the only practical method when updates are high dimensional (rainforest)
  - EP or VI needed when likelihood is highly nonlinear

- We compare against non-sequential baselines (SVGP and EP).
  - Sequential learning methods match the performance of batch methods, whilst scaling to larger data.

Results

- No consistently best inference method or EP power value:
  - EK-EP the only practical method when updates are high dimensional (rainforest)
  - EP or VI needed when likelihood is highly nonlinear
- We compare against non-sequential baselines (SVGP and EP).
  - Sequential learning methods match the performance of batch methods, whilst scaling to larger data.
- See the paper for full results table.

Thanks for Listening

Take home messages:

- Any approximate inference method can be framed as a simple parameter update rule during Kalman smoothing.
- Sequential methods match the performance of batch methods, and can be extended to multiple dimensions.
- We provide fast JAX code for all methods.

Contact: `william.wilkinson@aalto.fi`

JAX code: `https://github.com/AaltoML/kalman-jax`