Gaussian Process Classification, Approximations and Other GP Models

COMP9418 — Advanced Topics in Statistical Machine Learning

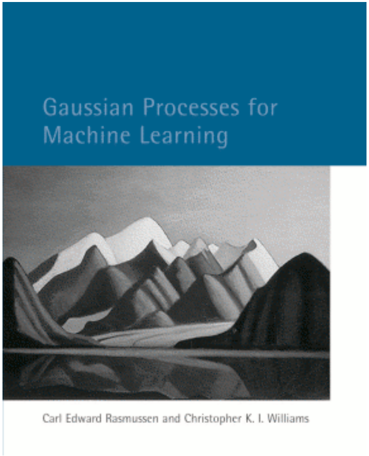
Edwin V. Bonilla

School of Computer Science and Engineering
UNSW Sydney

October 4th, 2017

(Last Update: Tuesday 3rd October, 2017 at 11:41)

Acknowledgements



Carl Edward Rasmussen and Christopher K. I. Williams

All chapters available online along with software and datasets:
`http://www.gaussianprocess.org/gpml`

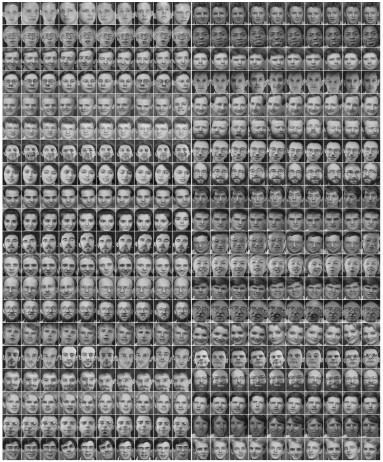
Aims

This lecture will allow you to understand and apply Gaussian process classification and approximations methods to large datasets. Following it you should be to:

- Understand and apply Gaussian process binary classification with posterior inference via the Laplace approximation.
- Understand and apply scalable approaches to Gaussian process regression from a unifying probabilistic framework.
- Understand other models with Gaussian process priors and non-linear likelihoods and their applications such as multi-task learning (MTGP) and latent-variable models (GPLVM).

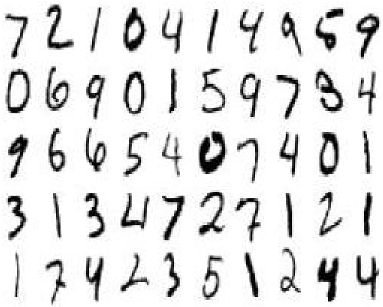
Classification Problems

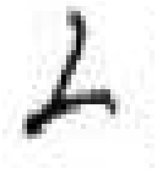
Facial Recognition


Examples of faces and their identity.

what is his/her identity?

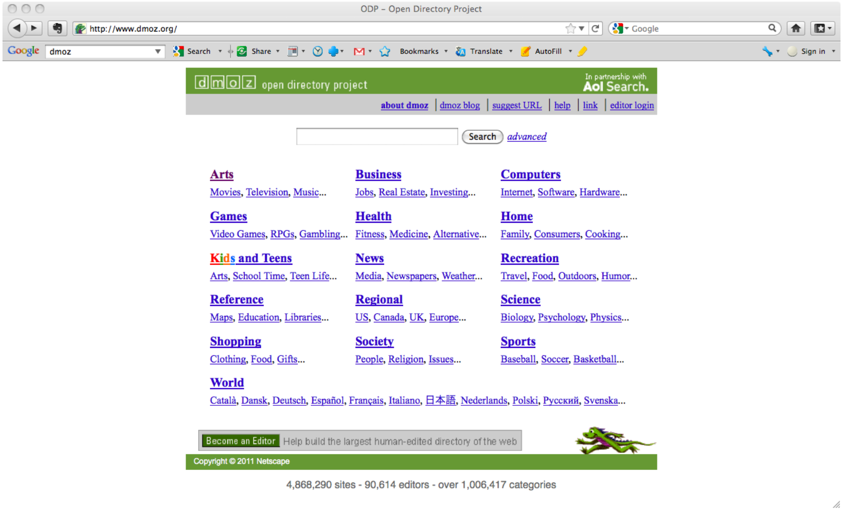
Classification Problems
Handwritten Digit Recognition


Examples of images and their corresponding digit.


What is the number in the image?

Classification Problems

Supervised Document Classification


ODP – Open Directory Project
http://www.dmoz.org/
Google
dmoz
Search
Share
Bookmarks
Translate
AutoFill
Sign in
dmoz open directory project
In partnership with Aol Search.
about dmoz | dmoz blog | suggest URL | help | link | editor login
Search
advanced

Arts
Movies, Television, Music...

Business
Jobs, Real Estate, Investing...

Computers
Internet, Software, Hardware...

Games
Video Games, RPGs, Gambling...

Health
Fitness, Medicine, Alternative...

Home
Family, Consumers, Cooking...

Kids and Teens
Arts, School Time, Teen Life...

News
Media, Newspapers, Weather...

Recreation
Travel, Food, Outdoors, Humor...

Reference
Maps, Education, Libraries...

Regional
US, Canada, UK, Europe...

Science
Biology, Psychology, Physics...

Shopping
Clothing, Food, Gifts...

Society
People, Religion, Issues...

Sports
Baseball, Soccer, Basketball...

World
Català, Dansk, Deutsch, Español, Français, Italiano, 日本語, Nederlands, Polski, Русский, Svenska...

Become an Editor Help build the largest human-edited directory of the web

Copyright © 2011 Netscape

4,868,290 sites - 90,614 editors - over 1,006,417 categories

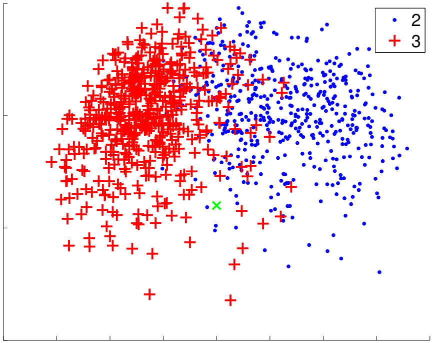
The Classification Problem

Problem Definition

In all previous problems we are dealing with **discrete** targets.

Given a set of input-output pairs $\mathcal{D} = \{\mathbf{x}^{(n)}, y^{(n)}\}_{n=1}^{N}$ where $\mathbf{x}$ is a D-dimensional feature vector and $y \in \{\mathcal{C}_1, \ldots, \mathcal{C}_M\}$.

Goal: Learn a mapping $f(\mathbf{x}) : \mathbf{x} \rightarrow y$ in order to make predictions at unseen datapoints $\mathbf{x}^*$.



The Classification Problem
Some Concepts

Distinguishing between 2s and 3s is an example of **binary classification** as we only have two classes.

The Classification Problem
Some Concepts

Distinguishing between 2s and 3s is an example of **binary classification** as we only have two classes.

Discriminating between the digits $0, 1, \ldots, 9$ is an example of a **multi-class** problem as we have more than two classes.

The Classification Problem

Some Concepts

Distinguishing between 2s and 3s is an example of **binary classification** as we only have two classes.

Discriminating between the digits $0, 1, \ldots, 9$ is an example of a **multi-class** problem as we have more than two classes.

Probabilistic methods: It is general useful to have some confidence on our predictions, e.g. for a *reject* option. However, some popular methods are inherently non-probabilistic.

The Classification Problem
Some Concepts

Distinguishing between 2s and 3s is an example of **binary classification** as we only have two classes.

Discriminating between the digits $0, 1, \ldots, 9$ is an example of a **multi-class** problem as we have more than two classes.

Probabilistic methods: It is general useful to have some confidence on our predictions, e.g. for a *reject* option. However, some popular methods are inherently non-probabilistic.

Encoding: We may need to encode the targets (for ease of computation)

The Classification Problem
Some Concepts

Distinguishing between 2s and 3s is an example of **binary classification** as we only have two classes.

Discriminating between the digits $0, 1, \ldots, 9$ is an example of a **multi-class** problem as we have more than two classes.

Probabilistic methods: It is general useful to have some confidence on our predictions, e.g. for a *reject* option. However, some popular methods are inherently non-probabilistic.

Encoding: We may need to encode the targets (for ease of computation)

- Binary classification: $\{0, 1\}$, $\{-1, 1\}$

The Classification Problem
Some Concepts

Distinguishing between 2s and 3s is an example of **binary classification** as we only have two classes.

Discriminating between the digits $0, 1, \ldots, 9$ is an example of a **multi-class** problem as we have more than two classes.

Probabilistic methods: It is general useful to have some confidence on our predictions, e.g. for a *reject* option. However, some popular methods are inherently non-probabilistic.

Encoding: We may need to encode the targets (for ease of computation)

- Binary classification: $\{0, 1\}$, $\{-1, 1\}$
- Multiclass classification: 1-of-M encoding
  e.g. $y = (0, 0, 0, 1, 0)^T$ denoting the instance belongs to class 4 (out of $M = 5$ classes)

Discriminative vs Generative Approaches

Generative

Model the joint $p(\mathbf{x}, y)$ via models for $p(\mathbf{x}|y)$ and $p(y)$.

Predictions $p(y|\mathbf{x})$ via Bayes rule

+ Amenable to incorporation of prior knowledge

– Indirect approach

Discriminative

Model $p(y|\mathbf{x})$ directly.

– Difficult to incorporate prior information

+ Focus on the task at hand

Decision Theory for Classification

A probabilistic classifier provides an elegant framework for decision theory:

Suppose that we have a predictive probability $p(y = c|\mathbf{x})$

Let $\underline{L}(c, c')$ be the loss incurred by making a decision c' when the true class is c.

Decision Theory for Classification

A probabilistic classifier provides an elegant framework for decision theory:

Suppose that we have a predictive probability $p(y = c|\mathbf{x})$

Let $\underline{L}(c, c')$ be the loss incurred by making a decision c' when the true class is c.

Then we predict c^* that solves:

$$c^* = \operatorname*{argmin}_{c'} \mathcal{R}(c' \mid \mathbf{x}_*) = \sum_c \underline{L}(c, c') p(c|\mathbf{x}_*)$$

Decision Theory for Classification

A probabilistic classifier provides an elegant framework for decision theory:

Suppose that we have a predictive probability $p(y = c|\mathbf{x})$

Let $\underline{L}(c, c')$ be the loss incurred by making a decision c' when the true class is c.

Then we predict c^* that solves:

$$c^* = \operatorname*{argmin}_{c'} \mathcal{R}(c' \mid \mathbf{x}_*) = \sum_{c} \underline{L}(c, c') p(c|\mathbf{x}_*)$$

For the case of the 0-1 loss (a unit penalty is paid for a misclassification), the optimal decision maximizes $p(c|\mathbf{x}_*)$

This optimal classifier is know as **Bayes classifier**.

