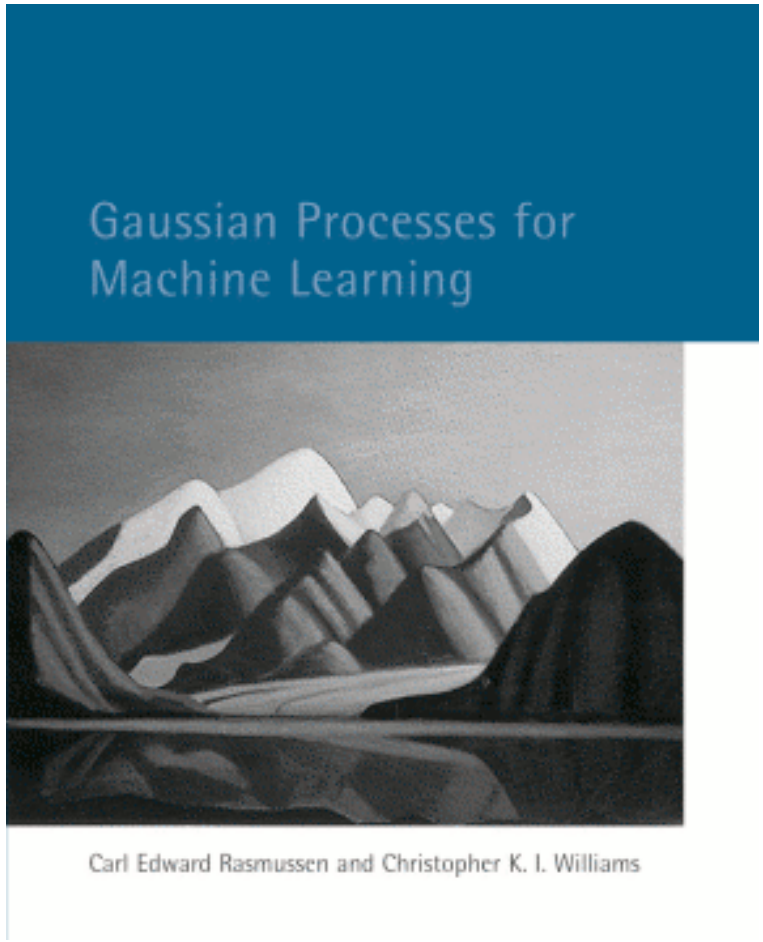
Gaussian Processes

Edwin V. Bonilla

Machine Learning Summer School

October 1st, 2010

The Book

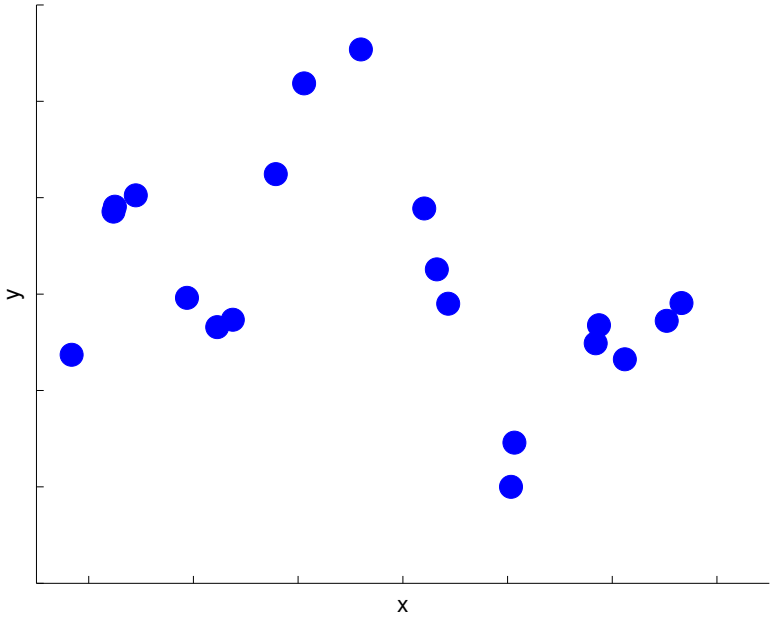


Carl Edward Rasmussen and Christopher K. I. Williams

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

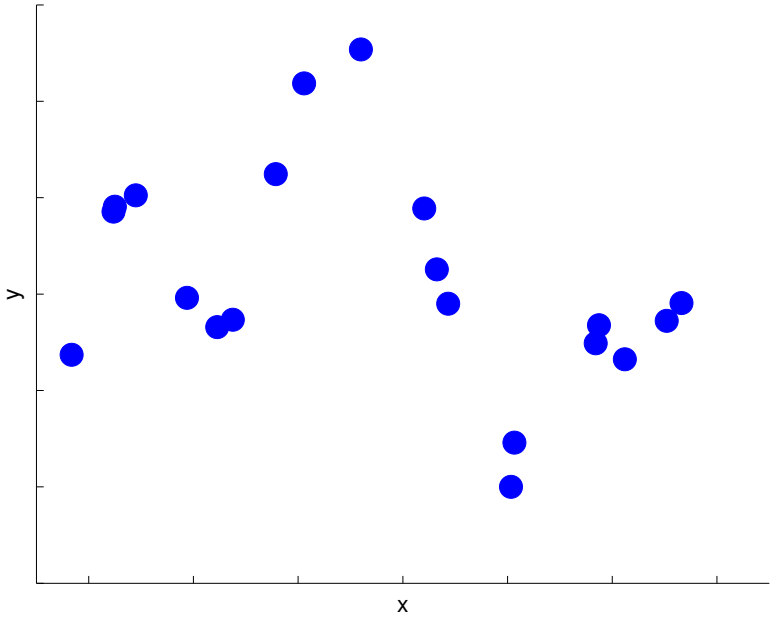
The Prediction Problem

Learn mapping $\mathbf{x} \rightarrow \mathrm{f}(\mathbf{x})$ from observations $\{(\mathbf{x_i}, \mathbf{y_i})\}_{\mathbf{i}=1}^{\mathbf{N}}$.



The Prediction Problem

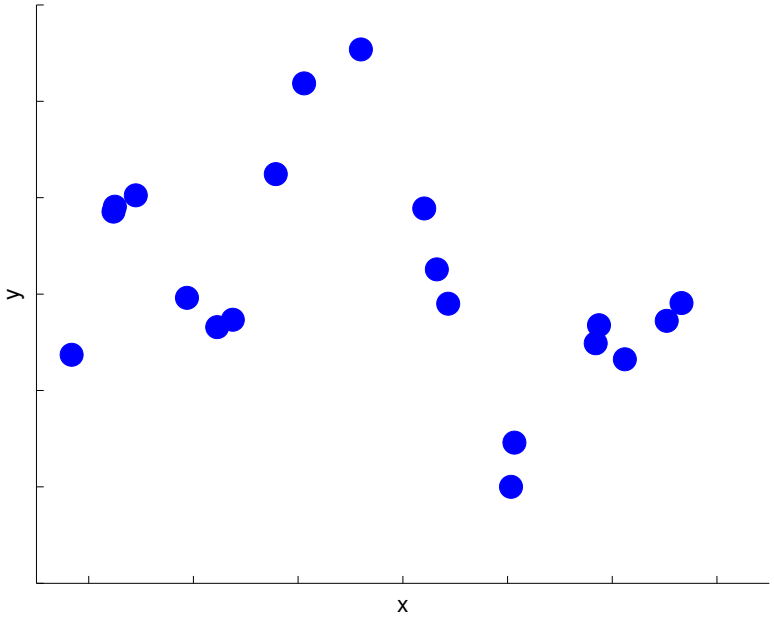
Learn mapping $\mathbf{x} \rightarrow \mathrm{f}(\mathbf{x})$ from observations $\{(\mathbf{x_i}, \mathbf{y_i})\}_{\mathbf{i}=1}^{\mathbf{N}}$.



- What parameterization?

The Prediction Problem

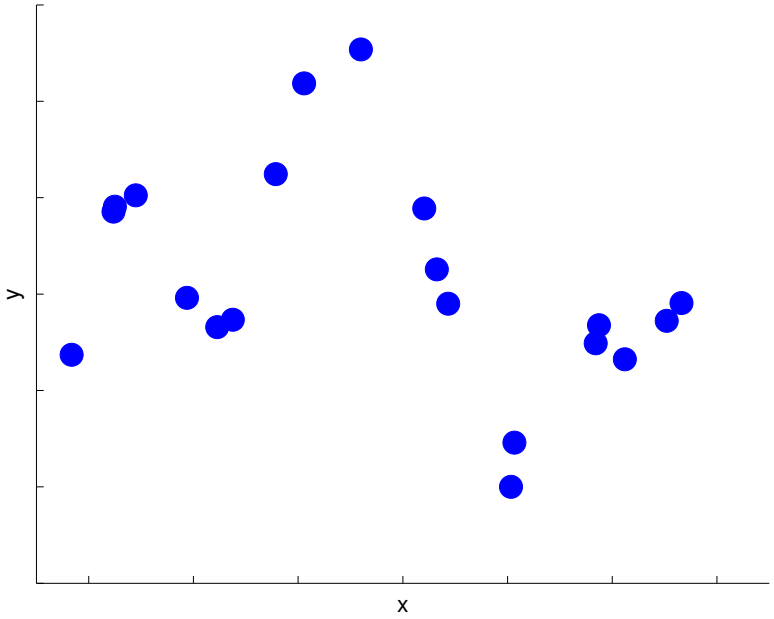
Learn mapping $\mathbf{x} \rightarrow \mathrm{f}(\mathbf{x})$ from observations $\{(\mathbf{x}_i, \mathrm{y}_i)\}_{i=1}^{N}$.



- What parameterization?
- $\mathrm{f}(\mathbf{x}) = \sum_i w_i x_i$

The Prediction Problem

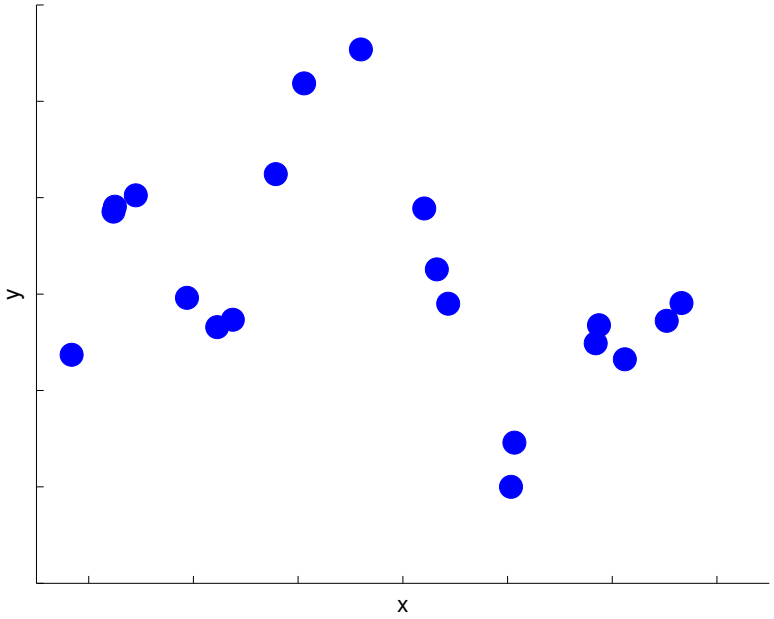
Learn mapping $\mathbf{x} \rightarrow \mathrm{f}(\mathbf{x})$ from observations $\{(\mathbf{x}_i, \mathrm{y}_i)\}_{i=1}^{N}$.



- What parameterization?
- $\mathrm{f}(\mathbf{x}) = \sum_i w_i \phi_i(\mathbf{x})$

The Prediction Problem

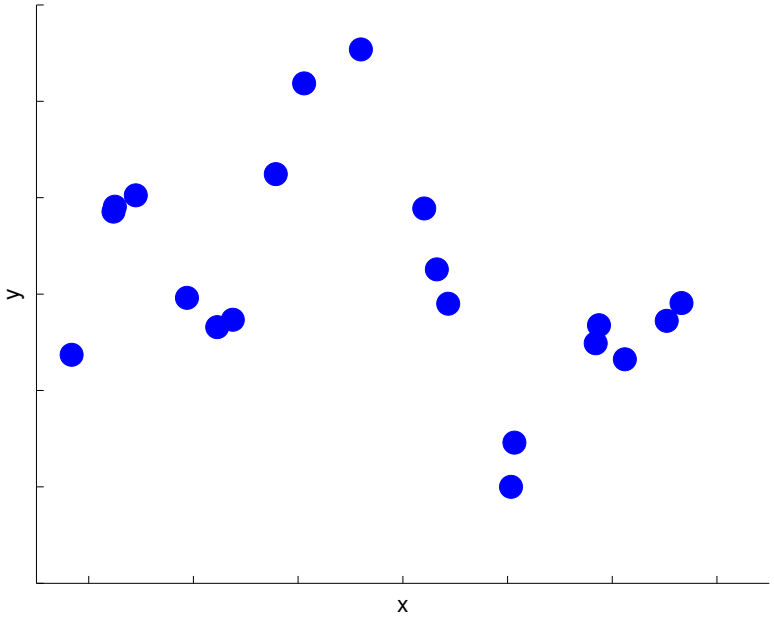
Learn mapping $\mathbf{x} \rightarrow \mathrm{f}(\mathbf{x})$ from observations $\{(\mathbf{x}_i, \mathrm{y}_i)\}_{i=1}^{N}$.



- What parameterization?
- $\mathrm{f}(\mathbf{x}) = \sum_i w_i \phi_i(\mathbf{x})$
- Flexibility v generalization

The Prediction Problem

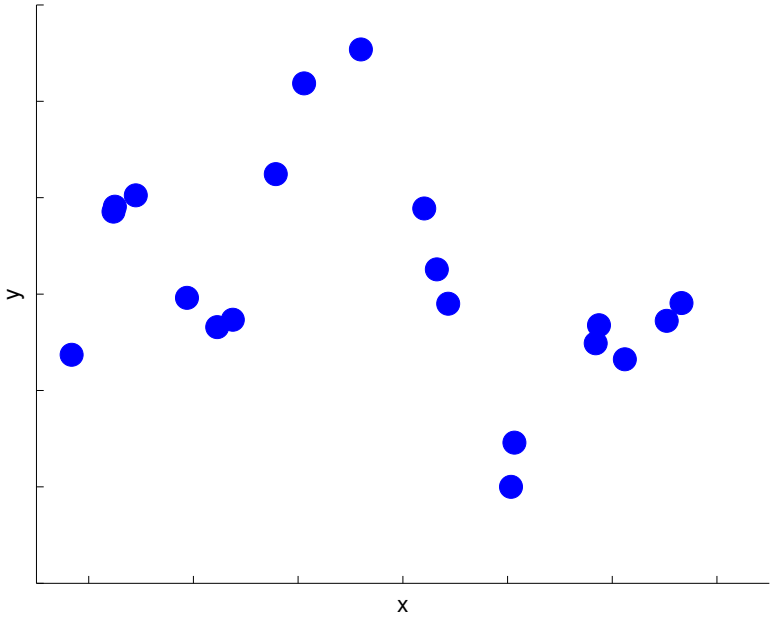
Learn mapping $\mathbf{x} \rightarrow \mathrm{f}(\mathbf{x})$ from observations $\{(\mathbf{x}_i, \mathrm{y}_i)\}_{i=1}^{N}$.



- What parameterization?
- $\mathrm{f}(\mathbf{x}) = \sum_i w_i \phi_i(\mathbf{x})$
- Flexibility v generalization
- What basis functions? How many?

The Prediction Problem

Learn mapping $\mathbf{x} \rightarrow \mathrm{f}(\mathbf{x})$ from observations $\{(\mathbf{x}_i, \mathrm{y}_i)\}_{i=1}^{N}$.

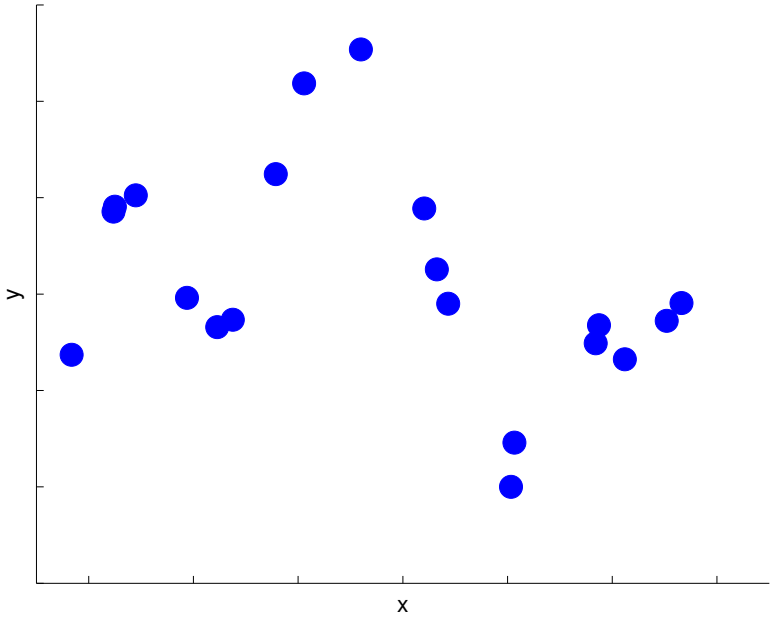


- What about Neural nets?

- What parameterization?
- $\mathrm{f}(\mathbf{x}) = \sum_i w_i \phi_i(\mathbf{x})$
- Flexibility v generalization
- What basis functions? How many?