Outline

Linear Models for Classification

Data : $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, $\mathbf{x} \in \mathbb{R}^{D}$, $y \in \{-1, +1\}$

Input : $(\mathbf{X})_{D \times N}$, Targets: $(\mathbf{y})_{N \times 1}$

Goal : Make predictions at $\mathbf{x}_*$

Model : $p(y = +1|\mathbf{X}, \mathbf{w}) = \sigma(\mathbf{w}^T \mathbf{x})$

Linear Models for Classification

Data : $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, $\mathbf{x} \in \mathbb{R}^D$, $y \in \{-1, +1\}$

Input : $(\mathbf{X})_{D \times N}$, Targets: $(\mathbf{y})_{N \times 1}$

Goal : Make predictions at $\mathbf{x}_*$

Model : $p(y = +1 | \mathbf{X}, \mathbf{w}) = \sigma(\mathbf{w}^T \mathbf{x})$

Two popular approaches:

- Logistic Regression $\sigma(z) = \dfrac{1}{1 + \exp(-z)}$

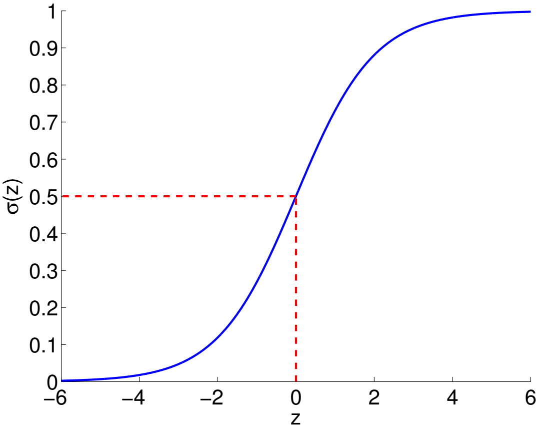
Linear Models for Classification

$$\begin{aligned}
\text{Data} &: \mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N},\ \mathbf{x} \in \mathbb{R}^D,\ y \in \{-1, +1\} \\
\text{Input} &: (\mathbf{X})_{D \times N},\ \text{Targets: } (\mathbf{y})_{N \times 1} \\
\text{Goal} &: \text{Make predictions at } \mathbf{x}_* \\
\text{Model} &: p(y = +1 | \mathbf{X}, \mathbf{w}) = \sigma(\mathbf{w}^T \mathbf{x})
\end{aligned}$$

Two popular approaches:

- Logistic Regression $\sigma(z) = \dfrac{1}{1 + \exp(-z)}$
- Probit Regression: $\sigma(z) = \int_{-\infty}^{z} \mathcal{N}(x|0, 1) dx$

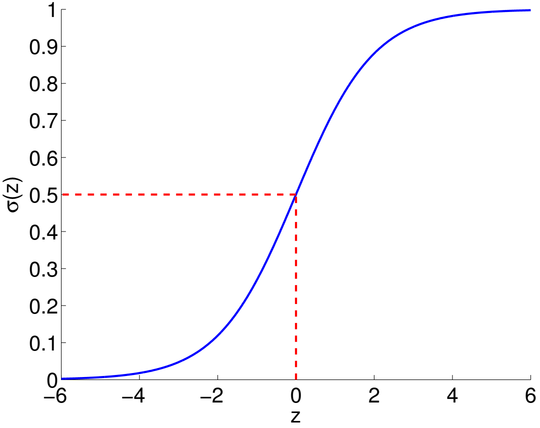
Logistic Regression
The Logit (or Logistic Sigmoid) Function



So $p(y = +1|\mathbf{x}) > p(y = -1|\mathbf{x})$ when $\mathbf{w}^T\mathbf{x} > 0$. We have a **linear decision boundary**.

Logistic Regression

The Logit (or Logistic Sigmoid) Function



So $p(y = +1|\mathbf{x}) > p(y = -1|\mathbf{x})$ when $\mathbf{w}^T\mathbf{x} > 0$. We have a **linear decision boundary**.

How do we learn the weights?

MAP Approach

As in Bayesian linear regression we can use the prior:

$$\mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{0}, \boldsymbol{\Sigma}_w).$$

MAP Approach

As in Bayesian linear regression we can use the prior:

$$\mathbf{w} \sim \mathcal{N}(\mathbf{w} | \mathbf{0}, \mathbf{\Sigma}_w).$$

However, the full posterior does not have a simple analytical form. We write down the un-normalized log-posterior:

$$\mathcal{L}^{\text{MAP}} = \sum_{i=1}^{N} \log \sigma(y^{(i)} f_i) - \frac{1}{2} \mathbf{w}^T \mathbf{\Sigma}_w^{-1} \mathbf{w},$$

Where $f_i \overset{\text{def}}{=} \mathbf{w}^T \mathbf{x}_i$.

MAP Approach

As in Bayesian linear regression we can use the prior:

$$\mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{0}, \mathbf{\Sigma}_w).$$

However, the full posterior does not have a simple analytical form. We write down the un-normalized log-posterior:

$$\mathcal{L}^{\text{MAP}} = \sum_{i=1}^{N} \log \sigma(y^{(i)} f_i) - \frac{1}{2}\mathbf{w}^T \mathbf{\Sigma}_w^{-1} \mathbf{w},$$

Where $f_i \stackrel{\text{def}}{=} \mathbf{w}^T \mathbf{x}_i$. This objective function is concave and finding its maximum is "easy", e.g. using Newton's method, so called IRLS (iterative reweighted least squares)

MAP Approach

As in Bayesian linear regression we can use the prior:

$$\mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{0}, \mathbf{\Sigma}_w).$$

However, the full posterior does not have a simple analytical form. We write down the un-normalized log-posterior:

$$\mathcal{L}^{\text{MAP}} = \sum_{i=1}^{N} \log \sigma(y^{(i)} f_i) - \frac{1}{2}\mathbf{w}^T \mathbf{\Sigma}_w^{-1}\mathbf{w},$$

Where $f_i \overset{\text{def}}{=} \mathbf{w}^T \mathbf{x}_i$. This objective function is concave and finding its maximum is "easy", e.g. using Newton's method, so called IRLS (iterative reweighted least squares)

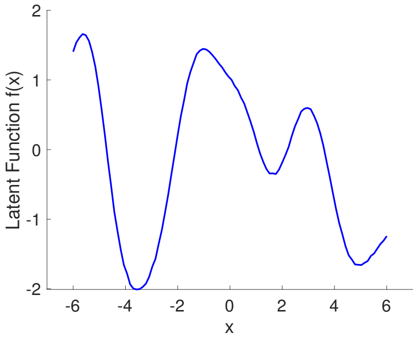
Multi-class case is addressed with a softmax function.

Gaussian Process Classification (GPC)

Binary Classification

Discriminative approach:

1. Place prior over the latent functions $f(\mathbf{x})$



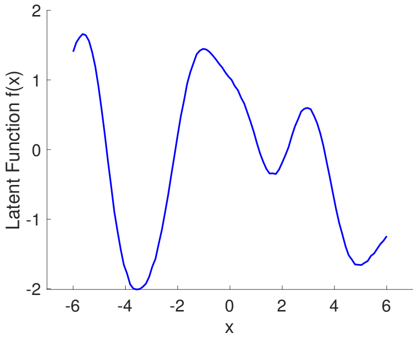
Sample from a GP

Gaussian Process Classification (GPC)

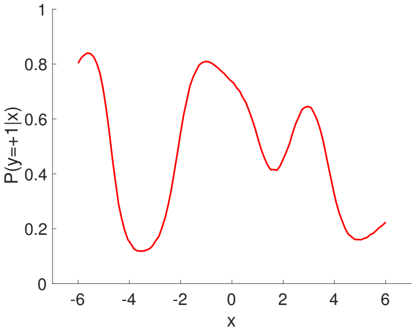
Binary Classification

Discriminative approach:

1. Place prior over the latent functions $f(\mathbf{x})$
2. Squash this through a sigmoid function: $p(y=+1|\mathbf{x}) = \sigma(f(\mathbf{x}))$



Sample from a GP



$\sigma(f(x)) = \frac{1}{1+e^{-f(x)}}$

GPC Model

Data : $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, $\mathbf{x} \in \mathbb{R}^{D}$, $y \in \{-1, +1\}$

Input : $(\mathbf{X})_{D \times N}$, Targets: $(\mathbf{y})_{N \times 1}$

GPC Model

Data : $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, $\mathbf{x} \in \mathbb{R}^D$, $y \in \{-1, +1\}$

Input : $(\mathbf{X})_{D \times N}$, Targets: $(\mathbf{y})_{N \times 1}$

Prior : $f \sim \mathcal{GP}\left(0, \kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta})\right) \rightarrow p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$,

Likelihood : $p(y = +1|f) = \sigma(f(x)) \overset{\text{iid}}{\rightarrow} p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{N} \sigma(y^{(i)} f_i)$,

- $(\mathbf{f})_{N \times 1}$ vector of latent variables
- $\mathbf{K} = \kappa(\mathbf{X}, \mathbf{X}; \boldsymbol{\theta})$

GPC Model

Data : $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, $\mathbf{x} \in \mathbb{R}^D$, $y \in \{-1, +1\}$

Input : $(\mathbf{X})_{D \times N}$, Targets: $(\mathbf{y})_{N \times 1}$

Prior : $f \sim \mathcal{GP}(0, \kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta})) \rightarrow p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$,

Likelihood : $p(y = +1|f) = \sigma(f(x)) \overset{\text{iid}}{\rightarrow} p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{N} \sigma(y^{(i)} f_i)$,

- $(\mathbf{f})_{N \times 1}$ vector of latent variables
- $\mathbf{K} = \kappa(\mathbf{X}, \mathbf{X}; \boldsymbol{\theta})$

Posterior $p(\mathbf{f}|\mathbf{X}, \mathbf{y}) \propto p(\mathbf{f}|\mathbf{X}) p(\mathbf{y}|\mathbf{f})$ analytically intractable

- Due to non-Gaussian likelihood $p(\mathbf{y}|\mathbf{f})$

Need to resort to approximations

GPC Inference

1. Compute predictive distribution of latent functions:

$$p(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{f}) p(\mathbf{f}|\mathbf{X}, \mathbf{y}) d\mathbf{f}$$

GPC Inference

1. Compute predictive distribution of latent functions:

$$p(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{f}) p(\mathbf{f}|\mathbf{X}, \mathbf{y}) d\mathbf{f}$$

   - Analytically intractable

GPC Inference

1. Compute predictive distribution of latent functions:

$$p(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{f}) p(\mathbf{f}|\mathbf{X}, \mathbf{y}) d\mathbf{f}$$

   - Analytically intractable

2. Compute probabilistic predictions:

$$p(y_* = +1|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int \sigma(f_*) p(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) df_*$$

GPC Inference

1. Compute predictive distribution of latent functions:

$$p(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{f}) p(\mathbf{f}|\mathbf{X}, \mathbf{y}) d\mathbf{f}$$

   - Analytically intractable

2. Compute probabilistic predictions:

$$p(y_* = +1|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int \sigma(f_*) p(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) df_*$$

   - Analytic solution for the probit model

GPC Inference

1. Compute predictive distribution of latent functions:

$$p(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{f})p(\mathbf{f}|\mathbf{X}, \mathbf{y})d\mathbf{f}$$

   - Analytically intractable

2. Compute probabilistic predictions:

$$p(y_* = +1|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int \sigma(f_*)p(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*)df_*$$

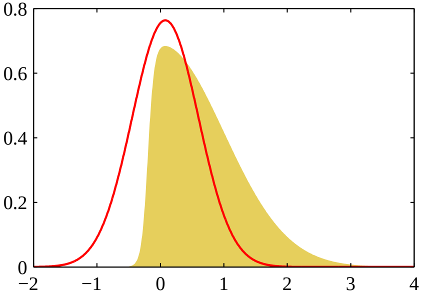
   - Analytic solution for the probit model
   - Require numerical approximations (1D) integral for other sigmoid functions

The Laplace Approximation

Idea: Find a Gaussian approximation to $p(z) = \frac{1}{Z} f(z)$, where Z is unknown. We centre the Gaussian approximation at the mode of $p(z)$.

The Laplace Approximation

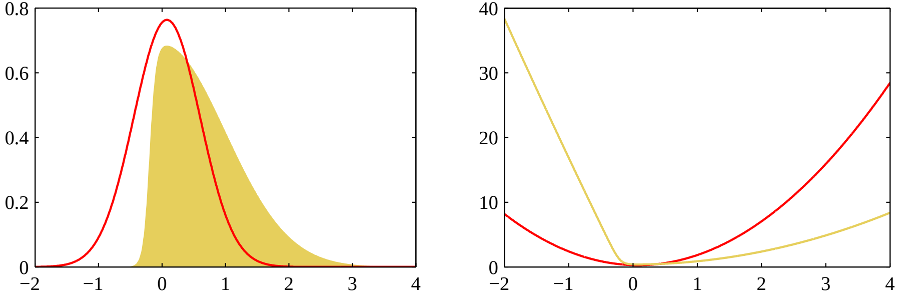
Idea: Find a Gaussian approximation to $p(z) = \frac{1}{Z} f(z)$, where Z is unknown. We centre the Gaussian approximation at the mode of $p(z)$.



Left : $p(z) \propto \exp(-z^2/2)\sigma(20z + 4)$ and corresponding Gaussian approximation.

The Laplace Approximation

Idea: Find a Gaussian approximation to $p(z) = \frac{1}{Z} f(z)$, where Z is unknown. We centre the Gaussian approximation at the mode of $p(z)$.



Figures by Christopher M. Bishop (MLPR, 2006)

Left : $p(z) \propto \exp(-z^2/2)\sigma(20z + 4)$ and corresponding Gaussian approximation.

Right : Negative logarithms of the corresponding curves.

The Laplace Approximation to the GP Binary Classifier

Gaussian approximation

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \boldsymbol{\theta}) \approx \mathcal{N}(\mathbf{f}|\hat{\mathbf{f}}, A^{-1})$$

where: $\hat{\mathbf{f}} = \text{argmax}_{\mathbf{f}}\, p(\mathbf{f}|\mathcal{D}, \boldsymbol{\theta}) = \text{argmax}_{\mathbf{f}}\;\; p(\mathbf{y}|\mathbf{f}, \boldsymbol{\theta}) p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta})$ and A is the Hessian of the negative log-posterior evaluated at $\hat{\mathbf{f}}$.

The Laplace Approximation to the GP Binary Classifier

Gaussian approximation

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \boldsymbol{\theta}) \approx \mathcal{N}(\mathbf{f}|\hat{\mathbf{f}}, A^{-1})$$

where: $\hat{\mathbf{f}} = \text{argmax}_{\mathbf{f}}\, p(\mathbf{f}|\mathcal{D}, \boldsymbol{\theta}) = \text{argmax}_{\mathbf{f}}\;\; p(\mathbf{y}|\mathbf{f}, \boldsymbol{\theta}) p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta})$ and A is the Hessian of the negative log-posterior evaluated at $\hat{\mathbf{f}}$.

Hence we focus on the maximization of:

$$\psi(\mathbf{f}) = \log p(\mathbf{y}|\mathbf{f}) - \frac{1}{2}\mathbf{f}^T\mathbf{K}^{-1}\mathbf{f} - \frac{1}{2}\log|\mathbf{K}| - \frac{N}{2}\log 2\pi$$

The Laplace Approximation to the GP Binary Classifier

Gaussian approximation

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \boldsymbol{\theta}) \approx \mathcal{N}(\mathbf{f}|\hat{\mathbf{f}}, A^{-1})$$

where: $\hat{\mathbf{f}} = \text{argmax}_{\mathbf{f}}\, p(\mathbf{f}|\mathcal{D}, \boldsymbol{\theta}) = \text{argmax}_{\mathbf{f}}\;\; p(\mathbf{y}|\mathbf{f}, \boldsymbol{\theta}) p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta})$ and A is the Hessian of the negative log-posterior evaluated at $\hat{\mathbf{f}}$.

Hence we focus on the maximization of:

$$\psi(\mathbf{f}) = \log p(\mathbf{y}|\mathbf{f}) - \frac{1}{2}\mathbf{f}^T\mathbf{K}^{-1}\mathbf{f} - \frac{1}{2}\log|\mathbf{K}| - \frac{N}{2}\log 2\pi$$

Using Newton's method we obtain the following update:

$$\mathbf{f}^{\text{new}} = (\mathbf{W} + \mathbf{K}^{-1})^{-1}\left(\frac{\partial \log p(\mathbf{y}|\mathbf{f})}{\partial \mathbf{f}} + \mathbf{W}\mathbf{f}\right)$$

with $\mathbf{W}_{pq} = \dfrac{\partial^2 \log p(\mathbf{y}|\mathbf{f})}{\partial f_p \partial f_q}$.

The Laplace Approximation to the GP Binary Classifier

Gaussian approximation

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \boldsymbol{\theta}) \approx \mathcal{N}(\mathbf{f}|\hat{\mathbf{f}}, A^{-1})$$

where: $\hat{\mathbf{f}} = \operatorname{argmax}_{\mathbf{f}}\, p(\mathbf{f}|\mathcal{D}, \boldsymbol{\theta}) = \operatorname{argmax}_{\mathbf{f}}\;\; p(\mathbf{y}|\mathbf{f}, \boldsymbol{\theta}) p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta})$ and A is the Hessian of the negative log-posterior evaluated at $\hat{\mathbf{f}}$.

Hence we focus on the maximization of:

$$\psi(\mathbf{f}) = \log p(\mathbf{y}|\mathbf{f}) - \frac{1}{2}\mathbf{f}^T\mathbf{K}^{-1}\mathbf{f} - \frac{1}{2}\log|\mathbf{K}| - \frac{N}{2}\log 2\pi$$

Using Newton's method we obtain the following update:

$$\mathbf{f}^{\text{new}} = (\mathbf{W} + \mathbf{K}^{-1})^{-1}\left(\frac{\partial \log p(\mathbf{y}|\mathbf{f})}{\partial \mathbf{f}} + \mathbf{W}\mathbf{f}\right)$$

with $\mathbf{W}_{pq} = \dfrac{\partial^2 \log p(\mathbf{y}|\mathbf{f})}{\partial f_p \partial f_q}$.

Constraint on A? What does this imply?

The Lapace Approximation to GPC

Convergence and Uniqueness:

- Note that $\mathbf{W}$ is a diagonal matrix due to iid assumption
- for concave likelihood functions the un-normalized log posterior has a unique maximum

Once we have found the maximum posterior $\hat{\mathbf{f}}$ by using the above iteration we can show that:

$$p(\mathbf{f}|\mathcal{D}, \boldsymbol{\theta}) \approx \mathcal{N}(\mathbf{f}|\hat{\mathbf{f}}, (\mathbf{W} + \mathbf{K}^{-1})^{-1}).$$

When is this approximation a good/bad idea?

The Lapace Approximation to GPC

Convergence and Uniqueness:

- Note that $\mathbf{W}$ is a diagonal matrix due to iid assumption
- for concave likelihood functions the un-normalized log posterior has a unique maximum

Once we have found the maximum posterior $\hat{\mathbf{f}}$ by using the above iteration we can show that:

$$p(\mathbf{f}|\mathcal{D}, \boldsymbol{\theta}) \approx \mathcal{N}(\mathbf{f}|\hat{\mathbf{f}}, (\mathbf{W} + \mathbf{K}^{-1})^{-1}).$$

When is this approximation a good/bad idea?

- Better when N is large
- Only model aspects of distribution at a specific value
- Essentially uncontrolled (Hessian may be poor approximation to true shape of the posterior)

Posterior and Predictive Distributions

Recalling the posterior distribution:

$$p(f_* | \mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int p(f_* | \mathbf{X}, \mathbf{x}_*, \mathbf{f}) p(\mathbf{f} | \mathbf{X}, \mathbf{y}) d\mathbf{f}$$

Posterior and Predictive Distributions

Recalling the posterior distribution:

$$p(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{f}) p(\mathbf{f}|\mathbf{X}, \mathbf{y}) d\mathbf{f}$$

Hence, under Laplace approximation:

$$\mathbb{E}[f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*] = \mathbf{k}(\mathbf{x}_*)^T \mathbf{K}^{-1} \hat{\mathbf{f}}$$

$$\mathbb{V}[f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*] = \kappa(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T (\mathbf{K} + \mathbf{W}^{-1})^{-1} \mathbf{k}_*$$

- Note similarity with GP regression predictive distribution

Posterior and Predictive Distributions

Recalling the posterior distribution:

$$p(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{f}) p(\mathbf{f}|\mathbf{X}, \mathbf{y}) d\mathbf{f}$$

Hence, under Laplace approximation:

$$\mathbb{E}[f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*] = \mathbf{k}(\mathbf{x}_*)^T \mathbf{K}^{-1} \hat{\mathbf{f}}$$

$$\mathbb{V}[f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*] = \kappa(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T (\mathbf{K} + \mathbf{W}^{-1})^{-1} \mathbf{k}_*$$

- Note similarity with GP regression predictive distribution

For predictions we have two alternatives:

Average $\bar{\pi}_* = p(y_* = +1|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int \sigma(f_*) p(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) df_*$

MAP $\hat{\pi}_* = \sigma(\mathbb{E}[f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*])$

Posterior and Predictive Distributions

Recalling the posterior distribution:

$$p(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{f}) p(\mathbf{f}|\mathbf{X}, \mathbf{y}) d\mathbf{f}$$

Hence, under Laplace approximation:

$$\mathbb{E}[f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*] = \mathbf{k}(\mathbf{x}_*)^T \mathbf{K}^{-1} \hat{\mathbf{f}}$$
$$\mathbb{V}[f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*] = \kappa(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T (\mathbf{K} + \mathbf{W}^{-1})^{-1} \mathbf{k}_*$$

- Note similarity with GP regression predictive distribution

For predictions we have two alternatives:

Average $\bar{\pi}_* = p(y_* = +1|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int \sigma(f_*) p(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) df_*$

MAP $\hat{\pi}_* = \sigma(\mathbb{E}[f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*])$

- They provide the same prediction when concerned with most probable classification