The Prediction Problem

Learn mapping $\mathbf{x} \rightarrow \mathrm{f}(\mathbf{x})$ from observations $\{(\mathbf{x}_i, \mathrm{y}_i)\}_{i=1}^{N}$.

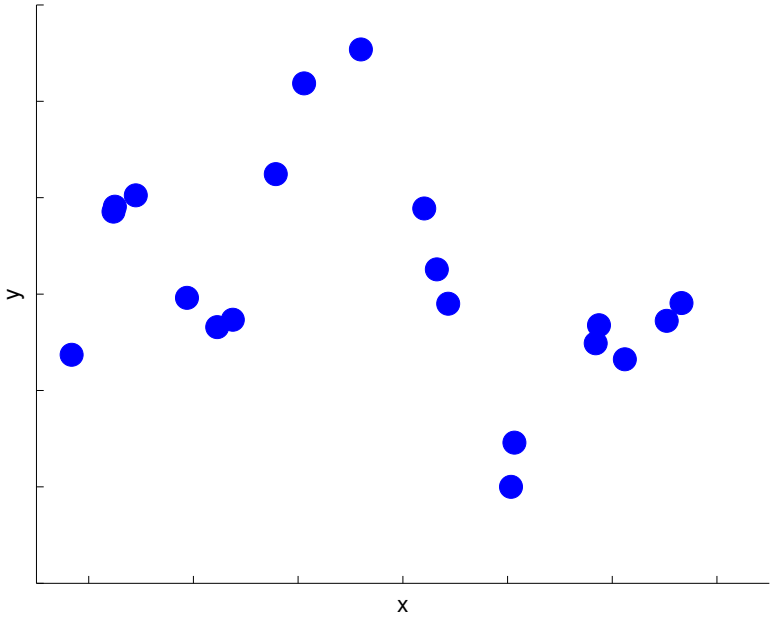


- What parameterization?
- $\mathrm{f}(\mathbf{x}) = \sum_i w_i \phi_i(\mathbf{x})$
- Flexibility v generalization
- What basis functions? How many?

- What about Neural nets?
- How to avoid overfitting? (cf regularization)

The Prediction Problem

Learn mapping $\mathbf{x} \rightarrow \mathrm{f}(\mathbf{x})$ from observations $\{(\mathbf{x}_i, \mathrm{y}_i)\}_{i=1}^{N}$.

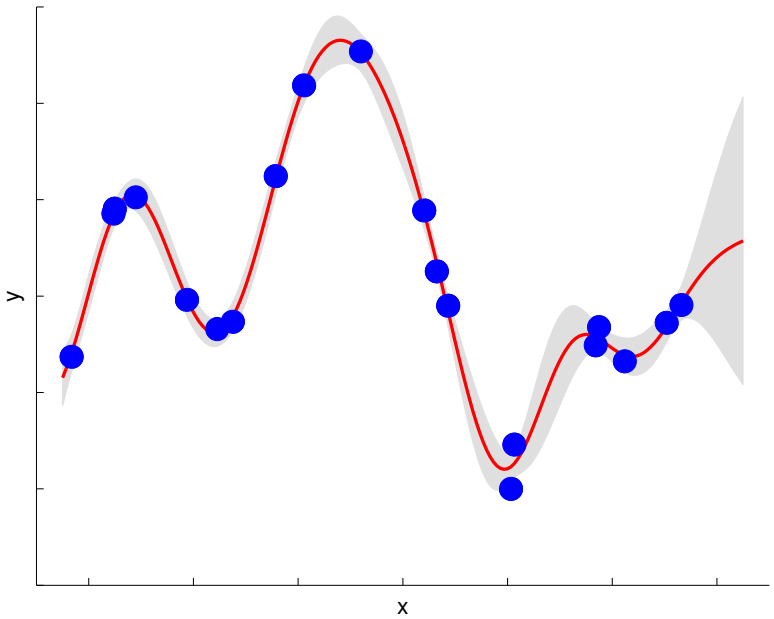


- What parameterization?
- $\mathrm{f}(\mathbf{x}) = \sum_i w_i \phi_i(\mathbf{x})$
- Flexibility v generalization
- What basis functions? How many?

- What about Neural nets?
- How to avoid overfitting? (cf regularization)
- Confidence on our predictions?

The Prediction Problem

Learn mapping $\mathbf{x} \rightarrow \mathrm{f}(\mathbf{x})$ from observations $\{(\mathbf{x}_i, \mathrm{y}_i)\}_{i=1}^{N}$.

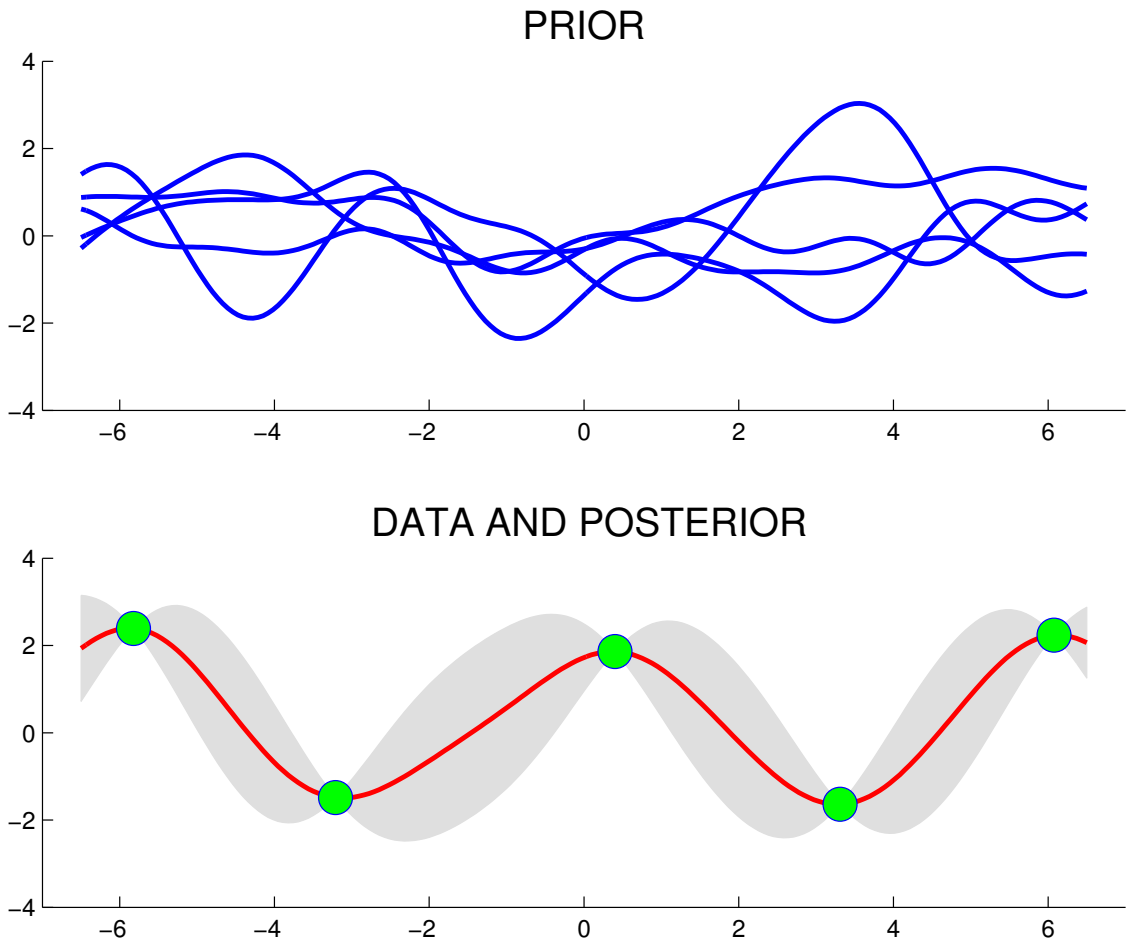


- What parameterization?
- $\mathrm{f}(\mathbf{x}) = \sum_i w_i \phi_i(\mathbf{x})$
- Flexibility v generalization
- What basis functions? How many?

- What about Neural nets?
- How to avoid overfitting? (cf regularization)
- Confidence on our predictions?

We can address these issues in a principled way with Gaussian Processes

Demo



- Smooth functions
- Closeness in input space $\rightarrow$ closeness in output space

Why Gaussian Processes

- Parametric models constrain the class of functions we consider
- Flexibility (no underfitting) due to non-parametric nature

Why Gaussian Processes

- Parametric models constrain the class of functions we consider
- Flexibility (no underfitting) due to non-parametric nature
- Generalization (no overfitting)
- Bayesian, distribution over functions: prior, likelihood, posterior

Why Gaussian Processes

- Parametric models constrain the class of functions we consider
- Flexibility (no underfitting) due to non-parametric nature
- Generalization (no overfitting)
- Bayesian, distribution over functions: prior, likelihood, posterior
- How can we do computations with infinite vectors?
  - "Efficient" Inference due to consistency (Gaussian distributions)

Why Gaussian Processes

- Parametric models constrain the class of functions we consider
- Flexibility (no underfitting) due to non-parametric nature
- Generalization (no overfitting)
- Bayesian, distribution over functions: prior, likelihood, posterior
- How can we do computations with infinite vectors?
  - "Efficient" Inference due to consistency (Gaussian distributions)
- Characteristics of the functions can be learned from data
  - Covariance function: smoothness, stationarity, length-scale
  - Hyperparameter learning

Why Gaussian Processes

- Parametric models constrain the class of functions we consider
- Flexibility (no underfitting) due to non-parametric nature
- Generalization (no overfitting)
- Bayesian, distribution over functions: prior, likelihood, posterior
- How can we do computations with infinite vectors?
  - "Efficient" Inference due to consistency (Gaussian distributions)
- Characteristics of the functions can be learned from data
  - Covariance function: smoothness, stationarity, length-scale
  - Hyperparameter learning
- Many standard regression models are special cases of GPs

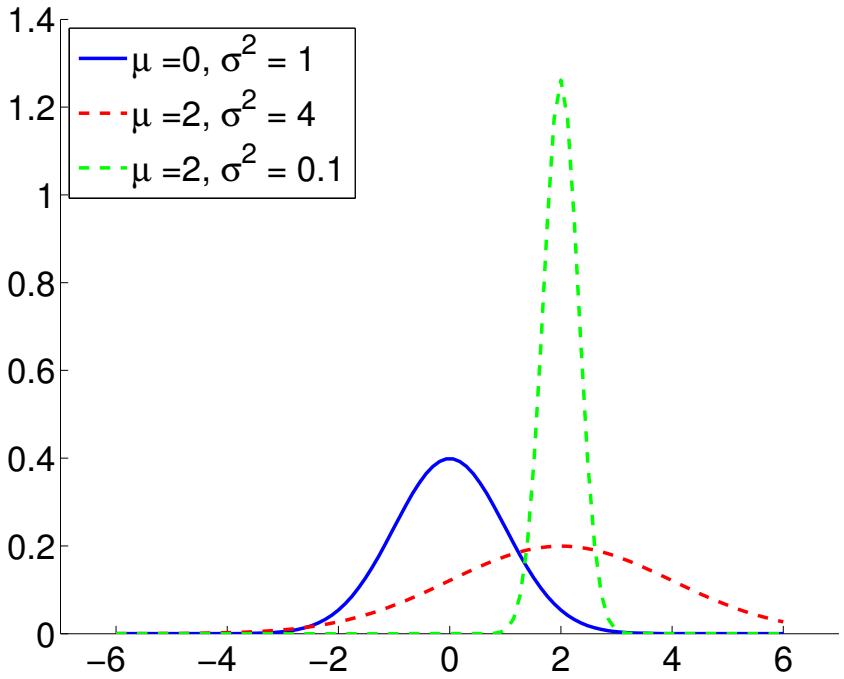
Why Gaussian Processes

- Parametric models constrain the class of functions we consider
- Flexibility (no underfitting) due to non-parametric nature
- Generalization (no overfitting)
- Bayesian, distribution over functions: prior, likelihood, posterior
- How can we do computations with infinite vectors?
  - "Efficient" Inference due to consistency (Gaussian distributions)
- Characteristics of the functions can be learned from data
  - Covariance function: smoothness, stationarity, length-scale
  - Hyperparameter learning
- Many standard regression models are special cases of GPs
- GP models also applicable to non-regression settings

Outline

The Gaussian Distribution

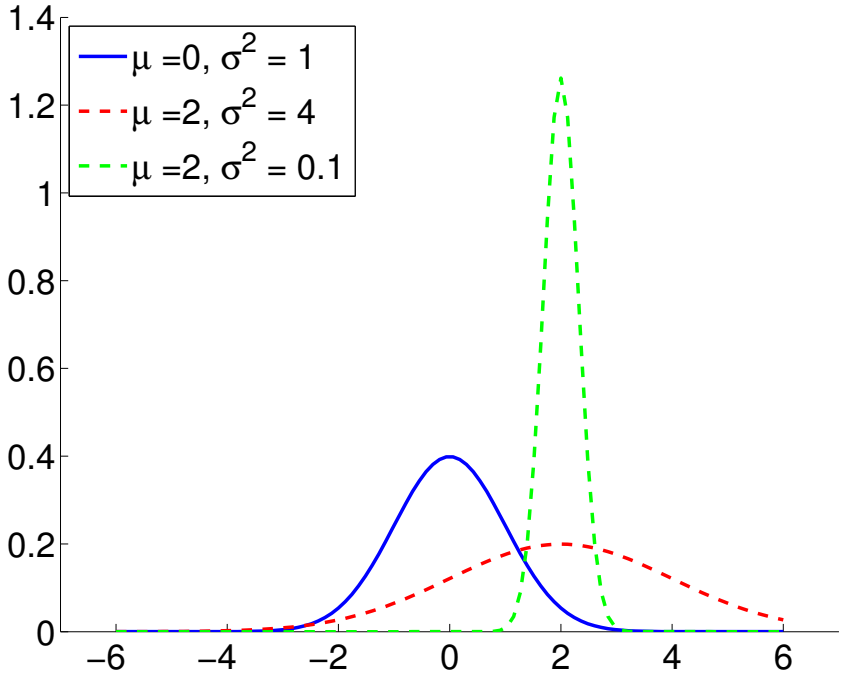
1D Example

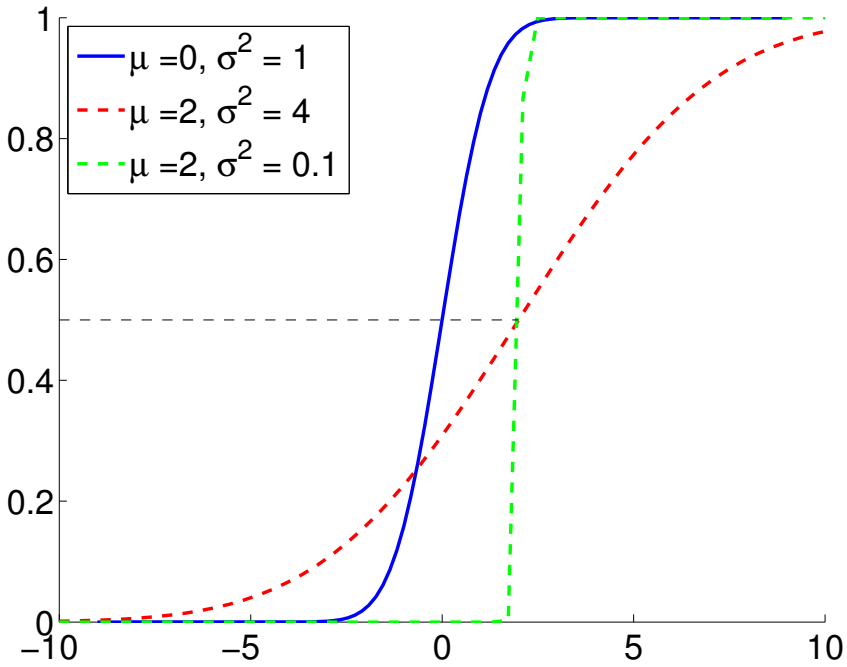


$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$

The Gaussian Distribution

1D Example



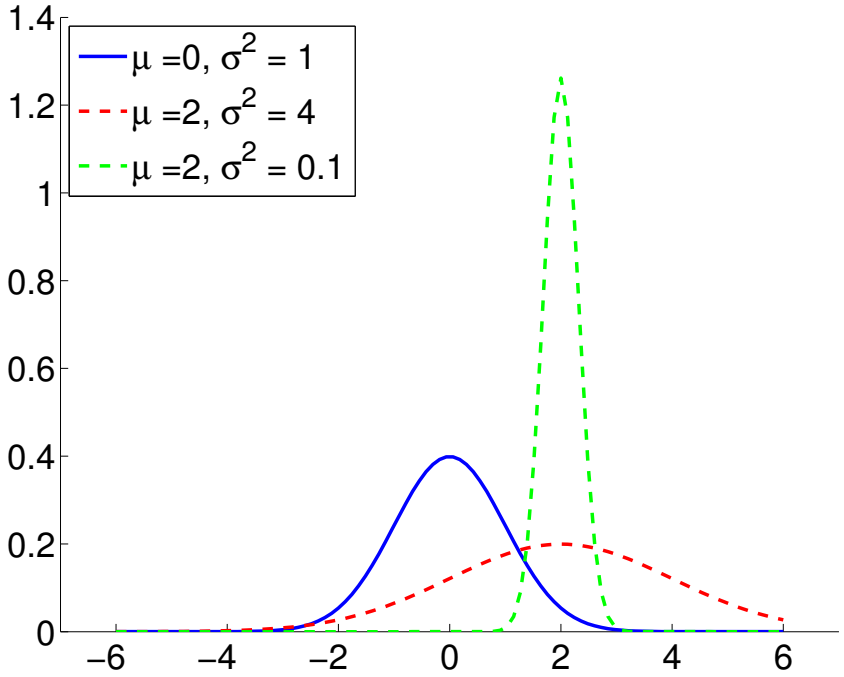


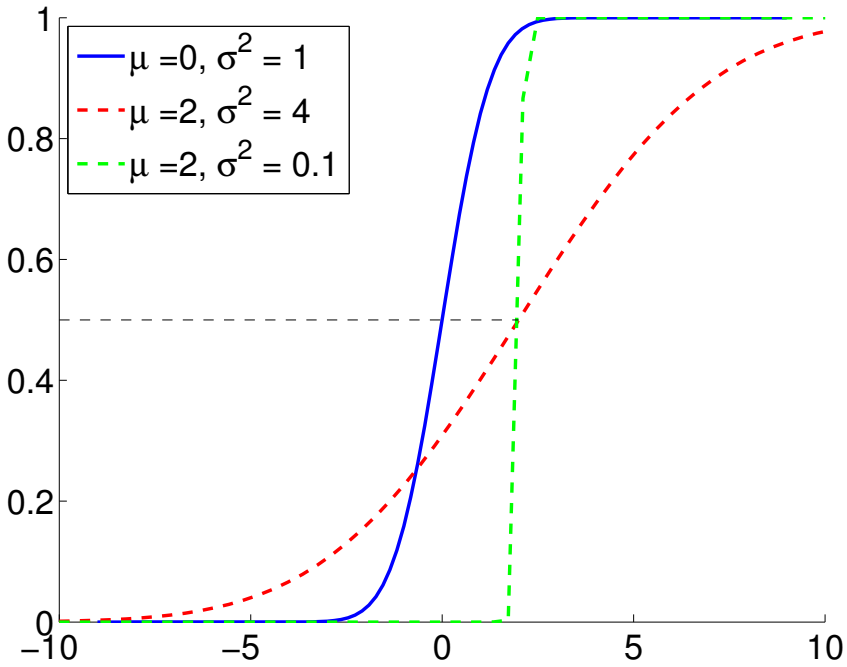
$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$