Posterior and Predictive Distributions

Recalling the posterior distribution:

$$p(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{f}) p(\mathbf{f}|\mathbf{X}, \mathbf{y}) d\mathbf{f}$$

Hence, under Laplace approximation:

$$\mathbb{E}[f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*] = \mathbf{k}(\mathbf{x}_*)^T \mathbf{K}^{-1} \hat{\mathbf{f}}$$
$$\mathbb{V}[f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*] = \kappa(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T (\mathbf{K} + \mathbf{W}^{-1})^{-1} \mathbf{k}_*$$

- Note similarity with GP regression predictive distribution

For predictions we have two alternatives:

Average $\bar{\pi}_* = p(y_* = +1|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int \sigma(f_*) p(f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) df_*$

MAP $\hat{\pi}_* = \sigma(\mathbb{E}[f_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*])$

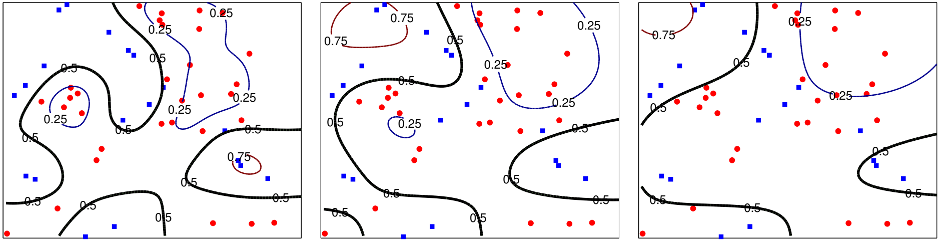
- They provide the same prediction when concerned with most probable classification
- Full distribution is required if we are concerned with confidence in the predictions (e.g. reject options)

Marginal Likelihood and hyper-parameter learning

We can also apply the Laplace approximation to the marginal likelihood:

$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) \approx -\frac{1}{2}\log|\mathbf{KW} + \mathbf{I}| - \frac{1}{2}\hat{\mathbf{f}}^T\mathbf{K}^{-1}\hat{\mathbf{f}} + \log p(\mathbf{y}|\hat{\mathbf{f}})$$

Predictive probability as a function of the length-scale $\ell = 0.1, 0.2, 0.3$:



Do we spend too much effort in modeling f?

GP Regression Computational Complexity

- Prediction: We need to compute the inverse of $\widetilde{\mathbf{K}} = \mathbf{K} + \sigma_n^2\mathbf{I}$, which scales $\mathcal{O}(N^3)$

GP Regression Computational Complexity

- Prediction: We need to compute the inverse of $\widetilde{\mathbf{K}} = \mathbf{K} + \sigma_n^2 \mathbf{I}$, which scales $\mathcal{O}(N^3)$
- In fact, we need to solve the linear system: $\widetilde{\mathbf{K}}\mathbf{b} = \mathbf{y}$

GP Regression Computational Complexity

- Prediction: We need to compute the inverse of $\widetilde{\mathbf{K}} = \mathbf{K} + \sigma_n^2\mathbf{I}$, which scales $\mathcal{O}(N^3)$
- In fact, we need to solve the linear system: $\widetilde{\mathbf{K}}\mathbf{b} = \mathbf{y}$
- Iterative solutions of Linear Systems (e.g. conjugate gradients):

GP Regression Computational Complexity

- Prediction: We need to compute the inverse of $\widetilde{\mathbf{K}} = \mathbf{K} + \sigma_n^2\mathbf{I}$, which scales $\mathcal{O}(N^3)$
- In fact, we need to solve the linear system: $\widetilde{\mathbf{K}}\mathbf{b} = \mathbf{y}$
- Iterative solutions of Linear Systems (e.g. conjugate gradients):
  - Exact when run for N iterations

GP Regression Computational Complexity

- Prediction: We need to compute the inverse of $\widetilde{\mathbf{K}} = \mathbf{K} + \sigma_n^2 \mathbf{I}$, which scales $\mathcal{O}(N^3)$
- In fact, we need to solve the linear system: $\widetilde{\mathbf{K}}\mathbf{b} = \mathbf{y}$
- Iterative solutions of Linear Systems (e.g. conjugate gradients):
  - Exact when run for N iterations
  - Approximate when run for $I < N$ iterations: $\mathcal{O}(I N^2)$

GP Regression Computational Complexity

- Prediction: We need to compute the inverse of $\widetilde{\mathbf{K}} = \mathbf{K} + \sigma_n^2\mathbf{I}$, which scales $\mathcal{O}(N^3)$
- In fact, we need to solve the linear system: $\widetilde{\mathbf{K}}\mathbf{b} = \mathbf{y}$
- Iterative solutions of Linear Systems (e.g. conjugate gradients):
  - Exact when run for N iterations
  - Approximate when run for $I < N$ iterations: $\mathcal{O}(IN^2)$
  - Not good enough

GP Regression Computational Complexity

- Prediction: We need to compute the inverse of $\widetilde{\mathbf{K}} = \mathbf{K} + \sigma_n^2 \mathbf{I}$, which scales $\mathcal{O}(N^3)$
- In fact, we need to solve the linear system: $\widetilde{\mathbf{K}}\mathbf{b} = \mathbf{y}$
- Iterative solutions of Linear Systems (e.g. conjugate gradients):
  - Exact when run for N iterations
  - Approximate when run for $I < N$ iterations: $\mathcal{O}(IN^2)$
  - Not good enough
- ML Approach:

GP Regression Computational Complexity

- Prediction: We need to compute the inverse of $\widetilde{\mathbf{K}} = \mathbf{K} + \sigma_n^2\mathbf{I}$, which scales $\mathcal{O}(N^3)$
- In fact, we need to solve the linear system: $\widetilde{\mathbf{K}}\mathbf{b} = \mathbf{y}$
- Iterative solutions of Linear Systems (e.g. conjugate gradients):
  - Exact when run for N iterations
  - Approximate when run for $I < N$ iterations: $\mathcal{O}(IN^2)$
  - Not good enough
- ML Approach:
  - Get a suitable decomposition of $\widetilde{\mathbf{K}}$ (e.g. using M inducing points)

GP Regression Computational Complexity

- Prediction: We need to compute the inverse of $\widetilde{\mathbf{K}} = \mathbf{K} + \sigma_n^2 \mathbf{I}$, which scales $\mathcal{O}(N^3)$
- In fact, we need to solve the linear system: $\widetilde{\mathbf{K}}\mathbf{b} = \mathbf{y}$
- Iterative solutions of Linear Systems (e.g. conjugate gradients):
  - Exact when run for N iterations
  - Approximate when run for $I < N$ iterations: $\mathcal{O}(I N^2)$
  - Not good enough
- ML Approach:
  - Get a suitable decomposition of $\widetilde{\mathbf{K}}$ (e.g. using M inducing points)
  - Apply matrix computational tricks (e.g. block inverses, Woodbury's formula)

GP Regression Computational Complexity

- Prediction: We need to compute the inverse of $\widetilde{\mathbf{K}} = \mathbf{K} + \sigma_n^2 \mathbf{I}$, which scales $\mathcal{O}(N^3)$
- In fact, we need to solve the linear system: $\widetilde{\mathbf{K}}\mathbf{b} = \mathbf{y}$
- Iterative solutions of Linear Systems (e.g. conjugate gradients):
  - Exact when run for N iterations
  - Approximate when run for $I < N$ iterations: $\mathcal{O}(IN^2)$
  - Not good enough
- ML Approach:
  - Get a suitable decomposition of $\widetilde{\mathbf{K}}$ (e.g. using M inducing points)
  - Apply matrix computational tricks (e.g. block inverses, Woodbury's formula)
  - Computations are usually $\mathcal{O}(M^2 N)$

GP Regression Computational Complexity

- Prediction: We need to compute the inverse of $\widetilde{\mathbf{K}} = \mathbf{K} + \sigma_n^2 \mathbf{I}$, which scales $\mathcal{O}(N^3)$
- In fact, we need to solve the linear system: $\widetilde{\mathbf{K}}\mathbf{b} = \mathbf{y}$
- Iterative solutions of Linear Systems (e.g. conjugate gradients):
  - Exact when run for N iterations
  - Approximate when run for $I < N$ iterations: $\mathcal{O}(IN^2)$
  - Not good enough
- ML Approach:
  - Get a suitable decomposition of $\widetilde{\mathbf{K}}$ (e.g. using M inducing points)
  - Apply matrix computational tricks (e.g. block inverses, Woodbury's formula)
  - Computations are usually $\mathcal{O}(M^2 N)$
  - Good enough?

Subset of Data-points (SD)

- Simplest approach: throw data away

Subset of Data-points (SD)

- Simplest approach: throw data away
- Keep the GP predictor on smaller set of size $M \rightarrow \mathcal{O}(M^3)$

Subset of Data-points (SD)

- Simplest approach: throw data away
- Keep the GP predictor on smaller set of size $M \to \mathcal{O}(M^3)$
- M data-points can be selected at random

Subset of Data-points (SD)

- Simplest approach: throw data away
- Keep the GP predictor on smaller set of size $M \to \mathcal{O}(M^3)$
- M data-points can be selected at random
- Alternatively, they can be selected in a greedy fashion in order to optimize an objective function.

Subset of Data-points (SD)

- Simplest approach: throw data away
- Keep the GP predictor on smaller set of size $M \to \mathcal{O}(M^3)$
- M data-points can be selected at random
- Alternatively, they can be selected in a greedy fashion in order to optimize an objective function.
- Lawrence et al (NIPS, 2003) propose the use of differential entropy:

$$\begin{aligned}\Delta_j &\overset{\text{def}}{=} H[p(f_j)] - H[p^{new}(f_j)] \\ &= \frac{1}{2}(1 + v_j/\sigma_n^2),\end{aligned}$$

where v_j is the posterior variance before the inclusion of the corresponding data-point.

Subset of Data-points (SD)

- Simplest approach: throw data away
- Keep the GP predictor on smaller set of size $M \rightarrow \mathcal{O}(M^3)$
- M data-points can be selected at random
- Alternatively, they can be selected in a greedy fashion in order to optimize an objective function.
- Lawrence et al (NIPS, 2003) propose the use of differential entropy:

$$\begin{aligned}\Delta_j &\overset{\text{def}}{=} H[p(f_j)] - H[p^{new}(f_j)] \\ &= \frac{1}{2}(1 + v_j/\sigma_n^2),\end{aligned}$$

  where v_j is the posterior variance before the inclusion of the corresponding data-point.
  - Simply choose the site with largest variance!

Subset of Data-points (SD)

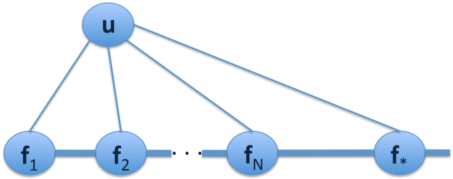
- Simplest approach: throw data away
- Keep the GP predictor on smaller set of size $M \rightarrow \mathcal{O}(M^3)$
- M data-points can be selected at random
- Alternatively, they can be selected in a greedy fashion in order to optimize an objective function.
- Lawrence et al (NIPS, 2003) propose the use of differential entropy:

$$\begin{aligned}\Delta_j &\overset{\text{def}}{=} H[p(f_j)] - H[p^{new}(f_j)] \\ &= \frac{1}{2}(1 + v_j/\sigma_n^2),\end{aligned}$$

  where v_j is the posterior variance before the inclusion of the corresponding data-point.
  - Simply choose the site with largest variance!
  - Overall complexity: $\mathcal{O}(M^2N)$

GP Approximations: A Unifying Framework (1)

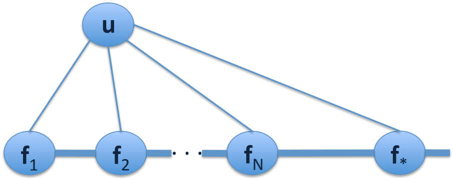
Quiñonero-Candela and Rasmussen (JMLR 2005)



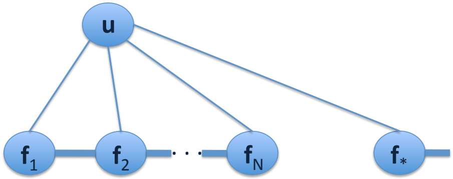
Full GP (no approximations). All latent functions are fully connected.

GP Approximations: A Unifying Framework (1)

Quiñonero-Candela and Rasmussen (JMLR 2005)



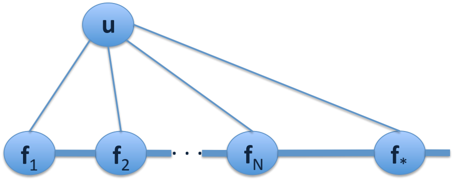
Full GP (no approximations). All latent functions are fully connected.



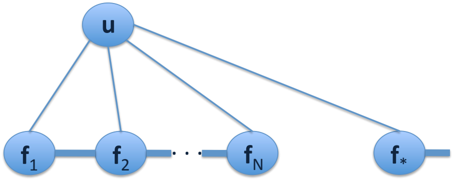
Traning and test values are conditionally independent given $\mathbf{u}$

GP Approximations: A Unifying Framework (1)

Quiñonero-Candela and Rasmussen (JMLR 2005)



Full GP (no approximations). All latent functions are fully connected.



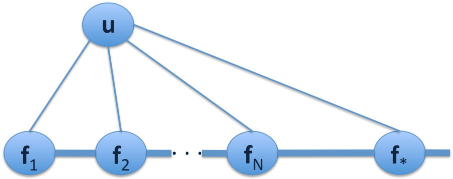
Traning and test values are conditionally independent given **u**

The joint prior is modified through the inducing variables $\mathbf{u} = \{u_1, \ldots, u_M\}$ which are indexed by the inducing inputs $\mathbf{Z} = \{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(M)}\}$:

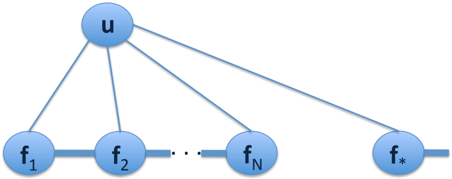
$$p(\mathbf{f}_*, \mathbf{f}) = \int p(\mathbf{f}_*, \mathbf{f}|\mathbf{u})p(\mathbf{u})d\mathbf{u} \quad \text{with } p(\mathbf{u}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{zz})$$

GP Approximations: A Unifying Framework (1)

Quiñonero-Candela and Rasmussen (JMLR 2005)



Full GP (no approximations). All latent functions are fully connected.



Traning and test values are conditionally independent given $\mathbf{u}$

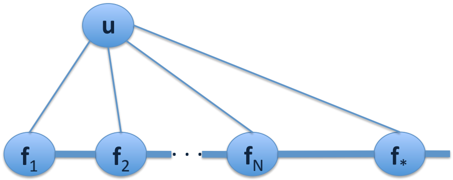
The joint prior is modified through the inducing variables $\mathbf{u} = \{u_1, \ldots, u_M\}$ which are indexed by the inducing inputs $\mathbf{Z} = \{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(M)}\}$:

$$p(\mathbf{f}_*, \mathbf{f}) = \int p(\mathbf{f}_*, \mathbf{f}|\mathbf{u}) p(\mathbf{u}) d\mathbf{u} \quad \text{with } p(\mathbf{u}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{zz})$$

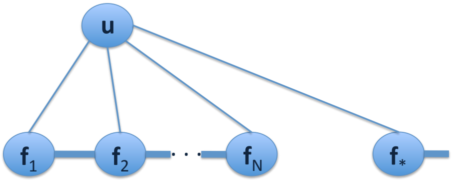
where $\mathbf{K}_{zz} = \kappa(\mathbf{Z}, \mathbf{Z}; \boldsymbol{\theta})$

GP Approximations: A Unifying Framework (2)

Quiñonero-Candela and Rasmussen (JMLR 2005)



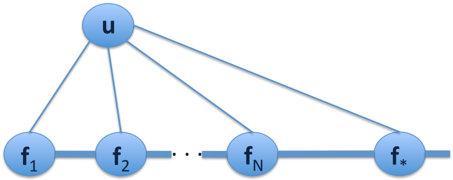
Full GP (no approximations). All latent functions are fully connected.



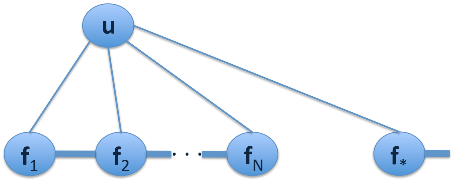
Traning and test values are conditionally independent given $\mathbf{u}$

GP Approximations: A Unifying Framework (2)

Quiñonero-Candela and Rasmussen (JMLR 2005)



Full GP (no approximations). All latent functions are fully connected.



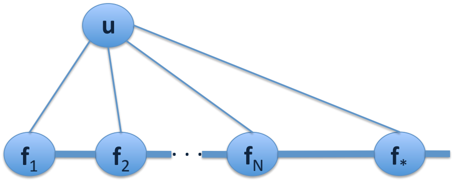
Traning and test values are conditionally independent given $\mathbf{u}$

The joint prior is modified through the inducing variables $u_1, \ldots, u_M$:

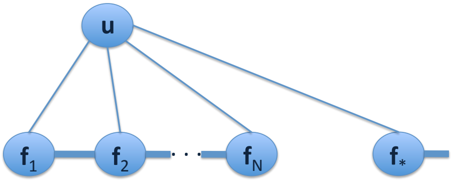
$$p(\mathbf{f}_*, \mathbf{f}) \approx q(\mathbf{f}_*, \mathbf{f}) \overset{\text{def}}{=} \int q(\mathbf{f}_*|\mathbf{u}) q(\mathbf{f}|\mathbf{u}) p(\mathbf{u}) d\mathbf{u}$$

GP Approximations: A Unifying Framework (2)

Quiñonero-Candela and Rasmussen (JMLR 2005)



Full GP (no approximations). All latent functions are fully connected.



Traning and test values are conditionally independent given $\mathbf{u}$

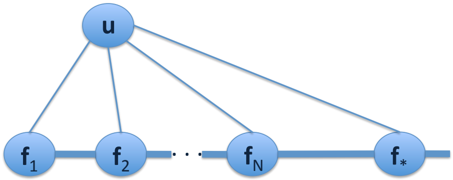
The joint prior is modified through the inducing variables $u_1, \ldots, u_M$:

$$p(\mathbf{f}_*, \mathbf{f}) \approx q(\mathbf{f}_*, \mathbf{f}) \overset{\text{def}}{=} \int q(\mathbf{f}_*|\mathbf{u}) q(\mathbf{f}|\mathbf{u}) p(\mathbf{u}) d\mathbf{u}$$

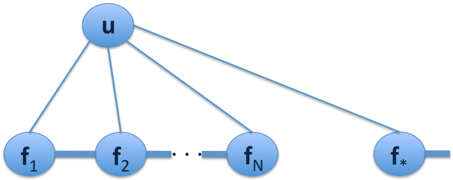
$q(\mathbf{f}|\mathbf{u})$ is the training conditional and $q(\mathbf{f}_*|\mathbf{u})$ is the test conditional.

GP Approximations: A Unifying Framework (2)

Quiñonero-Candela and Rasmussen (JMLR 2005)



Full GP (no approximations). All latent functions are fully connected.



Traning and test values are conditionally independent given $\mathbf{u}$

The joint prior is modified through the inducing variables $u_1, \ldots, u_M$:

$$p(\mathbf{f}_*, \mathbf{f}) \approx q(\mathbf{f}_*, \mathbf{f}) \overset{\text{def}}{=} \int q(\mathbf{f}_* | \mathbf{u}) q(\mathbf{f} | \mathbf{u}) p(\mathbf{u}) d\mathbf{u}$$

$q(\mathbf{f}|\mathbf{u})$ is the training conditional and $q(\mathbf{f}_*|\mathbf{u})$ is the test conditional.

Most approximation methods can be defined by:

- Different specifications of these conditionals.
- Different $\mathbf{Z}$: Subset of training/test points, new $\mathbf{x}$ points

Subset of Regressors (SR)

It can be shown that the mean GP predictor can be obtained by assuming:

$$\begin{aligned} \text{Prior} &: \boldsymbol{\alpha} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}^{-1}) \\ \text{Model} &: f(\mathbf{x}_*) = \textstyle\sum_{i=1}^{N} \alpha_i k(\mathbf{x}_*, \mathbf{x}^{(i)}) \end{aligned}$$

Subset of Regressors (SR)

It can be shown that the mean GP predictor can be obtained by assuming:

Prior : $\boldsymbol{\alpha} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}^{-1})$

Model : $f(\mathbf{x}_*) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_*, \mathbf{x}^{(i)})$

We can truncate the number of regressors needed:

$$f_{\mathrm{SR}}(\mathbf{x}_*) = \mathbf{k}_*^T \boldsymbol{\alpha}_{\mathrm{z}} \text{ with } \boldsymbol{\alpha}_{\mathrm{z}} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathrm{zz}}^{-1})$$

Subset of Regressors (SR)

It can be shown that the mean GP predictor can be obtained by assuming:

Prior : $\boldsymbol{\alpha} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}^{-1})$

Model : $f(\mathbf{x}_*) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_*, \mathbf{x}^{(i)})$

We can truncate the number of regressors needed:

$$f_{\mathrm{SR}}(\mathbf{x}_*) = \mathbf{k}_*^T \boldsymbol{\alpha}_{\mathrm{z}} \text{ with } \boldsymbol{\alpha}_{\mathrm{z}} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathrm{zz}}^{-1})$$

This implies that there is a deterministic relation between $\mathbf{f}_*$ and $\mathbf{u}$:

$$q_{\mathrm{SR}}(\mathbf{f}|\mathbf{u}) = \mathcal{N}(\mathbf{K}_{\mathrm{xz}}\mathbf{K}_{\mathrm{zz}}^{-1}\mathbf{u}, \mathbf{0}) \quad q_{\mathrm{SR}}(\mathbf{f}_*|\mathbf{u}) = \mathcal{N}(\mathbf{K}_{*\mathrm{x}}\mathbf{K}_{\mathrm{zz}}^{-1}\mathbf{u}, \mathbf{0})$$