$$F(x) = \int_{-\infty}^{x} \mathcal{N}(z|\mu, \sigma^2)\,dz$$

The Gaussian Distribution

1D Example





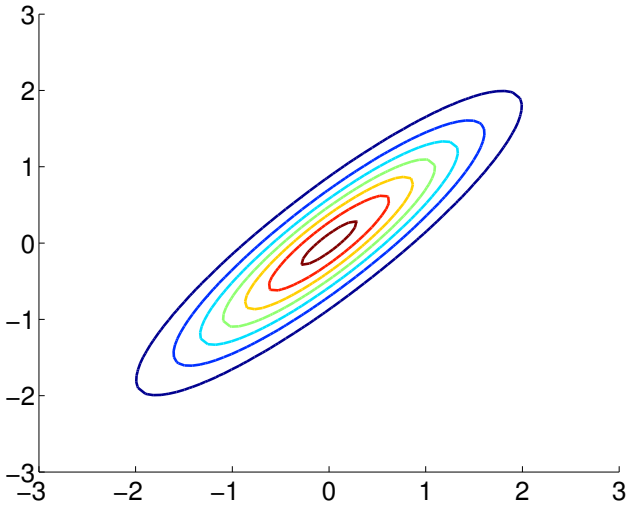
$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$

$$F(x) = \int_{-\infty}^{x} \mathcal{N}(z|\mu, \sigma^2)\,dz$$

In general: $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{|2\pi\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$

The Gaussian Distribution

2D Example

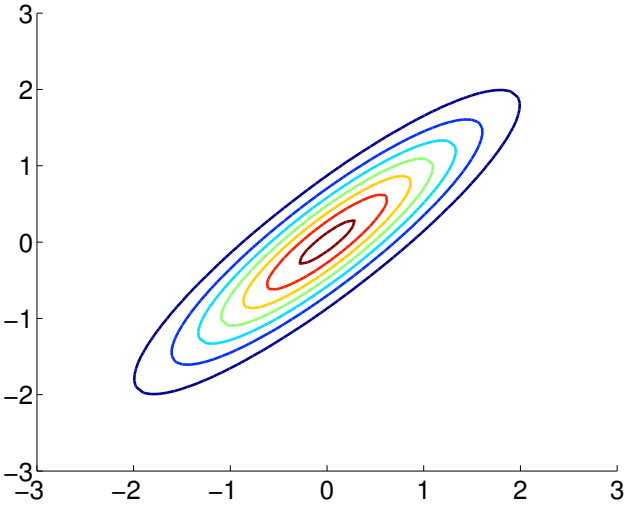


$$p(x_1, x_2) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Joint

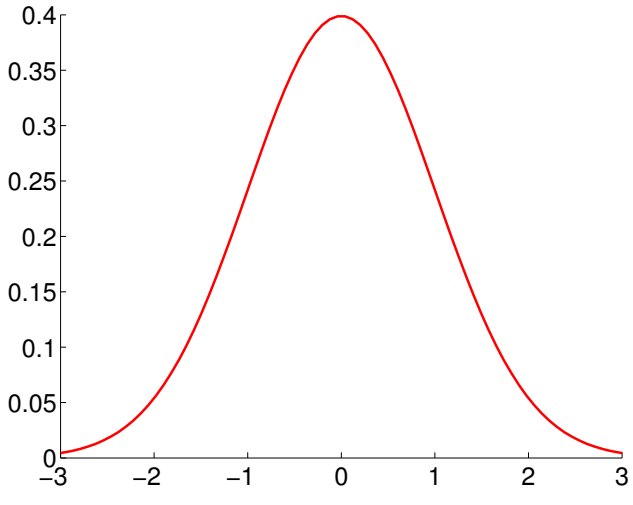
The Gaussian Distribution

2D Example



$$p(x_1, x_2) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Joint

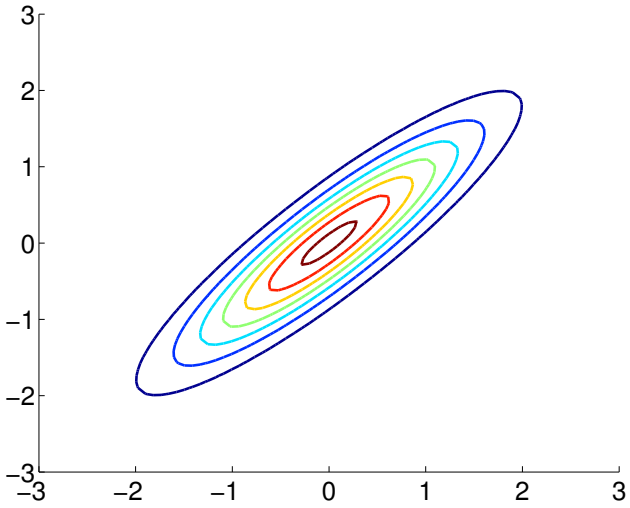


$$p(x1)$$

Marginal

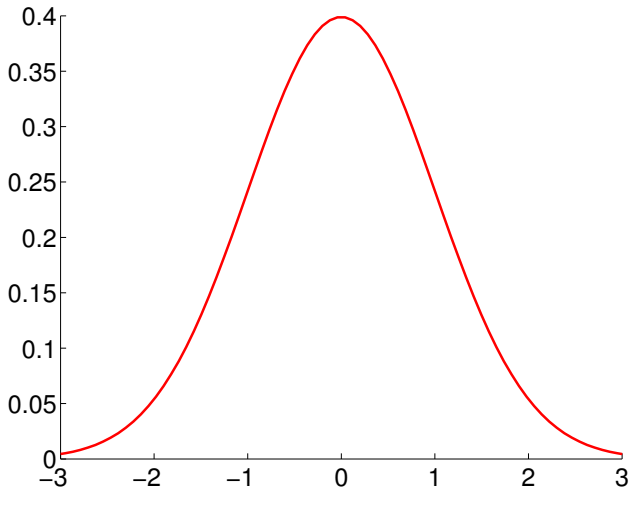
The Gaussian Distribution

2D Example



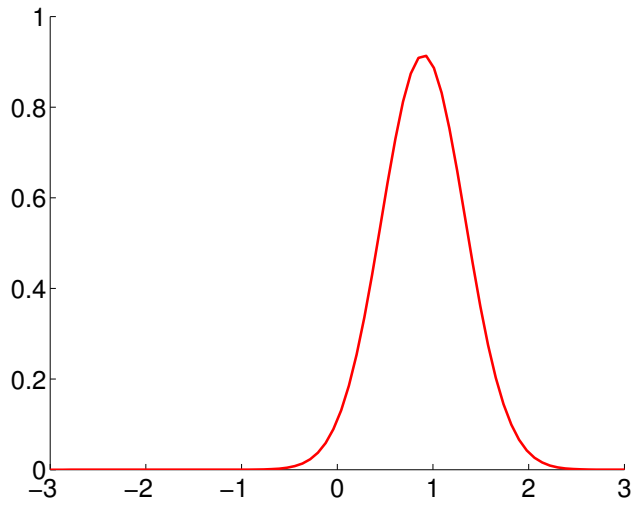
$$p(x_1, x_2) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Joint



$$p(x1)$$

Marginal

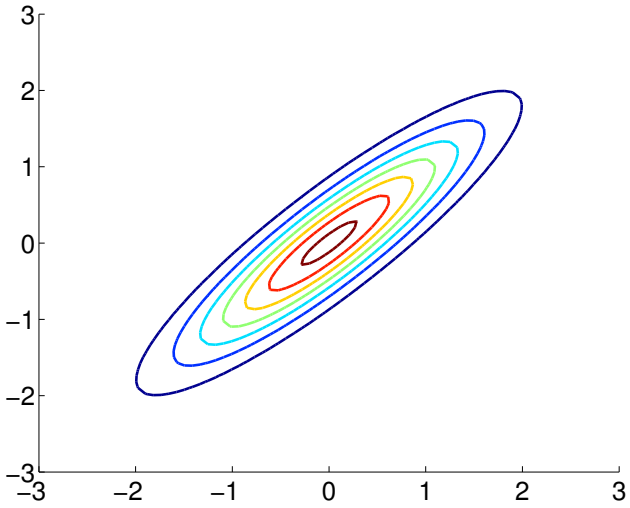


$$p(x_1|x_2)$$

Conditional

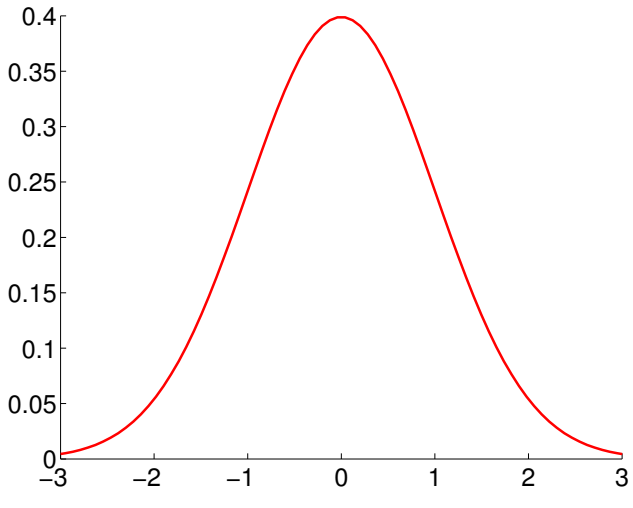
The Gaussian Distribution

2D Example



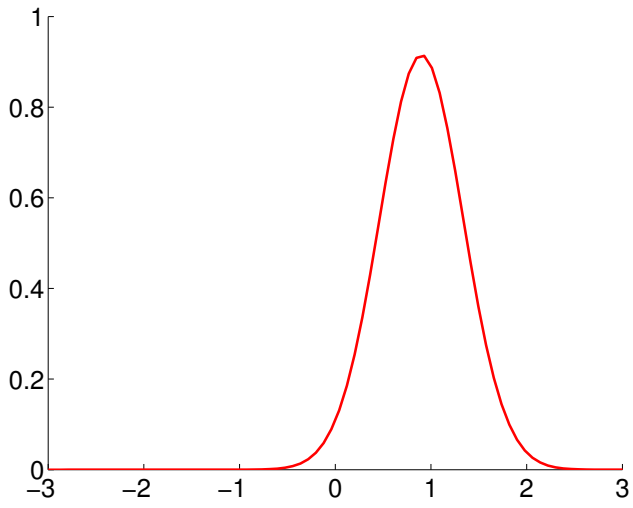
$p(x_1, x_2) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

Joint



$p(x1)$

Marginal



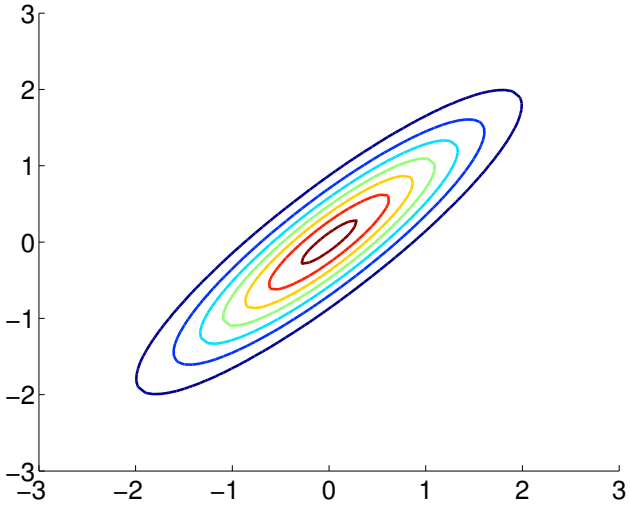
$p(x_1|x_2)$

Conditional

The marginal and the conditional distributions are also Gaussians:

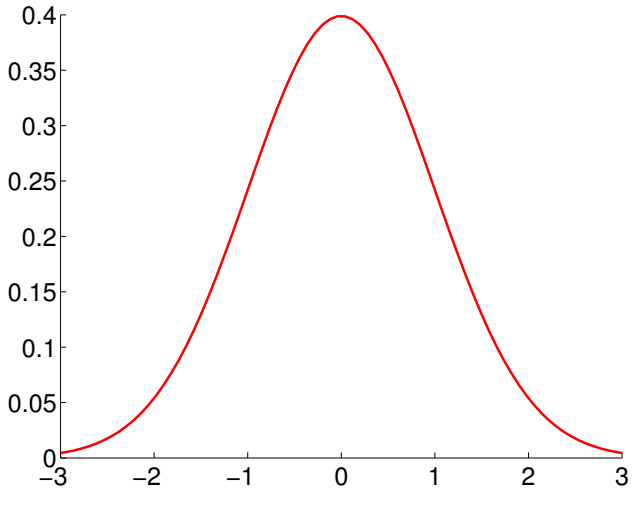
The Gaussian Distribution

2D Example



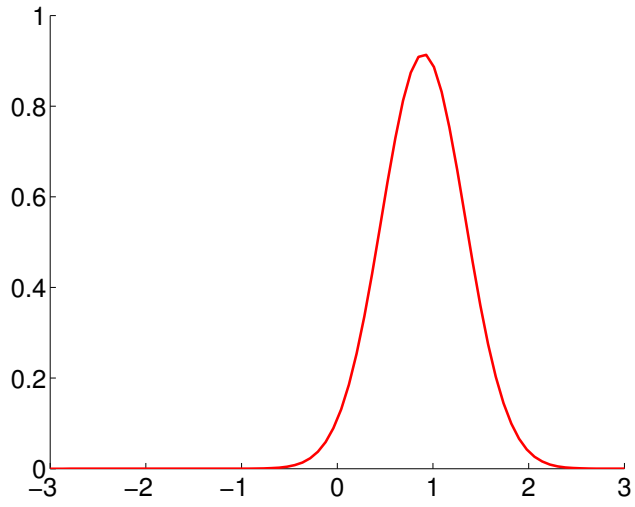
$$p(x_1, x_2) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Joint



$$p(x1)$$

Marginal



$$p(x_1|x_2)$$

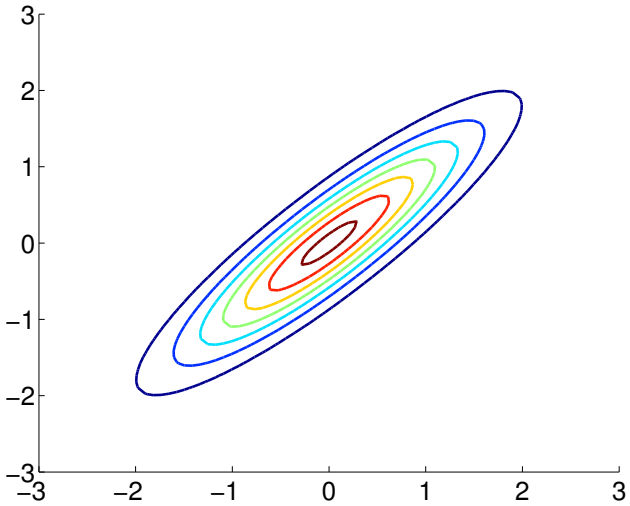
Conditional

The marginal and the conditional distributions are also Gaussians:

$$\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{12}^{\mathsf{T}} & \boldsymbol{\Sigma}_{22} \end{bmatrix} \right)$$