Subset of Regressors (SR)

It can be shown that the mean GP predictor can be obtained by assuming:

Prior : $\boldsymbol{\alpha} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}^{-1})$

Model : $f(\mathbf{x}_*) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_*, \mathbf{x}^{(i)})$

We can truncate the number of regressors needed:

$$f_{\mathrm{SR}}(\mathbf{x}_*) = \mathbf{k}_*^T \boldsymbol{\alpha}_{\mathrm{z}} \text{ with } \boldsymbol{\alpha}_{\mathrm{z}} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathrm{zz}}^{-1})$$

This implies that there is a deterministic relation between $\mathbf{f}_*$ and $\mathbf{u}$:

$$q_{\mathrm{SR}}(\mathbf{f}|\mathbf{u}) = \mathcal{N}(\mathbf{K}_{\mathrm{xz}}\mathbf{K}_{\mathrm{zz}}^{-1}\mathbf{u}, \mathbf{0}) \quad q_{\mathrm{SR}}(\mathbf{f}_*|\mathbf{u}) = \mathcal{N}(\mathbf{K}_{*\mathrm{x}}\mathbf{K}_{\mathrm{zz}}^{-1}\mathbf{u}, \mathbf{0})$$

Hence the predictive distribution is given by:

$$q_{\mathrm{SR}}(\mathbf{f}_*|\mathbf{y}) = \mathcal{N}(\mathbf{K}_{*\mathrm{x}}\boldsymbol{\Sigma}^{-1}\mathbf{K}_{\mathrm{zx}}\mathbf{y}, \mathbf{K}_{*\mathrm{x}}\boldsymbol{\Sigma}^{-1}\mathbf{K}_{\mathrm{x}*})$$

where $\boldsymbol{\Sigma} = \mathbf{K}_{\mathrm{zx}}\mathbf{K}_{\mathrm{xz}} + \sigma_n^2\mathbf{K}_{\mathrm{zz}}$.

Subset of Regressors (SR)

It can be shown that the mean GP predictor can be obtained by assuming:

$$\begin{aligned} \text{Prior} &: \boldsymbol{\alpha} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}^{-1}) \\ \text{Model} &: f(\mathbf{x}_*) = \textstyle\sum_{i=1}^{N} \alpha_i k(\mathbf{x}_*, \mathbf{x}^{(i)}) \end{aligned}$$

We can truncate the number of regressors needed:

$$f_{\text{SR}}(\mathbf{x}_*) = \mathbf{k}_*^T \boldsymbol{\alpha}_{\text{z}} \text{ with } \boldsymbol{\alpha}_{\text{z}} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{\text{zz}}^{-1})$$

This implies that there is a deterministic relation between $\mathbf{f}_*$ and $\mathbf{u}$:

$$q_{\text{SR}}(\mathbf{f}|\mathbf{u}) = \mathcal{N}(\mathbf{K}_{\text{xz}}\mathbf{K}_{\text{zz}}^{-1}\mathbf{u}, \mathbf{0}) \quad q_{\text{SR}}(\mathbf{f}_*|\mathbf{u}) = \mathcal{N}(\mathbf{K}_{*\text{x}}\mathbf{K}_{\text{zz}}^{-1}\mathbf{u}, \mathbf{0})$$

Hence the predictive distribution is given by:

$$q_{\text{SR}}(\mathbf{f}_*|\mathbf{y}) = \mathcal{N}(\mathbf{K}_{*\text{x}}\boldsymbol{\Sigma}^{-1}\mathbf{K}_{\text{zx}}\mathbf{y}, \mathbf{K}_{*\text{x}}\boldsymbol{\Sigma}^{-1}\mathbf{K}_{\text{x}*})$$

where $\boldsymbol{\Sigma} = \mathbf{K}_{\text{zx}}\mathbf{K}_{\text{xz}} + \sigma_n^2\mathbf{K}_{\text{zz}}$.

- This method corresponds to a degenerate GP prior

Subset of Regressors (SR)

It can be shown that the mean GP predictor can be obtained by assuming:

Prior : $\boldsymbol{\alpha} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}^{-1})$

Model : $f(\mathbf{x}_*) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_*, \mathbf{x}^{(i)})$

We can truncate the number of regressors needed:

$$f_{\mathrm{SR}}(\mathbf{x}_*) = \mathbf{k}_*^T \boldsymbol{\alpha}_{\mathrm{z}} \text{ with } \boldsymbol{\alpha}_{\mathrm{z}} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathrm{zz}}^{-1})$$

This implies that there is a deterministic relation between $\mathbf{f}_*$ and $\mathbf{u}$:

$$q_{\mathrm{SR}}(\mathbf{f}|\mathbf{u}) = \mathcal{N}(\mathbf{K}_{\mathrm{xz}}\mathbf{K}_{\mathrm{zz}}^{-1}\mathbf{u}, \mathbf{0}) \quad q_{\mathrm{SR}}(\mathbf{f}_*|\mathbf{u}) = \mathcal{N}(\mathbf{K}_{*\mathrm{x}}\mathbf{K}_{\mathrm{zz}}^{-1}\mathbf{u}, \mathbf{0})$$

Hence the predictive distribution is given by:

$$q_{\mathrm{SR}}(\mathbf{f}_*|\mathbf{y}) = \mathcal{N}(\mathbf{K}_{*\mathrm{x}}\boldsymbol{\Sigma}^{-1}\mathbf{K}_{\mathrm{zx}}\mathbf{y}, \mathbf{K}_{*\mathrm{x}}\boldsymbol{\Sigma}^{-1}\mathbf{K}_{\mathrm{x}*})$$

where $\boldsymbol{\Sigma} = \mathbf{K}_{\mathrm{zx}}\mathbf{K}_{\mathrm{xz}} + \sigma_n^2\mathbf{K}_{\mathrm{zz}}$.

- This method corresponds to a degenerate GP prior
- Complexity: $\mathcal{O}(M^2N)$ initially and $\mathcal{O}(M)$ and $\mathcal{O}(M^2)$ per test predictive mean and variance.

Projected Processes (PP)

$$q_{\mathrm{PP}}(\mathbf{f}|\mathbf{u}) = \mathcal{N}(\mathbf{K}_{\mathrm{xz}}\mathbf{K}_{\mathrm{zz}}^{-1}\mathbf{u}, \mathbf{0}) \quad q_{\mathrm{PP}}(\mathbf{f}_*|\mathbf{u}) = p(\mathbf{f}_*|\mathbf{u})$$

Projected Processes (PP)

$$q_{\mathsf{PP}}(\mathbf{f}|\mathbf{u}) = \mathcal{N}(\mathbf{K}_{\mathsf{xz}}\mathbf{K}_{\mathsf{zz}}^{-1}\mathbf{u}, \mathbf{0}) \quad q_{\mathsf{PP}}(\mathbf{f}_*|\mathbf{u}) = p(\mathbf{f}_*|\mathbf{u})$$

- Inducing variables are a subset of training points

Projected Processes (PP)

$$q_{\text{PP}}(\mathbf{f}|\mathbf{u}) = \mathcal{N}(\mathbf{K}_{\text{xz}}\mathbf{K}_{\text{zz}}^{-1}\mathbf{u}, \mathbf{0}) \quad q_{\text{PP}}(\mathbf{f}_*|\mathbf{u}) = p(\mathbf{f}_*|\mathbf{u})$$

- Inducing variables are a subset of training points
- As in SR, it imposes a deterministic training conditional but (unlike SR) it uses the exact test conditional.

Projected Processes (PP)

$$q_{\mathrm{PP}}(\mathbf{f}|\mathbf{u}) = \mathcal{N}(\mathbf{K}_{\mathrm{xz}}\mathbf{K}_{\mathrm{zz}}^{-1}\mathbf{u}, \mathbf{0}) \quad q_{\mathrm{PP}}(\mathbf{f}_*|\mathbf{u}) = p(\mathbf{f}_*|\mathbf{u})$$

- Inducing variables are a subset of training points
- As in SR, it imposes a deterministic training conditional but (unlike SR) it uses the exact test conditional.
- Same predictive mean as SR but variances are never smaller

Projected Processes (PP)

$$q_{\text{PP}}(\mathbf{f}|\mathbf{u}) = \mathcal{N}(\mathbf{K}_{\text{xz}}\mathbf{K}_{\text{zz}}^{-1}\mathbf{u}, \mathbf{0}) \quad q_{\text{PP}}(\mathbf{f}_*|\mathbf{u}) = p(\mathbf{f}_*|\mathbf{u})$$

- Inducing variables are a subset of training points
- As in SR, it imposes a deterministic training conditional but (unlike SR) it uses the exact test conditional.
- Same predictive mean as SR but variances are never smaller
- However, this definition implies that the covariances for training cases and test cases are computed differently and therefore this method does not correspond to a (consistent) GP.

FITC, PITC and BCM

FITC	: Fully independent training conditionals
PITC	: Partially independent training conditionals
BCM	: Bayesian Committee Machine

FITC, PITC and BCM

FITC	: Fully independent training conditionals
PITC	: Partially independent training conditionals
BCM	: Bayesian Committee Machine

- PP can make poor predictions in low noise

FITC, PITC and BCM

FITC	: Fully independent training conditionals
PITC	: Partially independent training conditionals
BCM	: Bayesian Committee Machine

- PP can make poor predictions in low noise
- FITC does not impose a deterministic relation between $\mathbf{f}$ and $\mathbf{u}$. It uses a a diagonal covariance whose entries correspond to the diagonal of the true training conditionals.

FITC, PITC and BCM

FITC	:	Fully independent training conditionals
PITC	:	Partially independent training conditionals
BCM	:	Bayesian Committee Machine

- PP can make poor predictions in low noise
- FITC does not impose a deterministic relation between $\mathbf{f}$ and $\mathbf{u}$. It uses a a diagonal covariance whose entries correspond to the diagonal of the true training conditionals.
- PITC uses block diagonal covariance to improve the approximation

FITC, PITC and BCM

FITC	:	Fully independent training conditionals
PITC	:	Partially independent training conditionals
BCM	:	Bayesian Committee Machine

- PP can make poor predictions in low noise
- FITC does not impose a deterministic relation between $\mathbf{f}$ and $\mathbf{u}$. It uses a a diagonal covariance whose entries correspond to the diagonal of the true training conditionals.
- PITC uses block diagonal covariance to improve the approximation
- BCM is the same as PITC where the choice of inducing variables depend on the test points, i.e. transductive setting
  - However, note that transduction cannot occur in exact GPs
  - Drawback regarding complexity of transductive models?
  - The choice of $\mathbf{u}$ should not be dictated only by the test points

Sparse GPs (Snelson and Ghahramani, 2006)

- Same as FITC but the inducing inputs do not belong to the training or test sets
- Both the locations of the input points and the values of the hyper-parameters are “learned” by optimization of the approximate marginal likelihood.