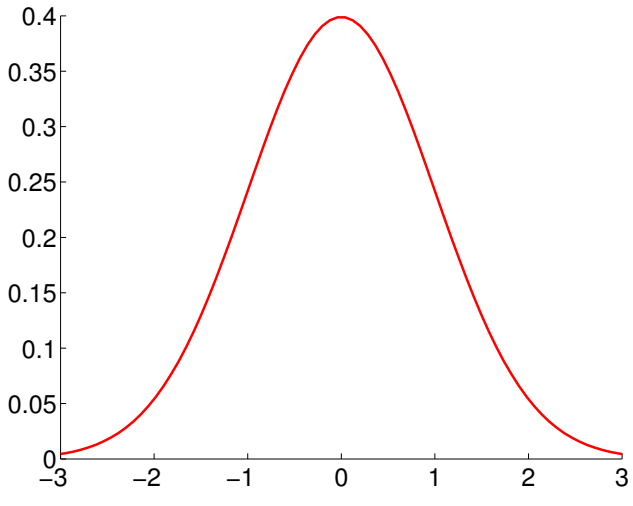
The Gaussian Distribution

2D Example



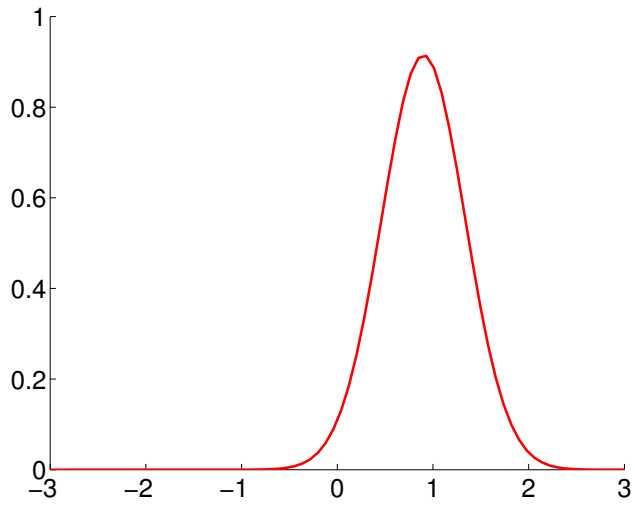
$$p(x_1, x_2) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Joint



$$p(x1)$$

Marginal



$$p(x_1|x_2)$$

Conditional

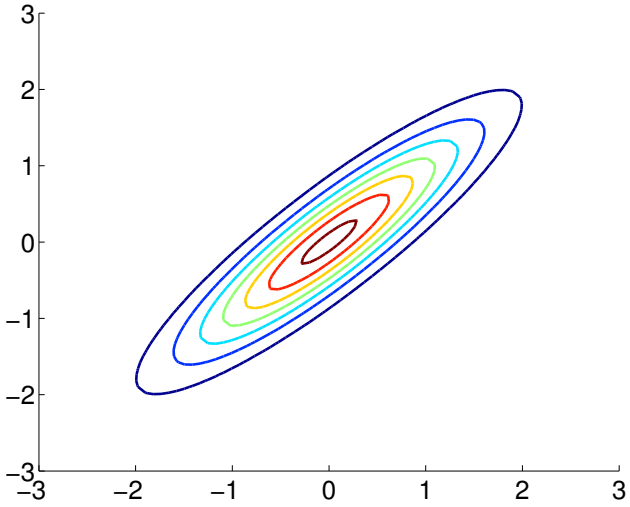
The marginal and the conditional distributions are also Gaussians:

$$\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{12}^{\mathsf{T}} & \boldsymbol{\Sigma}_{22} \end{bmatrix} \right)$$

$$\mathbf{x}_1 \sim \mathcal{N}(\mathbf{x}_1 | \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11})$$

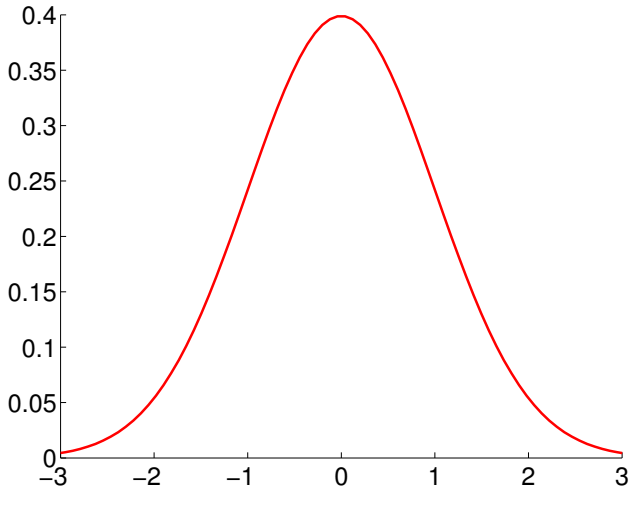
The Gaussian Distribution

2D Example



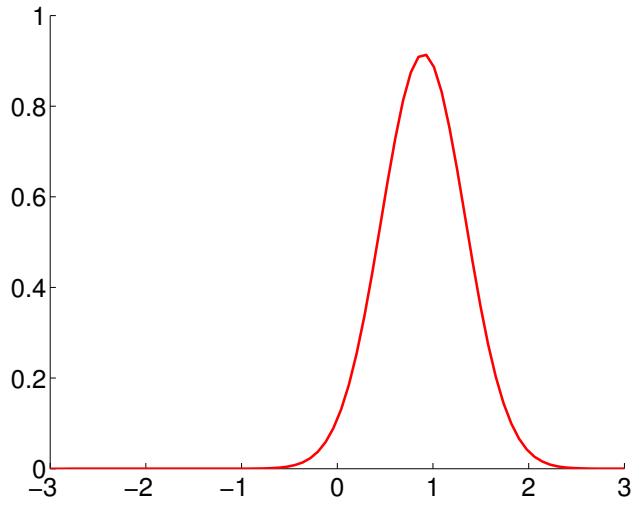
$$p(x_1, x_2) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Joint



$$p(x1)$$

Marginal



$$p(x_1|x_2)$$

Conditional

The marginal and the conditional distributions are also Gaussians:

$$\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{12}^{\mathsf{T}} & \boldsymbol{\Sigma}_{22} \end{bmatrix} \right)$$

$$\mathbf{x}_1 \sim \mathcal{N}(\mathbf{x}_1 | \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11})$$

$$\mathbf{x}_1 | \mathbf{x}_2 \sim \mathcal{N}(\mathbf{x}_1 | \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{x}_2 - \boldsymbol{\mu}_2), \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{12}^{\mathsf{T}})$$

The Gaussian Distribution

Covariance and Precision Matrices

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{|2\pi\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

$\boldsymbol{\Sigma}$: is the covariance matrix

$\boldsymbol{\Sigma}^{-1}$: is the precision matrix

The Gaussian Distribution

Covariance and Precision Matrices

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{|2\pi\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

$\boldsymbol{\Sigma}$: is the covariance matrix

$\boldsymbol{\Sigma}^{-1}$: is the precision matrix

- An entry $\boldsymbol{\Sigma}^{-1}_{ij} = 0$ indicates that the variables i and j are conditionally independent given all the other variables.

The Gaussian Distribution

Covariance and Precision Matrices

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{|2\pi\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

$\boldsymbol{\Sigma}$: is the covariance matrix

$\boldsymbol{\Sigma}^{-1}$: is the precision matrix

- An entry $\boldsymbol{\Sigma}^{-1}_{ij} = 0$ indicates that the variables i and j are conditionally independent given all the other variables.
- An entry $\boldsymbol{\Sigma}_{ij} = 0$ indicates that the variables i and j are marginally independent given all the other variables.

The Gaussian Distribution

Covariance and Precision Matrices

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{|2\pi\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

$\boldsymbol{\Sigma}$: is the covariance matrix

$\boldsymbol{\Sigma}^{-1}$: is the precision matrix

- An entry $\boldsymbol{\Sigma}^{-1}_{ij} = 0$ indicates that the variables i and j are conditionally independent given all the other variables.
- An entry $\boldsymbol{\Sigma}_{ij} = 0$ indicates that the variables i and j are marginally independent given all the other variables.
- Marginalizing out a variable leaves $\boldsymbol{\Sigma}$ unchanged but changes $\boldsymbol{\Sigma}^{-1}$.
  - This is crucial when parameterizing a Gaussian process.

Gaussian Quiz

The Standard Linear Regression Model

Notation and Settings

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

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

$$\text{Goal} : \mathbf{x} \overset{\mathbf{f}(\mathbf{x})}{\rightarrow} \mathbf{y}$$

The Standard Linear Regression Model

Notation and Settings

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

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

$$\text{Goal} : \mathbf{x} \overset{f(\mathbf{x})}{\rightarrow} \mathbf{y}$$

$$\begin{aligned}
\text{Model} \quad & f(\mathbf{x}) = \sum_{i=1}^{D} w_i x_i && = \mathbf{w}^T \mathbf{x} \\
\text{Noise} \quad & y = f(\mathbf{x}) + \eta && \text{with } \eta \sim \mathcal{N}(\eta|0, \sigma^2) \\
\text{Likelihood} \quad & y|f(\mathbf{x}) \sim \mathcal{N}(y|f(\mathbf{x}), \sigma^2) && = \mathcal{N}(y|\mathbf{w}^T \mathbf{x}, \sigma^2)
\end{aligned}$$

The Standard Linear Regression Model

Notation and Settings

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

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

$$\text{Goal} : \mathbf{x} \overset{f(\mathbf{x})}{\rightarrow} \mathbf{y}$$

$$\begin{aligned}
\text{Model} \quad & f(\mathbf{x}) = \textstyle\sum_{i=1}^{D} w_i x_i && = \mathbf{w}^T\mathbf{x} \\
\text{Noise} \quad & y = f(\mathbf{x}) + \eta && \text{with } \eta \sim \mathcal{N}(\eta|0, \sigma^2) \\
\text{Likelihood} \quad & y|f(\mathbf{x}) \sim \mathcal{N}(y|f(\mathbf{x}), \sigma^2) && = \mathcal{N}(y|\mathbf{w}^T\mathbf{x}, \sigma^2)
\end{aligned}$$

Thus, the data-likelihood is given by:

$$\begin{aligned}
p(\mathbf{y}|\mathbf{X}, \mathbf{w}) &= \prod_{i=1}^{N} p(y_i|\mathbf{x}_i, \mathbf{w}) = \prod_{i=1}^{N} \mathcal{N}(y_i|\mathbf{w}^T\mathbf{x}_i, \sigma^2) \\
&= \mathcal{N}(\mathbf{y}|\mathbf{X}^T\mathbf{w}, \sigma^2\mathbf{I})
\end{aligned}$$

The Standard Linear Regression Model

Notation and Settings

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

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

$$\text{Goal} : \mathbf{x} \overset{f(\mathbf{x})}{\rightarrow} \mathbf{y}$$

$$\begin{aligned}
\text{Model} \quad & f(\mathbf{x}) = \textstyle\sum_{i=1}^{D} w_i x_i && = \mathbf{w}^T\mathbf{x} \\
\text{Noise} \quad & y = f(\mathbf{x}) + \eta && \text{with } \eta \sim \mathcal{N}(\eta|0, \sigma^2) \\
\text{Likelihood} \quad & y|f(\mathbf{x}) \sim \mathcal{N}(y|f(\mathbf{x}), \sigma^2) && = \mathcal{N}(y|\mathbf{w}^T\mathbf{x}, \sigma^2)
\end{aligned}$$

Thus, the data-likelihood is given by:

$$\begin{aligned}
p(\mathbf{y}|\mathbf{X}, \mathbf{w}) &= \prod_{i=1}^{N} p(y_i|\mathbf{x}_i, \mathbf{w}) = \prod_{i=1}^{N} \mathcal{N}(y_i|\mathbf{w}^T\mathbf{x}_i, \sigma^2) \\
&= \mathcal{N}(\mathbf{y}|\mathbf{X}^T\mathbf{w}, \sigma^2\mathbf{I})
\end{aligned}$$

We need do to inference on $\mathbf{w}$.

Bayesian Linear Regression

Posterior Distribution

Consider a zero-mean Gaussian prior over the weights:

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

Bayesian Linear Regression

Posterior Distribution

Consider a zero-mean Gaussian prior over the weights:

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

Then the posterior distribution over the weights is given by:

$$\begin{aligned} \mathrm{p}(\mathbf{w}|\mathbf{X}, \mathbf{y}) &= \frac{\mathrm{p}(\mathbf{w})\ \mathrm{p}(\mathbf{y}|\mathbf{X}, \mathbf{w})}{\mathrm{p}(\mathbf{y}|\mathbf{X})} \\ &= \mathcal{N}(\mathbf{w}|\bar{\mathbf{w}}, \mathbf{A}^{-1}) \end{aligned}$$

where $\bar{\mathbf{w}} = \frac{1}{\sigma^2}\mathbf{A}^{-1}\mathbf{X}\mathbf{y}$, and $\mathbf{A} = (\frac{1}{\sigma^2}\mathbf{X}\mathbf{X}^{\mathsf{T}} + \boldsymbol{\Sigma}_w^{-1})$.

Bayesian Linear Regression

Posterior Distribution

Consider a zero-mean Gaussian prior over the weights:

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

Then the posterior distribution over the weights is given by:

$$\begin{aligned} \mathrm{p}(\mathbf{w}|\mathbf{X}, \mathbf{y}) &= \frac{\mathrm{p}(\mathbf{w})\ \mathrm{p}(\mathbf{y}|\mathbf{X}, \mathbf{w})}{\mathrm{p}(\mathbf{y}|\mathbf{X})} \\ &= \mathcal{N}(\mathbf{w}|\bar{\mathbf{w}}, \mathbf{A}^{-1}) \end{aligned}$$

where $\bar{\mathbf{w}} = \frac{1}{\sigma^2}\mathbf{A}^{-1}\mathbf{X}\mathbf{y}$, and $\mathbf{A} = (\frac{1}{\sigma^2}\mathbf{X}\mathbf{X}^{\mathsf{T}} + \mathbf{\Sigma}_w^{-1})$.

- Mean of posterior is equal to its mode

Bayesian Linear Regression

Posterior Distribution

Consider a zero-mean Gaussian prior over the weights:

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

Then the posterior distribution over the weights is given by:

$$\begin{aligned} \mathrm{p}(\mathbf{w}|\mathbf{X}, \mathbf{y}) &= \frac{\mathrm{p}(\mathbf{w})\ \mathrm{p}(\mathbf{y}|\mathbf{X}, \mathbf{w})}{\mathrm{p}(\mathbf{y}|\mathbf{X})} \\ &= \mathcal{N}(\mathbf{w}|\bar{\mathbf{w}}, \mathbf{A}^{-1}) \end{aligned}$$

where $\bar{\mathbf{w}} = \frac{1}{\sigma^2}\mathbf{A}^{-1}\mathbf{X}\mathbf{y}$, and $\mathbf{A} = (\frac{1}{\sigma^2}\mathbf{X}\mathbf{X}^{\mathsf{T}} + \mathbf{\Sigma}_w^{-1})$.

- Mean of posterior is equal to its mode
- MAP solution (non-Bayesian): negative log prior as penalty term

Bayesian Linear Regression

Posterior Distribution

Consider a zero-mean Gaussian prior over the weights:

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

Then the posterior distribution over the weights is given by:

$$\begin{aligned} \mathrm{p}(\mathbf{w}|\mathbf{X}, \mathbf{y}) &= \frac{\mathrm{p}(\mathbf{w})\ \mathrm{p}(\mathbf{y}|\mathbf{X}, \mathbf{w})}{\mathrm{p}(\mathbf{y}|\mathbf{X})} \\ &= \mathcal{N}(\mathbf{w}|\bar{\mathbf{w}}, \mathbf{A}^{-1}) \end{aligned}$$

where $\bar{\mathbf{w}} = \frac{1}{\sigma^2}\mathbf{A}^{-1}\mathbf{X}\mathbf{y}$, and $\mathbf{A} = (\frac{1}{\sigma^2}\mathbf{X}\mathbf{X}^{\mathsf{T}} + \mathbf{\Sigma}_w^{-1})$.

- Mean of posterior is equal to its mode
- MAP solution (non-Bayesian): negative log prior as penalty term
- This penalized maximum likelihood is known as *ridge regression*
  - Consider $\mathbf{\Sigma}_w = \lambda\mathbf{I}$ Then :

$$\bar{\mathbf{w}} = (\mathbf{X}\mathbf{X}^{\mathsf{T}} + \frac{1}{\lambda}\sigma^2\mathbf{I})^{-1}\mathbf{X}\mathbf{y}$$

Bayesian Linear Regression

Predictive Distribution

We are interested in making predictions at a new test point $\mathbf{x}_*$

- In fact we obtain the predictive distribution by averaging over all possible parameter values (weighted by their posterior probabilities):

$$\mathrm{p}(\mathrm{f}_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \int \mathrm{p}(\mathrm{f}_*|\mathbf{x}_*, \mathbf{w})\mathrm{p}(\mathbf{w}|\mathbf{X}, \mathbf{y})\ \mathrm{d}\mathbf{w} = \mathcal{N}(\mathrm{f}_*|\mathbf{x}_*^\mathsf{T}\bar{\mathbf{w}}, \mathbf{x}_*^\mathsf{T}\mathbf{A}^{-1}\mathbf{x}_*)$$

  - Predictive mean: linear combination of weights' posterior mean
  - Predictive variance: grows with the magnitude of the test point

Bayesian Linear Regression

Predictive Distribution

We are interested in making predictions at a new test point $\mathbf{x}_*$

- In fact we obtain the predictive distribution by averaging over all possible parameter values (weighted by their posterior probabilities):

$$\mathrm{p}(\mathrm{f}_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \int \mathrm{p}(\mathrm{f}_*|\mathbf{x}_*, \mathbf{w})\mathrm{p}(\mathbf{w}|\mathbf{X}, \mathbf{y})\ \mathrm{d}\mathbf{w} = \mathcal{N}(\mathrm{f}_*|\mathbf{x}_*^\mathsf{T}\bar{\mathbf{w}}, \mathbf{x}_*^\mathsf{T}\mathbf{A}^{-1}\mathbf{x}_*)$$

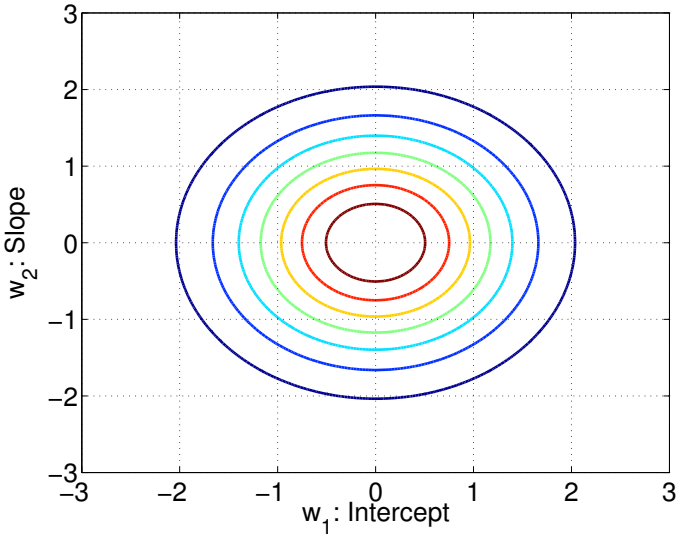
  - Predictive mean: linear combination of weights' posterior mean
  - Predictive variance: grows with the magnitude of the test point

- Point predictions: Need to consider the expected loss (or *risk*):

$$\mathrm{y}_{\mathrm{opt}} = \underset{\mathrm{y}_{\mathrm{pred}}}{\operatorname{argmin}} \int \mathcal{L}(\mathrm{f}_*, \mathrm{y}_{\mathrm{pred}})\mathrm{p}(\mathrm{f}_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y})\mathrm{d}\mathrm{f}_*$$

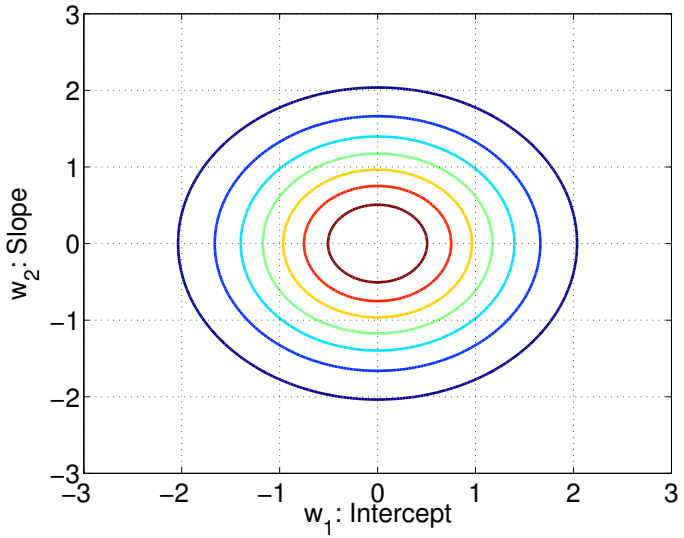
  - e.g. Square loss $\mathcal{L} = (\mathrm{y}_{\mathrm{pred}} - \mathrm{f}_*)^2$
  - c.f. Empirical risk minimization (ERM)

Bayesian Linear Regression Example

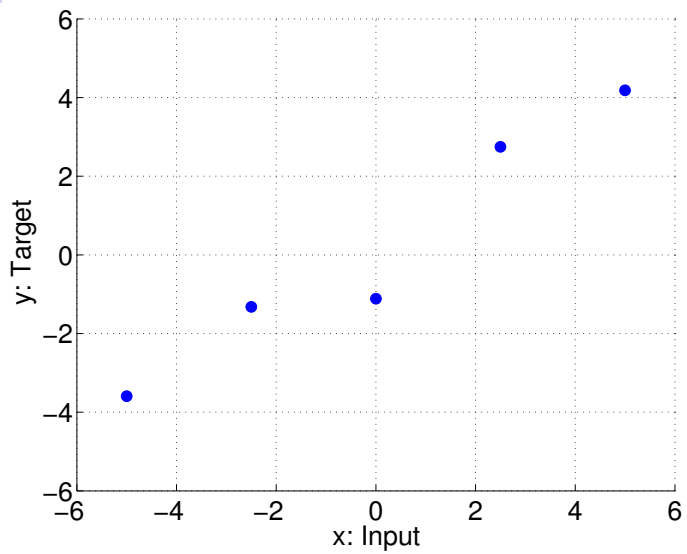


Prior Weights

Bayesian Linear Regression Example

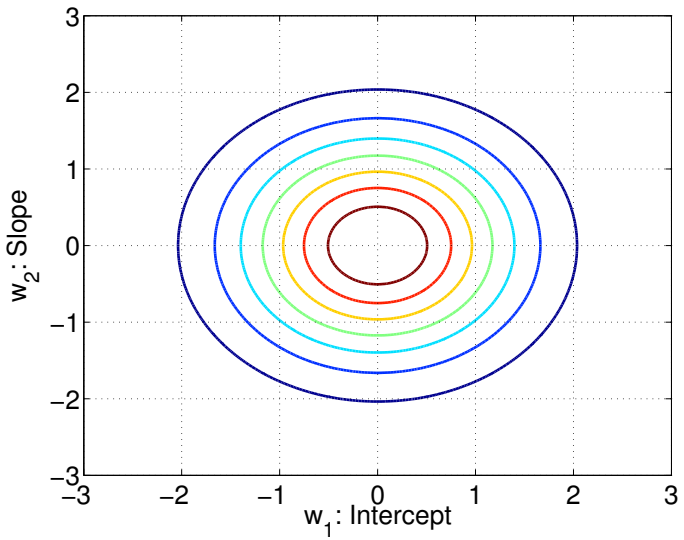


Prior Weights

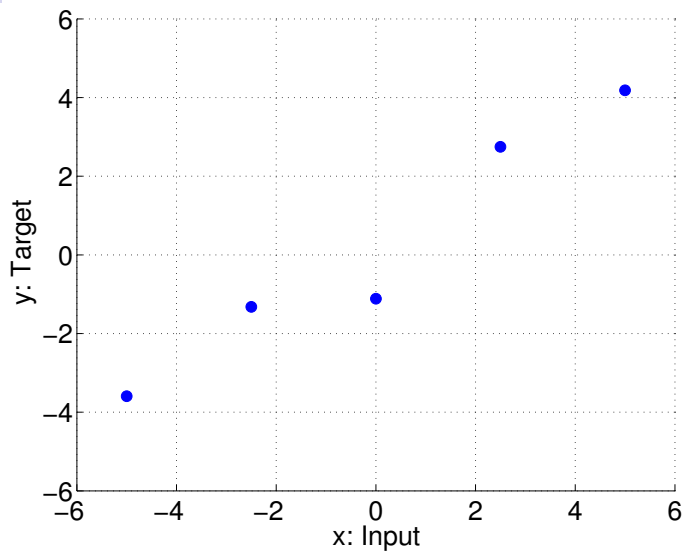


Observed Data

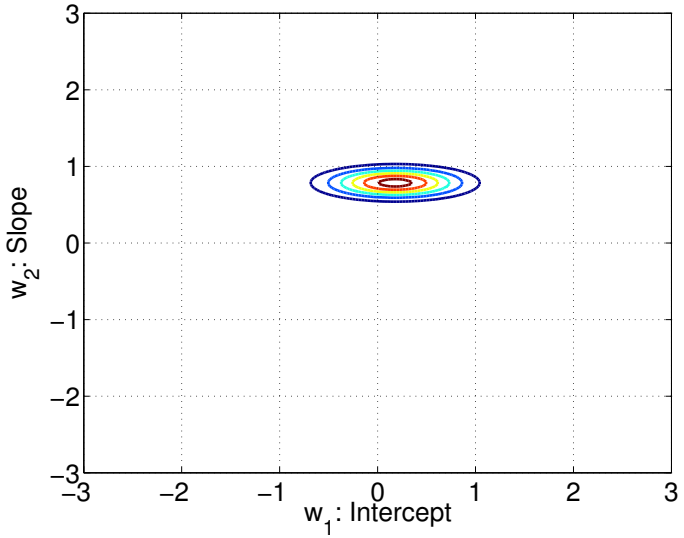
Bayesian Linear Regression Example



Prior Weights

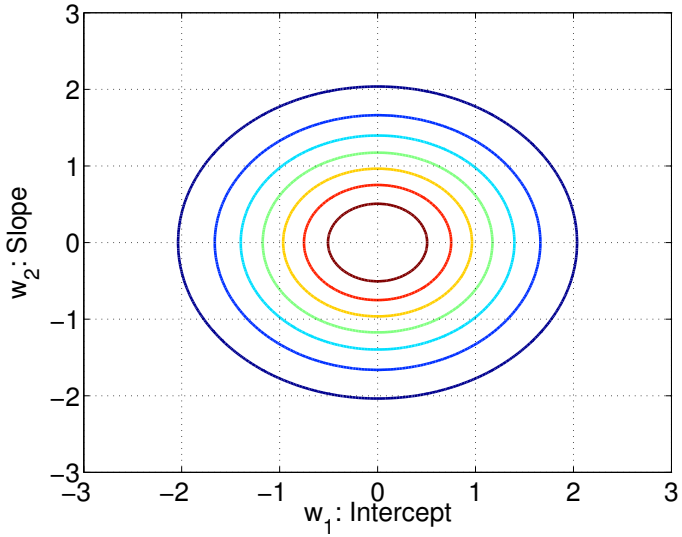


Observed Data

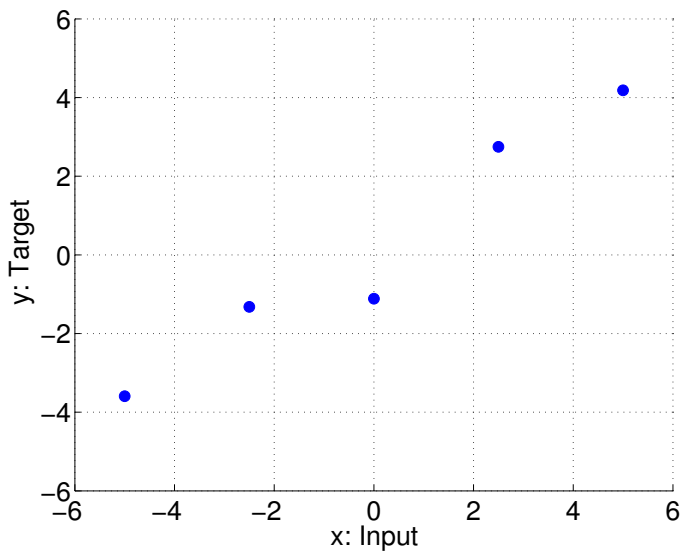


Likelihood

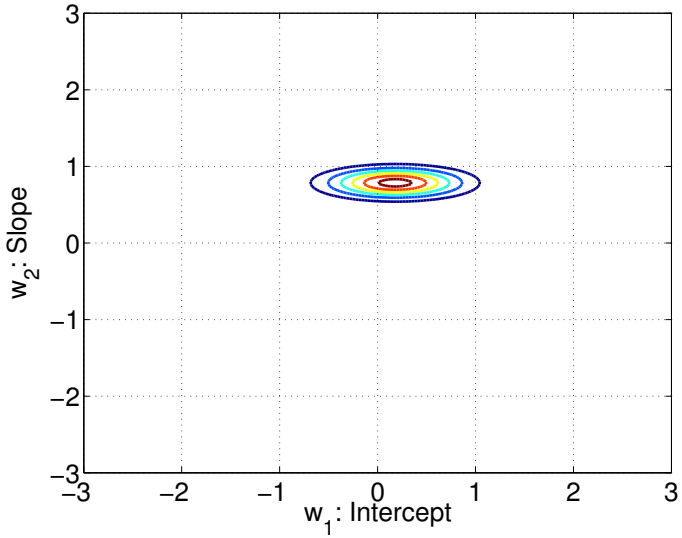
Bayesian Linear Regression Example



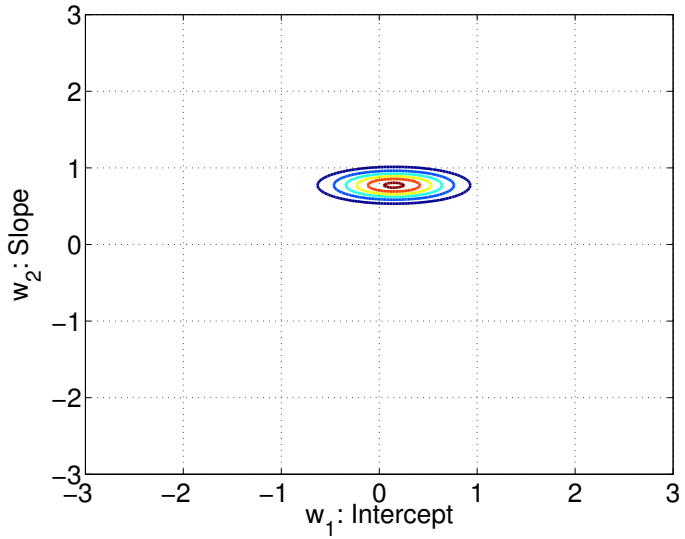
Prior Weights



Observed Data

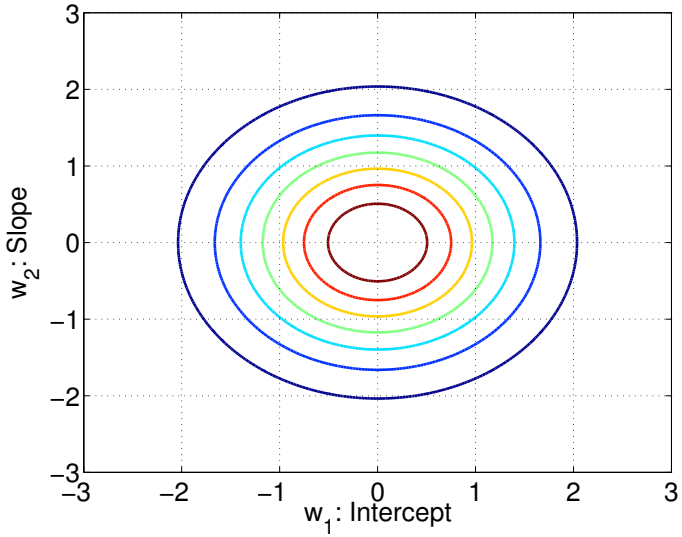


Likelihood

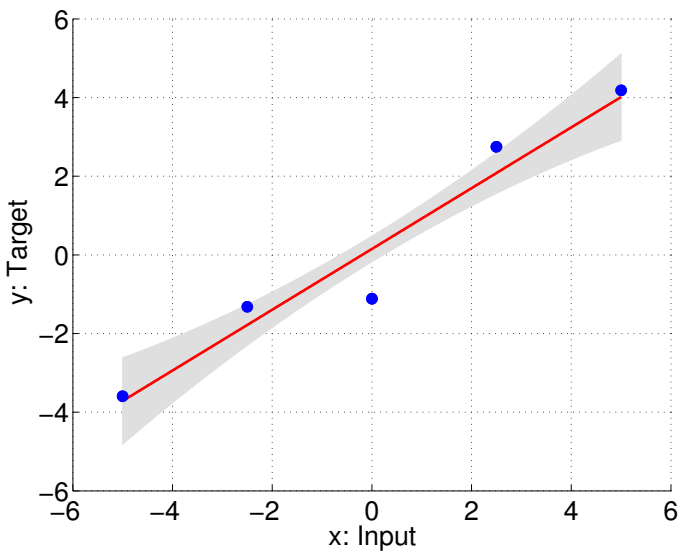


Posterior Weights

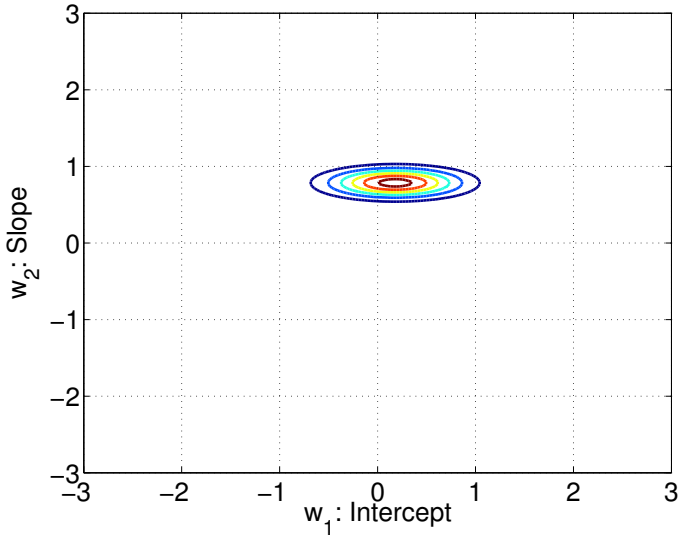
Bayesian Linear Regression Example



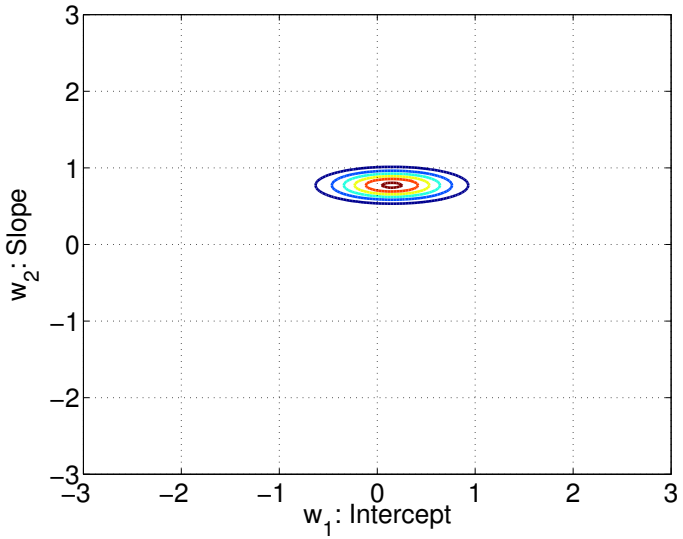
Prior Weights



Predictive Distribution



Likelihood



Posterior Weights

Non-linear Feature Spaces

- Consider the model $f(\mathbf{x}) = \sum_{i=1}^{D'} w_i \phi_i(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$
  - Each $\phi_i(\mathbf{x})$ is a (non-linear) feature on $\mathbf{x}$, e.g. $x_1, x_2, x_1^2, x_2^2, x_1 x_2 \ldots$
  - We have a non-linear mapping but a linear-in-the-parameters model
  - The number of these features can be very large, i.e. $D' \gg D$