GP Approximations: Final Remarks

- The order of computational complexity is identical for all methods (except SD)
- Hence, there is no "excuse for gross approximations"
- Inconclusive experiments on real datasets (See e.g. Rassmussen and Williams, 2006)
- Similar methods for GP classification but we also need to deal with non-Gaussian likelihoods (e.g. using Laplace)
  - Derivatives of the marginal likelihood can get complicated

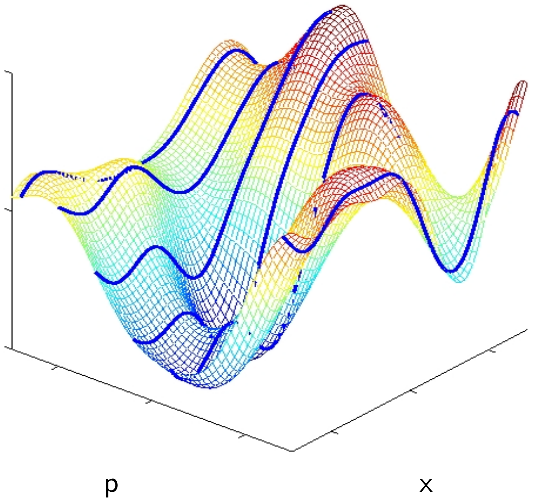
Multi-task Learning (MTL)

- General idea:
  - Sharing information across tasks (Caruana, 1997)
  - Very little data on test task
  - Exam score prediction, compiler performance prediction, robot inverse dynamics, multi-topic text categorisation, collaborative filtering, multi-level modelling

Multi-task Learning (MTL)

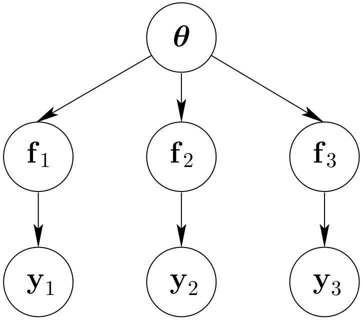
- General idea:
  - Sharing information across tasks (Caruana, 1997)
  - Very little data on test task
  - Exam score prediction, compiler performance prediction, robot inverse dynamics, multi-topic text categorisation, collaborative filtering, multi-level modelling
- Assuming task relatedness can be detrimental (Caruana, 1997; Baxter, 2000)
- Task descriptors may be available (Bonilla et al, AISTATS 2007)
- Tasks descriptors unavailable or difficult to define correctly (Bonilla et al, NIPS 2008)
  - e.g. Compiler performance prediction: code features, responses

Multi-task GP: Illustration



Sample functions for different values of tasks (on *p* axis) are correlated (cf **independent** draws over sample functions)

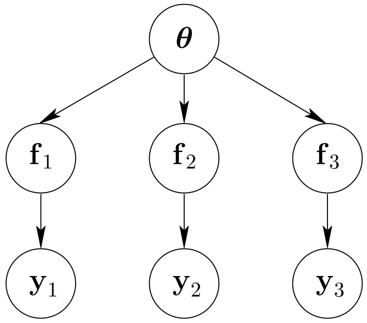
Inter-task Tying by Hyper-parameter Sharing


$$\begin{tikzcd} & \boldsymbol{\theta} \arrow[dl] \arrow[d] \arrow[dr] & \\ \mathbf{f}_1 \arrow[d] & \mathbf{f}_2 \arrow[d] & \mathbf{f}_3 \arrow[d] \\ \mathbf{y}_1 & \mathbf{y}_2 & \mathbf{y}_3 \end{tikzcd}$$

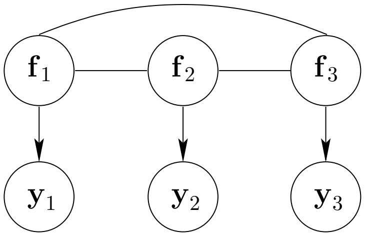
Other approaches

- Block diagonal covariance matrix, and each of the P blocks is induced from the same kernel function (Minka and Picard, 1999; Lawrence and Platt, 2004; Yu et al, 2005; Schwaighofer et al, 2005)

Inter-task Tying by Hyper-parameter Sharing



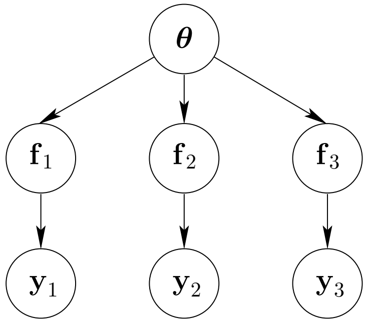
Other approaches



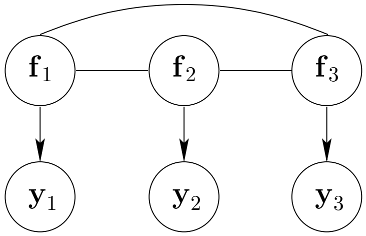
Multi-task GP

- Block diagonal covariance matrix, and each of the P blocks is induced from the same kernel function (Minka and Picard, 1999; Lawrence and Platt, 2004; Yu et al, 2005; Schwaighofer et al, 2005)

Inter-task Tying by Hyper-parameter Sharing



Other approaches



Multi-task GP

- Block diagonal covariance matrix, and each of the P blocks is induced from the same kernel function (Minka and Picard, 1999; Lawrence and Platt, 2004; Yu et al, 2005; Schwaighofer et al, 2005)
- Multi-task GP: Observations on one task affect predictions on the others

Multi-task GP (MTGP)

We place a (zero mean) GP prior over the latent functions $\{f_\ell\}$:

The Model

$$\langle f_\ell(\mathbf{x}) f_m(\mathbf{x}') \rangle = K^f_{\ell m} k^x(\mathbf{x}, \mathbf{x}') \qquad y_{i\ell} \sim \mathcal{N}(f_\ell(\mathbf{x}^{(i)}), \sigma^2_\ell),$$

K^f: PSD matrix that specifies the inter-task similarities

k^x: Covariance function over inputs

σ^2_ℓ: Noise variance for the ℓ^{th} task.

Multi-task GP (MTGP)

We place a (zero mean) GP prior over the latent functions $\{f_\ell\}$:

The Model

$$\langle f_\ell(\mathbf{x}) f_m(\mathbf{x}') \rangle = K^f_{\ell m} k^x(\mathbf{x}, \mathbf{x}') \qquad y_{i\ell} \sim \mathcal{N}(f_\ell(\mathbf{x}^{(i)}), \sigma^2_\ell),$$

K^f: PSD matrix that specifies the inter-task similarities

k^x: Covariance function over inputs

σ^2_ℓ: Noise variance for the ℓ^{th} task.

Additionally, k^x:

- stationary, **correlation** function
- e.g. squared exponential

Multi-task GP (MTGP)

We place a (zero mean) GP prior over the latent functions $\{f_\ell\}$:

The Model

$$\langle f_\ell(\mathbf{x}) f_m(\mathbf{x}') \rangle = K^f_{\ell m} k^x(\mathbf{x}, \mathbf{x}') \qquad y_{i\ell} \sim \mathcal{N}(f_\ell(\mathbf{x}^{(i)}), \sigma^2_\ell),$$

K^f: PSD matrix that specifies the inter-task similarities

k^x: Covariance function over inputs

σ^2_ℓ: Noise variance for the ℓ^{th} task.

Additionally, k^x:

- stationary, **correlation** function
- e.g. squared exponential

Correlations between tasks modelled directly via K^f

Multi-task GP Models

K^f can be:

- Full non-parametric: General PSD matrix, e.g. $K^f = (L^f)(L^f)^T$

Multi-task GP Models

K^f can be:

- Full non-parametric: General PSD matrix, e.g. $K^f = (L^f)(L^f)^T$
- Rank Constrained: e.g. $K^f = (\tilde{L}^f)(\tilde{L}^f)^T$

Multi-task GP Models

K^f can be:

- Full non-parametric: General PSD matrix, e.g. $K^f = (L^f)(L^f)^T$
- Rank Constrained: e.g. $K^f = (\tilde{L}^f)(\tilde{L}^f)^T$
- Parametric: K^f induced via a covariance function on task descriptors $k^f(\mathbf{t}, \mathbf{t}')$

Multi-task GP Models

K^f can be:

- Full non-parametric: General PSD matrix, e.g. $K^f = (L^f)(L^f)^T$
- Rank Constrained: e.g. $K^f = (\tilde{L}^f)(\tilde{L}^f)^T$
- Parametric: K^f induced via a covariance function on task descriptors $k^f(\mathbf{t}, \mathbf{t}')$
- Block diagonal: Implements task clustering. Cluster structure can be specified a priori. e.g. K^f is diagonal (all tasks are independent)

Multi-task GP Models

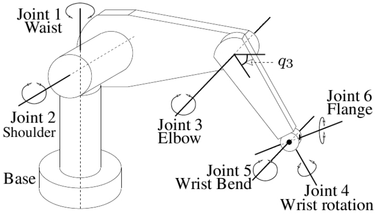
K^f can be:

- Full non-parametric: General PSD matrix, e.g. $K^f = (L^f)(L^f)^T$
- Rank Constrained: e.g. $K^f = (\tilde{L}^f)(\tilde{L}^f)^T$
- Parametric: K^f induced via a covariance function on task descriptors $k^f(\mathbf{t}, \mathbf{t}')$
- Block diagonal: Implements task clustering. Cluster structure can be specified a priori. e.g. K^f is diagonal (all tasks are independent)
- Mixture: All functions are independent except for one, which is a mixed version of the others. Effective for transferring to a new task:

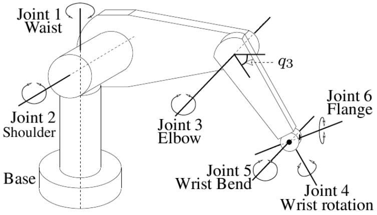
$$K^f = \begin{pmatrix} I & \boldsymbol{\pi} \\ \boldsymbol{\pi}^{\mathsf{T}} & \boldsymbol{\pi}^{\mathsf{T}}\boldsymbol{\pi} \end{pmatrix},$$

where $\boldsymbol{\pi}$ are mixing proportions, and may depend on task descriptors.