Non-linear Feature Spaces

- Consider the model $f(\mathbf{x}) = \sum_{i=1}^{D'} w_i \phi_i(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$
  - Each $\phi_i(\mathbf{x})$ is a (non-linear) feature on $\mathbf{x}$, e.g. $x_1, x_2, x_1^2, x_2^2, x_1 x_2 \ldots$
  - We have a non-linear mapping but a linear-in-the-parameters model
  - The number of these features can be very large, i.e. $D' \gg D$
- All the Bayesian analysis is similar to the standard linear model:

$$p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \mathcal{N}(f_*|\sigma^{-2}\boldsymbol{\phi}_*^T \mathbf{A}^{-1}\boldsymbol{\Phi}\mathbf{y}, \boldsymbol{\phi}_*^T \mathbf{A}^{-1}\boldsymbol{\phi}_*)$$

where: $\boldsymbol{\phi}_* = \boldsymbol{\phi}(\mathbf{x}_*)$, $\boldsymbol{\Phi} = \boldsymbol{\Phi}(\mathbf{X})$, and $\mathbf{A} = (\frac{1}{\sigma^2}\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Sigma}_w^{-1})$

Non-linear Feature Spaces

- Consider the model $f(\mathbf{x}) = \sum_{i=1}^{D'} w_i \phi_i(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$
  - Each $\phi_i(\mathbf{x})$ is a (non-linear) feature on $\mathbf{x}$, e.g. $x_1, x_2, x_1^2, x_2^2, x_1 x_2 \ldots$
  - We have a non-linear mapping but a linear-in-the-parameters model
  - The number of these features can be very large, i.e. $D' \gg D$
- All the Bayesian analysis is similar to the standard linear model:

$$p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \mathcal{N}(f_*|\sigma^{-2}\boldsymbol{\phi}_*^T \mathbf{A}^{-1}\boldsymbol{\Phi}\mathbf{y}, \boldsymbol{\phi}_*^T \mathbf{A}^{-1}\boldsymbol{\phi}_*)$$

where: $\boldsymbol{\phi}_* = \boldsymbol{\phi}(\mathbf{x}_*)$, $\boldsymbol{\Phi} = \boldsymbol{\Phi}(\mathbf{X})$, and $\mathbf{A} = (\frac{1}{\sigma^2}\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Sigma}_w^{-1})$

- Note we need to invert $\mathbf{A}$ of ? dimensions.

Non-linear Feature Spaces

- Consider the model $f(\mathbf{x}) = \sum_{i=1}^{D'} w_i \phi_i(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$
  - Each $\phi_i(\mathbf{x})$ is a (non-linear) feature on $\mathbf{x}$, e.g. $x_1, x_2, x_1^2, x_2^2, x_1 x_2 \ldots$
  - We have a non-linear mapping but a linear-in-the-parameters model
  - The number of these features can be very large, i.e. $D' \gg D$
- All the Bayesian analysis is similar to the standard linear model:

$$p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \mathcal{N}(f_*|\sigma^{-2}\boldsymbol{\phi}_*^T \mathbf{A}^{-1}\boldsymbol{\Phi}\mathbf{y}, \boldsymbol{\phi}_*^T \mathbf{A}^{-1}\boldsymbol{\phi}_*)$$

where: $\boldsymbol{\phi}_* = \boldsymbol{\phi}(\mathbf{x}_*)$, $\boldsymbol{\Phi} = \boldsymbol{\Phi}(\mathbf{X})$, and $\mathbf{A} = (\frac{1}{\sigma^2}\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Sigma}_w^{-1})$

  - Note we need to invert $\mathbf{A}$ of ? dimensions.
- We can rewrite the predictive distribution as:

$$p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \mathcal{N}(f_*|\mathbf{k}_*^T \widetilde{\mathbf{K}}^{-1}\mathbf{y}, k_{**} - \mathbf{k}_*^T \widetilde{\mathbf{K}}^{-1}\mathbf{k}_*)$$

where $\mathbf{k}_* = \boldsymbol{\Phi}^T \boldsymbol{\Sigma}_w \boldsymbol{\phi}_*$, $k_{**} = \boldsymbol{\phi}_*^T \boldsymbol{\Sigma}_w \boldsymbol{\phi}_*$, and $\widetilde{\mathbf{K}} = \boldsymbol{\Phi}^T \boldsymbol{\Sigma}_w \boldsymbol{\Phi} + \sigma^2 \mathbf{I}$

Non-linear Feature Spaces

- Consider the model $f(\mathbf{x}) = \sum_{i=1}^{D'} w_i \phi_i(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$
  - Each $\phi_i(\mathbf{x})$ is a (non-linear) feature on $\mathbf{x}$, e.g. $x_1, x_2, x_1^2, x_2^2, x_1 x_2 \ldots$
  - We have a non-linear mapping but a linear-in-the-parameters model
  - The number of these features can be very large, i.e. $D' \gg D$
- All the Bayesian analysis is similar to the standard linear model:

$$p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \mathcal{N}(f_*|\sigma^{-2}\boldsymbol{\phi}_*^T \mathbf{A}^{-1}\boldsymbol{\Phi}\mathbf{y}, \boldsymbol{\phi}_*^T \mathbf{A}^{-1}\boldsymbol{\phi}_*)$$

where: $\boldsymbol{\phi}_* = \boldsymbol{\phi}(\mathbf{x}_*)$, $\boldsymbol{\Phi} = \boldsymbol{\Phi}(\mathbf{X})$, and $\mathbf{A} = (\frac{1}{\sigma^2}\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Sigma}_w^{-1})$

  - Note we need to invert $\mathbf{A}$ of ? dimensions.
- We can rewrite the predictive distribution as:

$$p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \mathcal{N}(f_*|\mathbf{k}_*^T \widetilde{\mathbf{K}}^{-1}\mathbf{y}, k_{**} - \mathbf{k}_*^T \widetilde{\mathbf{K}}^{-1}\mathbf{k}_*)$$

where $\mathbf{k}_* = \boldsymbol{\Phi}^T \boldsymbol{\Sigma}_w \boldsymbol{\phi}_*$, $k_{**} = \boldsymbol{\phi}_*^T \boldsymbol{\Sigma}_w \boldsymbol{\phi}_*$, and $\widetilde{\mathbf{K}} = \boldsymbol{\Phi}^T \boldsymbol{\Sigma}_w \boldsymbol{\Phi} + \sigma^2 \mathbf{I}$

  - Now we need to invert $\widetilde{\mathbf{K}}$ of ? dimensions ▸ GP prediction

The Kernel Trick

- Note that in:

$$\mathbf{k}_* = \mathbf{\Phi}^\mathsf{T}\mathbf{\Sigma}_w\boldsymbol{\phi}_*,\ \mathrm{k}_{\star\star} = \boldsymbol{\phi}_*^\mathsf{T}\mathbf{\Sigma}_w\boldsymbol{\phi}_* \text{ and } \widetilde{\mathbf{K}} = \mathbf{\Phi}^\mathsf{T}\mathbf{\Sigma}_w\mathbf{\Phi} + \sigma^2\mathbf{I}$$

  the features always enter in the form $\boldsymbol{\phi}(\mathbf{x})^\mathsf{T}\mathbf{\Sigma}_w\boldsymbol{\phi}(\mathbf{x}')$

The Kernel Trick

- Note that in:

$$\mathbf{k}_* = \mathbf{\Phi}^\mathsf{T}\mathbf{\Sigma}_w\boldsymbol{\phi}_*,\ \mathrm{k}_{\star\star} = \boldsymbol{\phi}_*^\mathsf{T}\mathbf{\Sigma}_w\boldsymbol{\phi}_* \text{ and } \widetilde{\mathbf{K}} = \mathbf{\Phi}^\mathsf{T}\mathbf{\Sigma}_w\mathbf{\Phi} + \sigma^2\mathbf{I}$$

  the features always enter in the form $\boldsymbol{\phi}(\mathbf{x})^\mathsf{T}\mathbf{\Sigma}_w\boldsymbol{\phi}(\mathbf{x}')$
- This is an inner product wrt $\mathbf{\Sigma}_w$

The Kernel Trick

- Note that in:

$$\mathbf{k}_* = \mathbf{\Phi}^\mathsf{T}\mathbf{\Sigma}_w\boldsymbol{\phi}_*,\ \mathrm{k}_{\star\star} = \boldsymbol{\phi}_*^\mathsf{T}\mathbf{\Sigma}_w\boldsymbol{\phi}_* \text{ and } \widetilde{\mathbf{K}} = \mathbf{\Phi}^\mathsf{T}\mathbf{\Sigma}_w\mathbf{\Phi} + \sigma^2\mathbf{I}$$

  the features always enter in the form $\boldsymbol{\phi}(\mathbf{x})^\mathsf{T}\mathbf{\Sigma}_w\boldsymbol{\phi}(\mathbf{x}')$
- This is an inner product wrt $\mathbf{\Sigma}_w$
- As $\mathbf{\Sigma}_w$ is PD we can rewrite:

$$\begin{aligned}\boldsymbol{\phi}(\mathbf{x})^\mathsf{T}\mathbf{\Sigma}_w\boldsymbol{\phi}(\mathbf{x}') &= \boldsymbol{\phi}(\mathbf{x})^\mathsf{T}\mathbf{\Sigma}_w^{1/2}\mathbf{\Sigma}_w^{1/2}\boldsymbol{\phi}(\mathbf{x}') \\ &= (\underbrace{\mathbf{\Sigma}_w^{1/2}\boldsymbol{\phi}(\mathbf{x})}_{\boldsymbol{\psi}(\mathbf{x})})^\mathsf{T}(\underbrace{\mathbf{\Sigma}_w^{1/2}\boldsymbol{\phi}(\mathbf{x}')}_{\boldsymbol{\psi}(\mathbf{x}')}) \\ \kappa(\mathbf{x},\mathbf{x}') &= \boldsymbol{\psi}(\mathbf{x})\cdot\boldsymbol{\psi}(\mathbf{x}')\end{aligned}$$

The Kernel Trick

- Note that in:

$$\mathbf{k}_* = \mathbf{\Phi}^\mathsf{T}\mathbf{\Sigma}_w\boldsymbol{\phi}_*,\ \mathrm{k}_{\star\star} = \boldsymbol{\phi}_*^\mathsf{T}\mathbf{\Sigma}_w\boldsymbol{\phi}_* \text{ and } \widetilde{\mathbf{K}} = \mathbf{\Phi}^\mathsf{T}\mathbf{\Sigma}_w\mathbf{\Phi} + \sigma^2\mathbf{I}$$

  the features always enter in the form $\boldsymbol{\phi}(\mathbf{x})^\mathsf{T}\mathbf{\Sigma}_w\boldsymbol{\phi}(\mathbf{x}')$
- This is an inner product wrt $\mathbf{\Sigma}_w$
- As $\mathbf{\Sigma}_w$ is PD we can rewrite:

$$\begin{aligned}\boldsymbol{\phi}(\mathbf{x})^\mathsf{T}\mathbf{\Sigma}_w\boldsymbol{\phi}(\mathbf{x}') &= \boldsymbol{\phi}(\mathbf{x})^\mathsf{T}\mathbf{\Sigma}_w^{1/2}\mathbf{\Sigma}_w^{1/2}\boldsymbol{\phi}(\mathbf{x}') \\ &= (\underbrace{\mathbf{\Sigma}_w^{1/2}\boldsymbol{\phi}(\mathbf{x})}_{\boldsymbol{\psi}(\mathbf{x})})^\mathsf{T}(\underbrace{\mathbf{\Sigma}_w^{1/2}\boldsymbol{\phi}(\mathbf{x}')}_{\boldsymbol{\psi}(\mathbf{x}')}) \\ \kappa(\mathbf{x},\mathbf{x}') &= \boldsymbol{\psi}(\mathbf{x})\cdot\boldsymbol{\psi}(\mathbf{x}')\end{aligned}$$

- $\kappa(\cdot,\cdot)$ is called a kernel or covariance function

The Kernel Trick

- Note that in:

$$\mathbf{k}_* = \mathbf{\Phi}^\mathsf{T}\mathbf{\Sigma}_w\boldsymbol{\phi}_*,\ \mathsf{k}_{\star\star} = \boldsymbol{\phi}_*^\mathsf{T}\mathbf{\Sigma}_w\boldsymbol{\phi}_* \text{ and } \widetilde{\mathbf{K}} = \mathbf{\Phi}^\mathsf{T}\mathbf{\Sigma}_w\mathbf{\Phi} + \sigma^2\mathbf{I}$$

  the features always enter in the form $\boldsymbol{\phi}(\mathbf{x})^\mathsf{T}\mathbf{\Sigma}_w\boldsymbol{\phi}(\mathbf{x}')$
- This is an inner product wrt $\mathbf{\Sigma}_w$
- As $\mathbf{\Sigma}_w$ is PD we can rewrite:

$$\begin{aligned}\boldsymbol{\phi}(\mathbf{x})^\mathsf{T}\mathbf{\Sigma}_w\boldsymbol{\phi}(\mathbf{x}') &= \boldsymbol{\phi}(\mathbf{x})^\mathsf{T}\mathbf{\Sigma}_w^{1/2}\mathbf{\Sigma}_w^{1/2}\boldsymbol{\phi}(\mathbf{x}') \\ &= (\underbrace{\mathbf{\Sigma}_w^{1/2}\boldsymbol{\phi}(\mathbf{x})}_{\boldsymbol{\psi}(\mathbf{x})})^\mathsf{T}(\underbrace{\mathbf{\Sigma}_w^{1/2}\boldsymbol{\phi}(\mathbf{x}')}_{\boldsymbol{\psi}(\mathbf{x}')}) \\ \kappa(\mathbf{x},\mathbf{x}') &= \boldsymbol{\psi}(\mathbf{x})\cdot\boldsymbol{\psi}(\mathbf{x}')\end{aligned}$$

- $\kappa(\cdot,\cdot)$ is called a kernel or covariance function
- We can replace all occurrences of inner products by $\kappa(\cdot,\cdot)$

The Kernel Trick

- Note that in:

$$\mathbf{k}_* = \mathbf{\Phi}^\mathsf{T}\mathbf{\Sigma}_w\boldsymbol{\phi}_*,\ \mathrm{k}_{\star\star} = \boldsymbol{\phi}_*^\mathsf{T}\mathbf{\Sigma}_w\boldsymbol{\phi}_* \text{ and } \widetilde{\mathbf{K}} = \mathbf{\Phi}^\mathsf{T}\mathbf{\Sigma}_w\mathbf{\Phi} + \sigma^2\mathbf{I}$$

  the features always enter in the form $\boldsymbol{\phi}(\mathbf{x})^\mathsf{T}\mathbf{\Sigma}_w\boldsymbol{\phi}(\mathbf{x}')$
- This is an inner product wrt $\mathbf{\Sigma}_w$
- As $\mathbf{\Sigma}_w$ is PD we can rewrite:

$$\begin{aligned}\boldsymbol{\phi}(\mathbf{x})^\mathsf{T}\mathbf{\Sigma}_w\boldsymbol{\phi}(\mathbf{x}') &= \boldsymbol{\phi}(\mathbf{x})^\mathsf{T}\mathbf{\Sigma}_w^{1/2}\mathbf{\Sigma}_w^{1/2}\boldsymbol{\phi}(\mathbf{x}') \\ &= (\underbrace{\mathbf{\Sigma}_w^{1/2}\boldsymbol{\phi}(\mathbf{x})}_{\boldsymbol{\psi}(\mathbf{x})})^\mathsf{T}(\underbrace{\mathbf{\Sigma}_w^{1/2}\boldsymbol{\phi}(\mathbf{x}')}_{\boldsymbol{\psi}(\mathbf{x}')}) \\ \kappa(\mathbf{x},\mathbf{x}') &= \boldsymbol{\psi}(\mathbf{x})\cdot\boldsymbol{\psi}(\mathbf{x}')\end{aligned}$$

- $\kappa(\cdot,\cdot)$ is called a kernel or covariance function
- We can replace all occurrences of inner products by $\kappa(\cdot,\cdot)$
- We do not need to compute the feature vectors explicitly

From a Prior over Weights to a Prior over Functions

- Consider the kinds of functions that can be generated from a set of basis functions with random weights.

From a Prior over Weights to a Prior over Functions

- Consider the kinds of functions that can be generated from a set of basis functions with random weights.
- The $\mathrm{f}(\mathbf{x})$ at a particular point is a random variable:

$$\mathrm{f}(\mathbf{x}) = \mathbf{w}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{x}) \text{ with } \mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{0}, \boldsymbol{\Sigma}_{\mathcal{W}})$$

defined as a linear combination of Gaussian random variables.

From a Prior over Weights to a Prior over Functions

- Consider the kinds of functions that can be generated from a set of basis functions with random weights.
- The $\mathrm{f}(\mathbf{x})$ at a particular point is a random variable:

$$\mathrm{f}(\mathbf{x}) = \mathbf{w}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{x}) \text{ with } \mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{0}, \boldsymbol{\Sigma}_w)$$

  defined as a linear combination of Gaussian random variables.
- A collection of these random variables indexed by $\mathbf{x}$: $\mathrm{f}(\mathbf{x}_1), \ldots, \mathrm{f}(\mathbf{x}_N)$, define a stochastic process in a consistent way.

From a Prior over Weights to a Prior over Functions

- Consider the kinds of functions that can be generated from a set of basis functions with random weights.
- The $\mathrm{f}(\mathbf{x})$ at a particular point is a random variable:

$$\mathrm{f}(\mathbf{x}) = \mathbf{w}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{x}) \text{ with } \mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{0}, \boldsymbol{\Sigma}_{\mathcal{W}})$$

  defined as a linear combination of Gaussian random variables.
- A collection of these random variables indexed by $\mathbf{x}$: $\mathrm{f}(\mathbf{x}_1), \ldots, \mathrm{f}(\mathbf{x}_{\mathrm{N}})$, define a stochastic process in a consistent way.
- The mean and the covariance function for this stochastic process is given by:

$$\mathbb{E}_{\mathcal{W}}[\mathrm{f}(\mathbf{x})] = 0$$

$$\mathbb{E}_{\mathcal{W}}[\mathrm{f}(\mathbf{x})\mathrm{f}(\mathbf{x}')] = \boldsymbol{\phi}^{\mathsf{T}}(\mathbf{x})\boldsymbol{\Sigma}_{\mathcal{W}}\boldsymbol{\phi}(\mathbf{x}')$$

From a Prior over Weights to a Prior over Functions

- Consider the kinds of functions that can be generated from a set of basis functions with random weights.
- The $\mathrm{f}(\mathbf{x})$ at a particular point is a random variable:

$$\mathrm{f}(\mathbf{x}) = \mathbf{w}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{x}) \text{ with } \mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{0}, \boldsymbol{\Sigma}_{\mathcal{W}})$$

  defined as a linear combination of Gaussian random variables.
- A collection of these random variables indexed by $\mathbf{x}$: $\mathrm{f}(\mathbf{x}_1), \ldots, \mathrm{f}(\mathbf{x}_{\mathrm{N}})$, define a stochastic process in a consistent way.
- The mean and the covariance function for this stochastic process is given by:

$$\mathbb{E}_{\mathcal{W}}[\mathrm{f}(\mathbf{x})] = 0$$

$$\mathbb{E}_{\mathcal{W}}[\mathrm{f}(\mathbf{x})\mathrm{f}(\mathbf{x}')] = \boldsymbol{\phi}^{\mathsf{T}}(\mathbf{x})\boldsymbol{\Sigma}_{\mathcal{W}}\boldsymbol{\phi}(\mathbf{x}')$$

- The Bayesian linear model is a Gaussian process

From a Prior over Weights to a Prior over Functions

- Consider the kinds of functions that can be generated from a set of basis functions with random weights.
- The $\mathrm{f}(\mathbf{x})$ at a particular point is a random variable:

$$\mathrm{f}(\mathbf{x}) = \mathbf{w}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{x}) \text{ with } \mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{0}, \boldsymbol{\Sigma}_{\mathcal{W}})$$

  defined as a linear combination of Gaussian random variables.
- A collection of these random variables indexed by $\mathbf{x}$: $\mathrm{f}(\mathbf{x}_1), \ldots, \mathrm{f}(\mathbf{x}_{\mathrm{N}})$, define a stochastic process in a consistent way.
- The mean and the covariance function for this stochastic process is given by:

$$\mathbb{E}_{\mathcal{W}}[\mathrm{f}(\mathbf{x})] = 0$$

$$\mathbb{E}_{\mathcal{W}}[\mathrm{f}(\mathbf{x})\mathrm{f}(\mathbf{x}')] = \boldsymbol{\phi}^{\mathsf{T}}(\mathbf{x})\boldsymbol{\Sigma}_{\mathcal{W}}\boldsymbol{\phi}(\mathbf{x}')$$

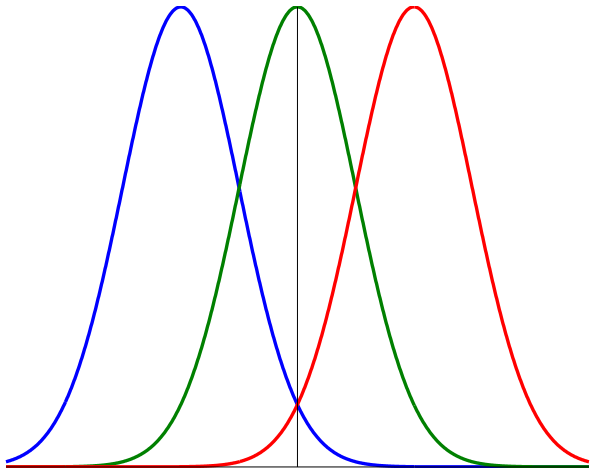
- The Bayesian linear model is a Gaussian process
  - The Function values corresponding to any number of inputs have a joint Gaussian distribution.

Sample Functions from the Linear Model

1. Define $\phi_i(x) = \exp(-\frac{1}{2}(x - \mu_i)^2)$, for $i = 1, 2, 3$
2. Construct $\Phi(i, j) = \phi_i(x_j)$, for $i = 1, 2, 3$
3. Draw $\mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{0}, \mathbf{I})$
4. Draw $\mathbf{f} = \mathbf{\Phi}^T \mathbf{w}$

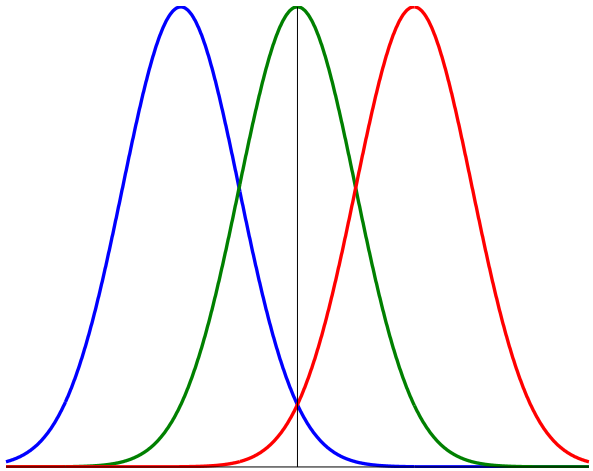
Sample Functions from the Linear Model

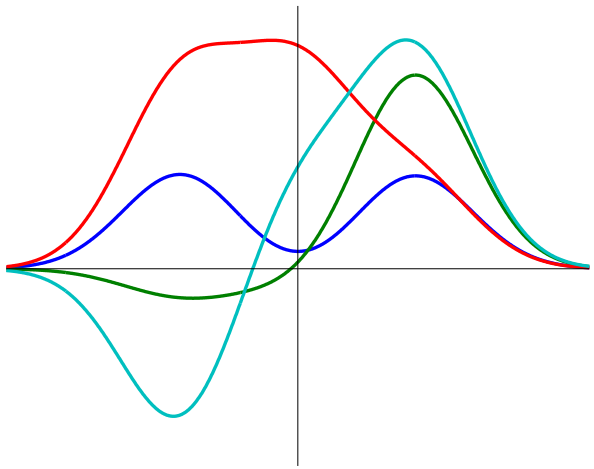
1. Define $\phi_i(x) = \exp(-\frac{1}{2}(x - \mu_i)^2)$, for $i = 1, 2, 3$
2. Construct $\Phi(i, j) = \phi_i(x_j)$, for $i = 1, 2, 3$
3. Draw $\mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{0}, \mathbf{I})$
4. Draw $\mathbf{f} = \mathbf{\Phi}^T \mathbf{w}$



Sample Functions from the Linear Model

1. Define $\phi_i(x) = \exp(-\frac{1}{2}(x - \mu_i)^2)$, for $i = 1, 2, 3$
2. Construct $\Phi(i, j) = \phi_i(x_j)$, for $i = 1, 2, 3$
3. Draw $\mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{0}, \mathbf{I})$
4. Draw $\mathbf{f} = \mathbf{\Phi}^T\mathbf{w}$





Function-space View

Gaussian Process (GP)

$\mathrm{f}(\mathbf{x})$ *is a Gaussian process if for any finite subset of points* $\mathbf{x}_1, \ldots, \mathbf{x}_N$*, the function values* $\mathrm{f}(\mathbf{x}_1), \ldots, \mathrm{f}(\mathbf{x}_N)$ *follow a Gaussian distribution.*

$$\begin{aligned} \mathrm{f}(\mathbf{x}) &\sim \mathcal{GP}(\mu(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}')), \\ \mu(\mathbf{x}) &= \mathbb{E}[\mathrm{f}(\mathbf{x})], \\ \kappa(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(\mathrm{f}(\mathbf{x}) - \mu(\mathbf{x}))(\mathrm{f}(\mathbf{x}') - \mu(\mathbf{x}'))], \end{aligned}$$

$\mu(\mathbf{x})$: mean function, consider $\mu(\mathbf{x}) \equiv \mathbf{0}$

$\kappa(\mathbf{x}, \mathbf{x}')$: parameterized covariance function, notion of similarity

Function-space View

Gaussian Process (GP)

$\mathrm{f}(\mathbf{x})$ *is a Gaussian process if for any finite subset of points* $\mathbf{x}_1, \ldots, \mathbf{x}_N$*, the function values* $\mathrm{f}(\mathbf{x}_1), \ldots, \mathrm{f}(\mathbf{x}_N)$ *follow a Gaussian distribution.*

$$\begin{aligned} \mathrm{f}(\mathbf{x}) &\sim \mathcal{GP}(\mu(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}')), \\ \mu(\mathbf{x}) &= \mathbb{E}[\mathrm{f}(\mathbf{x})], \\ \kappa(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(\mathrm{f}(\mathbf{x}) - \mu(\mathbf{x}))(\mathrm{f}(\mathbf{x}') - \mu(\mathbf{x}'))], \end{aligned}$$

$\mu(\mathbf{x})$: mean function, consider $\mu(\mathbf{x}) \equiv \mathbf{0}$

$\kappa(\mathbf{x}, \mathbf{x}')$: parameterized covariance function, notion of similarity

- Stochastic process: collection of random variables

Function-space View

Gaussian Process (GP)

$\mathrm{f}(\mathbf{x})$ *is a Gaussian process if for any finite subset of points* $\mathbf{x}_1, \ldots, \mathbf{x}_N$*, the function values* $\mathrm{f}(\mathbf{x}_1), \ldots, \mathrm{f}(\mathbf{x}_N)$ *follow a Gaussian distribution.*

$$\begin{aligned} \mathrm{f}(\mathbf{x}) &\sim \mathcal{GP}(\mu(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}')), \\ \mu(\mathbf{x}) &= \mathbb{E}[\mathrm{f}(\mathbf{x})], \\ \kappa(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(\mathrm{f}(\mathbf{x}) - \mu(\mathbf{x}))(\mathrm{f}(\mathbf{x}') - \mu(\mathbf{x}'))], \end{aligned}$$

$\mu(\mathbf{x})$: mean function, consider $\mu(\mathbf{x}) \equiv \mathbf{0}$

$\kappa(\mathbf{x}, \mathbf{x}')$: parameterized covariance function, notion of similarity

- Stochastic process: collection of random variables
- These variables are the values of the function $\mathrm{f}(\mathbf{x})$ *indexed* by the set of all possible input

Function-space View

Gaussian Process (GP)

$\mathrm{f}(\mathbf{x})$ *is a Gaussian process if for any finite subset of points* $\mathbf{x}_1, \ldots, \mathbf{x}_N$*, the function values* $\mathrm{f}(\mathbf{x}_1), \ldots, \mathrm{f}(\mathbf{x}_N)$ *follow a Gaussian distribution.*

$$\begin{aligned} \mathrm{f}(\mathbf{x}) &\sim \mathcal{GP}(\mu(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}')), \\ \mu(\mathbf{x}) &= \mathbb{E}[\mathrm{f}(\mathbf{x})], \\ \kappa(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(\mathrm{f}(\mathbf{x}) - \mu(\mathbf{x}))(\mathrm{f}(\mathbf{x}') - \mu(\mathbf{x}'))], \end{aligned}$$

$\mu(\mathbf{x})$: mean function, consider $\mu(\mathbf{x}) \equiv \mathbf{0}$

$\kappa(\mathbf{x}, \mathbf{x}')$: parameterized covariance function, notion of similarity

- Stochastic process: collection of random variables
- These variables are the values of the function $\mathrm{f}(\mathbf{x})$ *indexed* by the set of all possible input
- Consistency: marginalization property
  $(\mathrm{f}_1, \mathrm{f}_2) \sim \mathcal{N}(\mathbf{f}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \rightarrow \mathrm{f}_1 \sim \mathcal{N}(\mathrm{f}_1|\mu_1, \Sigma_{11})$

The Covariance Function

- It specifies the covariance between pairs of random variables:

$$\mathrm{Cov}(\mathrm{f}(\mathbf{x_p}), \mathrm{f}(\mathbf{x_q})) = \kappa(\mathbf{x_p}, \mathbf{x_q})$$

The Covariance Function

- It specifies the covariance between pairs of random variables:

$$\text{Cov}(\text{f}(\mathbf{x_p}), \text{f}(\mathbf{x_q})) = \kappa(\mathbf{x_p}, \mathbf{x_q})$$

  - Covariance between outputs as a function of the inputs

The Covariance Function

- It specifies the covariance between pairs of random variables:

$$\mathrm{Cov}(\mathrm{f}(\mathbf{x_p}), \mathrm{f}(\mathbf{x_q})) = \kappa(\mathbf{x_p}, \mathbf{x_q})$$

  - Covariance between outputs as a function of the inputs
  - A crucial component in GPs

The Covariance Function

- It specifies the covariance between pairs of random variables:

$$\mathrm{Cov}(\mathrm{f}(\mathbf{x_p}), \mathrm{f}(\mathbf{x_q})) = \kappa(\mathbf{x_p}, \mathbf{x_q})$$

  - Covariance between outputs as a function of the inputs
  - A crucial component in GPs
  - Intuitively, it describes the notion of similarity

The Covariance Function

- It specifies the covariance between pairs of random variables:

$$\text{Cov}(\text{f}(\mathbf{x_p}), \text{f}(\mathbf{x_q})) = \kappa(\mathbf{x_p}, \mathbf{x_q})$$

  - Covariance between outputs as a function of the inputs
  - A crucial component in GPs
  - Intuitively, it describes the notion of similarity
  - It can be parametrized and we can learn its hyperparameters from data

The Covariance Function

- It specifies the covariance between pairs of random variables:

$$\mathrm{Cov}(\mathrm{f}(\mathbf{x_p}), \mathrm{f}(\mathbf{x_q})) = \kappa(\mathbf{x_p}, \mathbf{x_q})$$

  - Covariance between outputs as a function of the inputs
  - A crucial component in GPs
  - Intuitively, it describes the notion of similarity
  - It can be parametrized and we can learn its hyperparameters from data
- The matrix $\mathbf{K}$ such that $\mathrm{K}_{i,j} = \kappa(x_i, x_j)$ all pairwise input points is known as the covariance matrix or Gram matrix.