Learning Robot Inverse Dynamics (Chai et al, NIPS 2009)

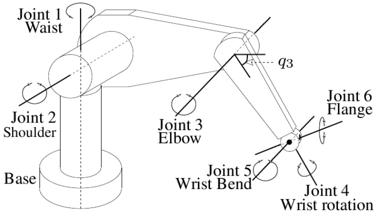


Learning Robot Inverse Dynamics (Chai et al, NIPS 2009)



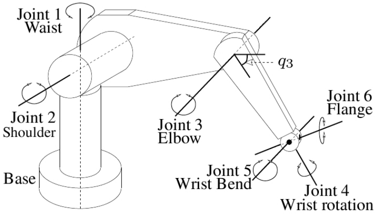
- $\boldsymbol{\tau}$: Torques needed at joints to drive a trajectory $\mathbf{x} = (\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$
- Unfeasible analytical model, e.g. friction, uncertainty in physical parameters
- Need to be controlled while having different loads (tasks)

Learning Robot Inverse Dynamics (Chai et al, NIPS 2009)



- $\boldsymbol{\tau}$: Torques needed at joints to drive a trajectory $\mathbf{x} = (\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$
- Unfeasible analytical model, e.g. friction, uncertainty in physical parameters
- Need to be controlled while having different loads (tasks)
- torque function changes as a function of the load on end effector

Learning Robot Inverse Dynamics (Chai et al, NIPS 2009)

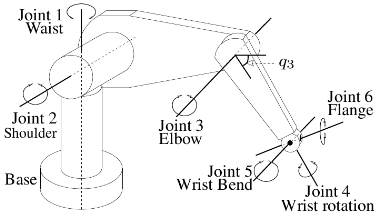


- $\boldsymbol{\tau}$: Torques needed at joints to drive a trajectory $\mathbf{x} = (\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$
- Unfeasible analytical model, e.g. friction, uncertainty in physical parameters
- Need to be controlled while having different loads (tasks)

- torque function changes as a function of the load on end effector

$$\tau_j^m(\mathbf{x}) = \mathbf{z}_j(\mathbf{x})^T \boldsymbol{\rho}_j^m$$

Learning Robot Inverse Dynamics (Chai et al, NIPS 2009)



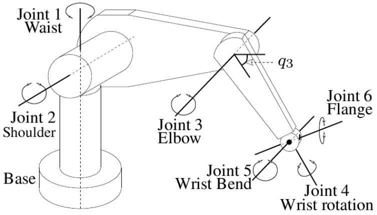
- $\boldsymbol{\tau}$: Torques needed at joints to drive a trajectory $\mathbf{x} = (\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$
- Unfeasible analytical model, e.g. friction, uncertainty in physical parameters
- Need to be controlled while having different loads (tasks)

- torque function changes as a function of the load on end effector

$$\tau_j^m(\mathbf{x}) = \mathbf{z}_j(\mathbf{x})^T \boldsymbol{\rho}_j^m$$

Indep. GP prior $\langle z_{j\alpha}(\mathbf{x}) z_{j'\alpha'}(\mathbf{x}') \rangle = \delta_{jj'} \delta_{\alpha\alpha'} k_j^x(\mathbf{x}, \mathbf{x}')$

Learning Robot Inverse Dynamics (Chai et al, NIPS 2009)



- $\boldsymbol{\tau}$: Torques needed at joints to drive a trajectory $\mathbf{x} = (\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$
- Unfeasible analytical model, e.g. friction, uncertainty in physical parameters
- Need to be controlled while having different loads (tasks)

- torque function changes as a function of the load on end effector

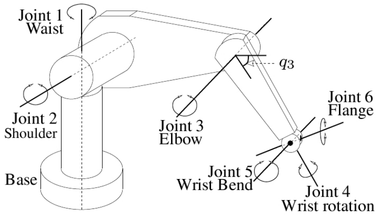
$$\tau_j^m(\mathbf{x}) = \mathbf{z}_j(\mathbf{x})^T \boldsymbol{\rho}_j^m$$

Indep. GP prior $\langle z_{j\alpha}(\mathbf{x}) z_{j'\alpha'}(\mathbf{x}') \rangle = \delta_{jj'} \delta_{\alpha\alpha'} k_j^x(\mathbf{x}, \mathbf{x}')$

$$\Downarrow$$

MTGP prior $\langle \tau_j^m(\mathbf{x}) \tau_j^{m'}(\mathbf{x}') \rangle = (K_j^\rho)_{mm'} k_j^x(\mathbf{x}, \mathbf{x}')$

Learning Robot Inverse Dynamics (Chai et al, NIPS 2009)



- $\boldsymbol{\tau}$: Torques needed at joints to drive a trajectory $\mathbf{x} = (\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$
- Unfeasible analytical model, e.g. friction, uncertainty in physical parameters
- Need to be controlled while having different loads (tasks)

- torque function changes as a function of the load on end effector

$$\tau_j^m(\mathbf{x}) = \mathbf{z}_j(\mathbf{x})^T \boldsymbol{\rho}_j^m$$

Indep. GP prior $\quad \langle z_{j\alpha}(\mathbf{x}) z_{j'\alpha'}(\mathbf{x}') \rangle = \delta_{jj'} \delta_{\alpha\alpha'} k_j^x(\mathbf{x}, \mathbf{x}')$

$$\Downarrow$$

MTGP prior $\quad \langle \tau_j^m(\mathbf{x}) \tau_j^{m'}(\mathbf{x}') \rangle = (K_j^\rho)_{mm'} k_j^x(\mathbf{x}, \mathbf{x}')$

The MTGP model matches the correlations between torque functions

Other Non-Gaussian Likelihood Models

- We have encountered this in GP classification

Other Non-Gaussian Likelihood Models

- We have encountered this in GP classification
- Ordinal regression: Chu and Ghahramani, JMLR 2005

Other Non-Gaussian Likelihood Models

- We have encountered this in GP classification
- Ordinal regression: Chu and Ghahramani, JMLR 2005
- Preference Learning: Chu and Ghahramani, ICML 2005

Other Non-Gaussian Likelihood Models

- We have encountered this in GP classification
- Ordinal regression: Chu and Ghahramani, JMLR 2005
- Preference Learning: Chu and Ghahramani, ICML 2005
- Preference Elicitation (PE): Bonilla et al, NIPS 2010
  - Make optimal recommendations to users by actively querying their preferences.
  - Bayesian decision-theoretic PE approach
  - Correlated GP prior over user's latent utility functions
  - Reduce elicitation burden by leveraging information from previous users

Other Non-Gaussian Likelihood Models

- We have encountered this in GP classification
- Ordinal regression: Chu and Ghahramani, JMLR 2005
- Preference Learning: Chu and Ghahramani, ICML 2005
- Preference Elicitation (PE): Bonilla et al, NIPS 2010
  - Make optimal recommendations to users by actively querying their preferences.
  - Bayesian decision-theoretic PE approach
  - Correlated GP prior over user's latent utility functions
  - Reduce elicitation burden by leveraging information from previous users
- Log Cox Gaussian Process (LGCP)
  - Modelling count data

Latent Variable Models

The Gaussian Process Latent Variable Model (GPLVM; Lawrence, NIPS 2004) is a probabilistic model for non-linear dimensionality reduction.

Latent Variable Models

The Gaussian Process Latent Variable Model (GPLVM; Lawrence, NIPS 2004) is a probabilistic model for non-linear dimensionality reduction.

- Main idea: Some high-dimensional data can be embedded into a low-dimensional non-linear manifold.

Latent Variable Models

The Gaussian Process Latent Variable Model (GPLVM; Lawrence, NIPS 2004) is a probabilistic model for non-linear dimensionality reduction.

- Main idea: Some high-dimensional data can be embedded into a low-dimensional non-linear manifold.
- model each dimension of $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$ with a corresponding latent point $\mathbf{z}_i$ through a non-linear mapping.

Latent Variable Models

The Gaussian Process Latent Variable Model (GPLVM; Lawrence, NIPS 2004) is a probabilistic model for non-linear dimensionality reduction.

- Main idea: Some high-dimensional data can be embedded into a low-dimensional non-linear manifold.
- model each dimension of $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$ with a corresponding latent point $\mathbf{z}_i$ through a non-linear mapping.
- Use an independent GP for this mapping

Latent Variable Models

The Gaussian Process Latent Variable Model (GPLVM; Lawrence, NIPS 2004) is a probabilistic model for non-linear dimensionality reduction.

- Main idea: Some high-dimensional data can be embedded into a low-dimensional non-linear manifold.
- model each dimension of $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$ with a corresponding latent point $\mathbf{z}_i$ through a non-linear mapping.
- Use an independent GP for this mapping
- Likelihood maximization in order to find the latent projection $\mathbf{z}^{(i)}$

Latent Variable Models

The Gaussian Process Latent Variable Model (GPLVM; Lawrence, NIPS 2004) is a probabilistic model for non-linear dimensionality reduction.

- Main idea: Some high-dimensional data can be embedded into a low-dimensional non-linear manifold.
- model each dimension of $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$ with a corresponding latent point $\mathbf{z}_i$ through a non-linear mapping.
- Use an independent GP for this mapping
- Likelihood maximization in order to find the latent projection $\mathbf{z}^{(i)}$
- GP models for pose estimation: `http://grail.cs.washington.edu/projects/styleik`

Modeling of Human Poses with GPLVM

Grochow et al, SIGGRAPH 2004

- Style-Based Inverse Kinematics: Given a set of constraints, produce the most likely pose.

Modeling of Human Poses with GPLVM

Grochow et al, SIGGRAPH 2004

- Style-Based Inverse Kinematics: Given a set of constraints, produce the most likely pose.
- Feature vectors are derived from pose information (e.g from mo-cap data).
  - joint angles, vertical orientation, velocity and accelerations.

Modeling of Human Poses with GPLVM

Grochow et al, SIGGRAPH 2004

- Style-Based Inverse Kinematics: Given a set of constraints, produce the most likely pose.
- Feature vectors are derived from pose information (e.g from mo-cap data).
    - joint angles, vertical orientation, velocity and accelerations.
- The problem is inherently underdetermined but some poses are more likely than others.

Modeling of Human Poses with GPLVM

Grochow et al, SIGGRAPH 2004

- Style-Based Inverse Kinematics: Given a set of constraints, produce the most likely pose.
- Feature vectors are derived from pose information (e.g from mo-cap data).
  - joint angles, vertical orientation, velocity and accelerations.
- The problem is inherently underdetermined but some poses are more likely than others.
- Low dimensional representations are learned from previous poses using GPLVM

Modeling of Human Poses with GPLVM

Grochow et al, SIGGRAPH 2004

- Style-Based Inverse Kinematics: Given a set of constraints, produce the most likely pose.
- Feature vectors are derived from pose information (e.g from mo-cap data).
  - joint angles, vertical orientation, velocity and accelerations.
- The problem is inherently underdetermined but some poses are more likely than others.
- Low dimensional representations are learned from previous poses using GPLVM
- GPLVM predictive distribution is used in objective function to find new poses given the constraints.

Pose Estimation Movies

From `http://grail.cs.washington.edu/projects/styleik`

Style Pitch

Style Track

Pose Track

Image Pose Basketball

Image Pose Baseball

Interpolation

Conclusions

- GPs as flexible non-parameteric Bayesian technique for regression, classification and other machine learning problems.
- The covariance function is a crucial component in GPs.
- Analytic solutions for standard regression setting and approximate inference for classification.
- Computational issues dealt with through the idea of inducing variables and a single unifying probabilistic framework.
- Dealing with non-standard settings, e.g. preference learning and multi-task learning.
- Reading:
  - Rasmussen & Williams (GPML, 2006): Ch. 3 (except sec 3.5, 3.6), Ch. 8
  - [Strongly recommended] Quiñonero-Candela and Rasmussen (A Unifying View of Sparse Approximate Gaussian Process Regression, JMLR, 2005)