The Covariance Function

- It specifies the covariance between pairs of random variables:

$$\mathrm{Cov}(\mathrm{f}(\mathbf{x_p}), \mathrm{f}(\mathbf{x_q})) = \kappa(\mathbf{x_p}, \mathbf{x_q})$$

  - Covariance between outputs as a function of the inputs
  - A crucial component in GPs
  - Intuitively, it describes the notion of similarity
  - It can be parametrized and we can learn its hyperparameters from data
- The matrix $\mathbf{K}$ such that $\mathrm{K}_{i,j} = \kappa(x_i, x_j)$ all pairwise input points is known as the covariance matrix or Gram matrix.
- it must generate a positive semidefinite (PSD) matrix at any subset of points, i.e. $\mathbf{b}^{\mathsf{T}}\mathbf{K}\mathbf{b} \geqslant 0$, $\forall \mathbf{b} \in \mathbb{R}^{\mathrm{N}}$

The Covariance Function

- It specifies the covariance between pairs of random variables:

$$\mathrm{Cov}(\mathrm{f}(\mathbf{x_p}), \mathrm{f}(\mathbf{x_q})) = \kappa(\mathbf{x_p}, \mathbf{x_q})$$

  - Covariance between outputs as a function of the inputs
  - A crucial component in GPs
  - Intuitively, it describes the notion of similarity
  - It can be parametrized and we can learn its hyperparameters from data
- The matrix $\mathbf{K}$ such that $\mathrm{K}_{i,j} = \kappa(x_i, x_j)$ all pairwise input points is known as the covariance matrix or Gram matrix.
- it must generate a positive semidefinite (PSD) matrix at any subset of points, i.e. $\mathbf{b}^{\mathsf{T}}\mathbf{K}\mathbf{b} \geqslant 0$, $\forall \mathbf{b} \in \mathbb{R}^{\mathrm{N}}$
- Stationary: $\varphi(\mathbf{x} - \mathbf{x}')$ - translation invariant

The Covariance Function

- It specifies the covariance between pairs of random variables:

$$\mathrm{Cov}(\mathrm{f}(\mathbf{x_p}), \mathrm{f}(\mathbf{x_q})) = \kappa(\mathbf{x_p}, \mathbf{x_q})$$

  - Covariance between outputs as a function of the inputs
  - A crucial component in GPs
  - Intuitively, it describes the notion of similarity
  - It can be parametrized and we can learn its hyperparameters from data
- The matrix $\mathbf{K}$ such that $\mathrm{K}_{i,j} = \kappa(x_i, x_j)$ all pairwise input points is known as the covariance matrix or Gram matrix.
- it must generate a positive semidefinite (PSD) matrix at any subset of points, i.e. $\mathbf{b}^{\mathsf{T}}\mathbf{K}\mathbf{b} \geqslant 0$, $\forall \mathbf{b} \in \mathbb{R}^{\mathrm{N}}$
- Stationary: $\varphi(\mathbf{x} - \mathbf{x}')$ - translation invariant
- Isotropic: $\varphi(\|\mathbf{x} - \mathbf{x}'\|)$

The Squared Exponential (SE) Covariance Function

$$\kappa(\mathbf{x}, \mathbf{x}') = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^\mathsf{T} \mathbf{C} (\mathbf{x} - \mathbf{x}')\right)$$

- σ_s^2 is the signal variance

The Squared Exponential (SE) Covariance Function

$$\kappa(\mathbf{x}, \mathbf{x}') = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^\mathsf{T} \mathbf{C} (\mathbf{x} - \mathbf{x}')\right)$$

- σ_s^2 is the signal variance
- $\mathbf{C}$ is a symmetric matrix that can have different parameterizations

The Squared Exponential (SE) Covariance Function

$$\kappa(\mathbf{x}, \mathbf{x}') = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^\mathsf{T}\mathbf{C}(\mathbf{x} - \mathbf{x}')\right)$$

- σ_s^2 is the signal variance
- $\mathbf{C}$ is a symmetric matrix that can have different parameterizations
- $\mathbf{C} = \ell^{-2}\mathbf{I}$: isotropic SE

The Squared Exponential (SE) Covariance Function

$$\kappa(\mathbf{x}, \mathbf{x}') = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^\mathsf{T}\mathbf{C}(\mathbf{x} - \mathbf{x}')\right)$$

- σ_s^2 is the signal variance
- $\mathbf{C}$ is a symmetric matrix that can have different parameterizations
- $\mathbf{C} = \ell^{-2}\mathbf{I}$: isotropic SE
- $\mathbf{C} = \text{diag}(\boldsymbol{\ell})^{-2}$ with $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_\text{D})$: Automatic Relevance Determination (ARD)

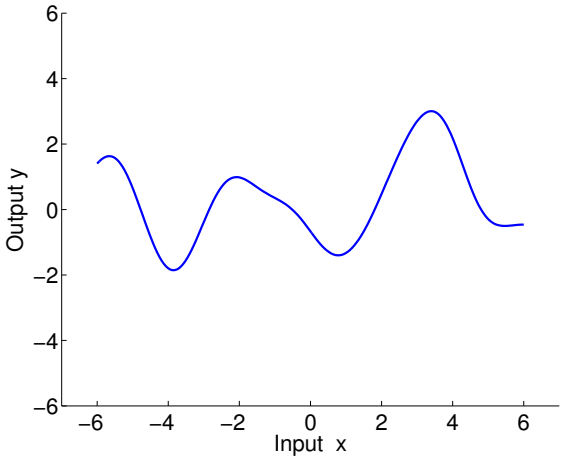
The Squared Exponential (SE) Covariance Function

$$\kappa(\mathbf{x}, \mathbf{x}') = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{x}')^{\mathsf{T}}\mathbf{C}(\mathbf{x}-\mathbf{x}')\right)$$

- σ_s^2 is the signal variance
- $\mathbf{C}$ is a symmetric matrix that can have different parameterizations
- $\mathbf{C} = \ell^{-2}\mathbf{I}$: isotropic SE
- $\mathbf{C} = \text{diag}(\boldsymbol{\ell})^{-2}$ with $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_D)$: Automatic Relevance Determination (ARD)
- Each ℓ_j is known as the characteristic length-scale: distance for which the function values are expected to vary significantly

The Squared Exponential (SE) Covariance Function

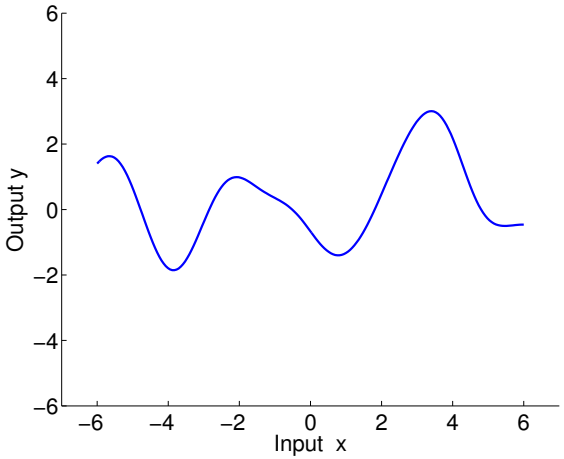
Example



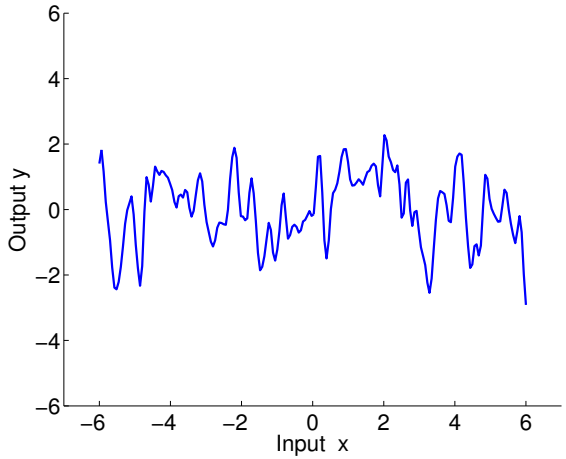
$$\ell = 1,\ \sigma_s^2 = 1$$

The Squared Exponential (SE) Covariance Function

Example



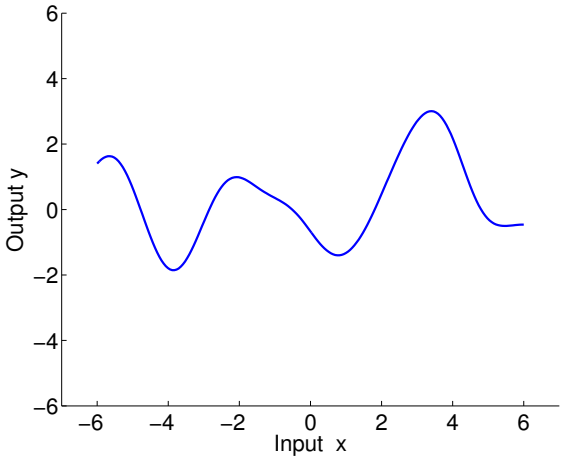
$$\ell = 1,\ \sigma_s^2 = 1$$



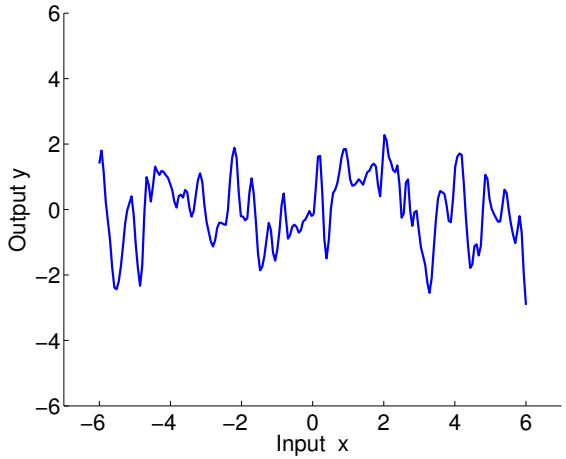
$$\ell = 0.1,\ \sigma_s^2 = 1$$

The Squared Exponential (SE) Covariance Function

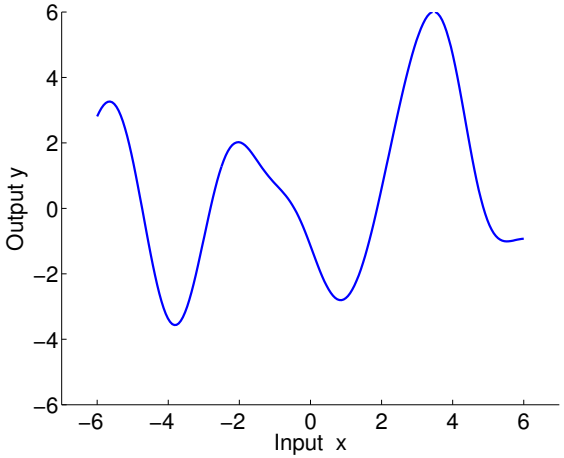
Example



$\ell = 1,\ \sigma_s^2 = 1$



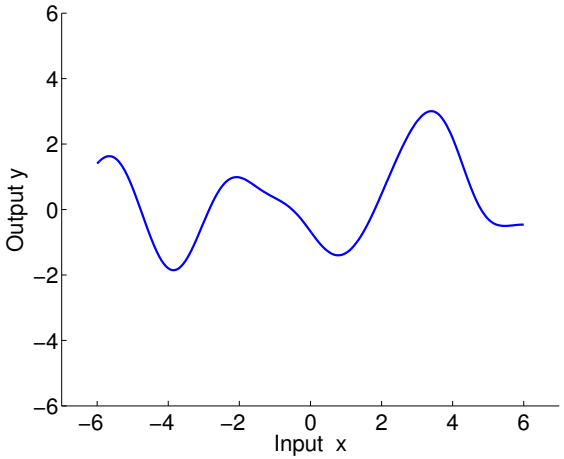
$\ell = 0.1,\ \sigma_s^2 = 1$



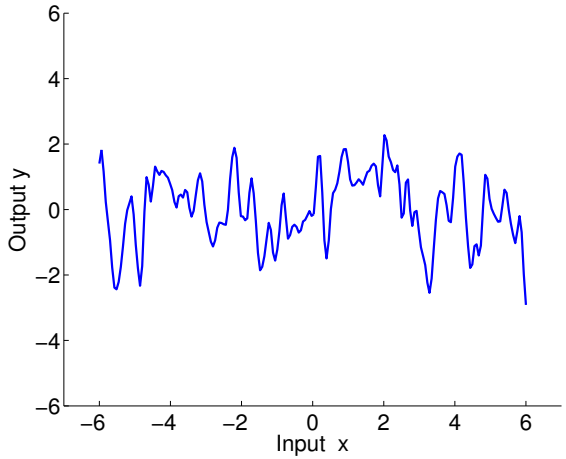
$\ell = 1,\ \sigma_s^2 = 4$

The Squared Exponential (SE) Covariance Function

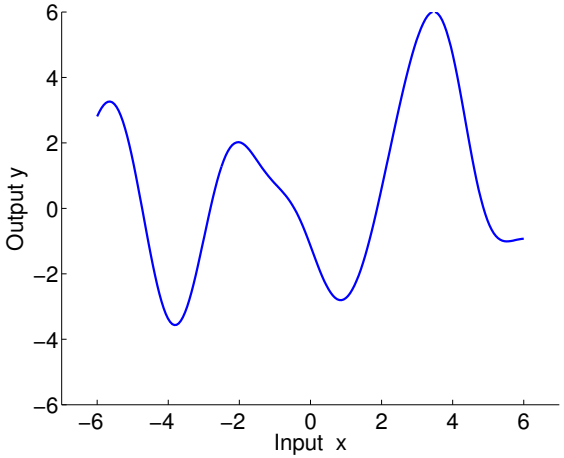
Example



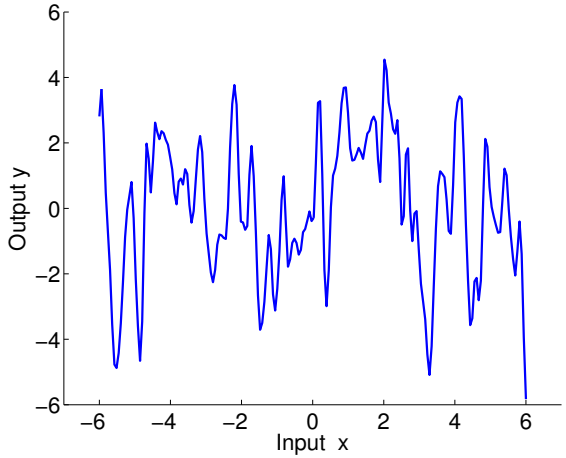
$\ell = 1,\ \sigma_s^2 = 1$



$\ell = 0.1,\ \sigma_s^2 = 1$



$\ell = 1,\ \sigma_s^2 = 4$



$\ell = 0.1,\ \sigma_s^2 = 4$

Standard GP Regression Model: Predictions (1)

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

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

Goal : Make predictions $f_* = f(\mathbf{x}_*)$ at $\mathbf{x}_*$

Standard GP Regression Model: Predictions (1)

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

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

Goal : Make predictions $f_* = f(\mathbf{x}_*)$ at $\mathbf{x}_*$

Prior $f(\mathbf{x}) \sim \mathcal{GP}(\mathbf{0}, \kappa(\mathbf{x}, \mathbf{x}'))$

Standard GP Regression Model: Predictions (1)

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

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

Goal : Make predictions $f_* = f(\mathbf{x}_*)$ at $\mathbf{x}_*$

$$\text{Prior} \quad f(\mathbf{x}) \sim \mathcal{GP}(\mathbf{0}, \kappa(\mathbf{x}, \mathbf{x}'))$$

$$\text{Noise} \quad y = f(\mathbf{x}) + \eta \qquad \eta \sim \mathcal{N}(0, \sigma_n^2)$$

Standard GP Regression Model: Predictions (1)

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

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

$$\text{Goal} : \text{Make predictions } f_* = f(\mathbf{x}_*) \text{ at } \mathbf{x}_*$$

$$\begin{aligned} \text{Prior} \quad f(\mathbf{x}) &\sim \mathcal{GP}(\mathbf{0}, \kappa(\mathbf{x}, \mathbf{x}')) \\ \text{Noise} \quad y &= f(\mathbf{x}) + \eta \qquad \eta \sim \mathcal{N}(0, \sigma_n^2) \end{aligned}$$

- The joint distribution of $\mathbf{y}$ and f_* is a Gaussian

Standard GP Regression Model: Predictions (1)

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

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

$$\text{Goal} : \text{Make predictions } f_* = f(\mathbf{x}_*) \text{ at } \mathbf{x}_*$$

$$\begin{aligned} \text{Prior} \quad f(\mathbf{x}) &\sim \mathcal{GP}(\mathbf{0}, \kappa(\mathbf{x}, \mathbf{x}')) \\ \text{Noise} \quad y &= f(\mathbf{x}) + \eta \qquad \eta \sim \mathcal{N}(0, \sigma_n^2) \end{aligned}$$

- The joint distribution of $\mathbf{y}$ and f_* is a Gaussian
- We simply need to figure out the covariance structure:
  $\text{Cov}(y_p, y_q) = \kappa(\mathbf{x}_p, \mathbf{x}_q) + \sigma_n^2 \delta_{pq} \rightarrow \text{Cov}(\mathbf{y}) = \mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}$

Standard GP Regression Model: Predictions (1)

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

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

$$\text{Goal} : \text{Make predictions } f_* = f(\mathbf{x}_*) \text{ at } \mathbf{x}_*$$

$$\begin{aligned} \text{Prior} \quad f(\mathbf{x}) &\sim \mathcal{GP}(\mathbf{0}, \kappa(\mathbf{x}, \mathbf{x}')) \\ \text{Noise} \quad y &= f(\mathbf{x}) + \eta \qquad \eta \sim \mathcal{N}(0, \sigma_n^2) \end{aligned}$$

- The joint distribution of $\mathbf{y}$ and f_* is a Gaussian
- We simply need to figure out the covariance structure:
  $\text{Cov}(y_p, y_q) = \kappa(\mathbf{x}_p, \mathbf{x}_q) + \sigma_n^2 \delta_{pq} \rightarrow \text{Cov}(\mathbf{y}) = \mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}$
- To get the posterior on f_* we need to constrain this distribution to agree with the observed data $(\mathbf{X}, \mathbf{y})$

Standard GP Regression Model: Predictions (1)

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

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

$$\text{Goal} : \text{Make predictions } f_* = f(\mathbf{x}_*) \text{ at } \mathbf{x}_*$$

$$\begin{aligned} \text{Prior} \quad f(\mathbf{x}) &\sim \mathcal{GP}(\mathbf{0}, \kappa(\mathbf{x}, \mathbf{x}')) \\ \text{Noise} \quad y &= f(\mathbf{x}) + \eta \qquad \eta \sim \mathcal{N}(0, \sigma_n^2) \end{aligned}$$

- The joint distribution of $\mathbf{y}$ and f_* is a Gaussian
- We simply need to figure out the covariance structure:
  $\text{Cov}(y_p, y_q) = \kappa(\mathbf{x}_p, \mathbf{x}_q) + \sigma_n^2 \delta_{pq} \rightarrow \text{Cov}(\mathbf{y}) = \mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}$
- To get the posterior on f_* we need to constrain this distribution to agree with the observed data $(\mathbf{X}, \mathbf{y})$
- This is achieved simply by conditioning: $p(f_* | \mathbf{X}, \mathbf{y}, \mathbf{x}_*)$

Standard GP Regression Model: Predictions (2)

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{matrix} \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I} & \mathbf{k}(\mathbf{X},\mathbf{x}_*) \\ \mathbf{k}(\mathbf{x}_*,\mathbf{X}) & \kappa(\mathbf{x}_*\mathbf{x}_*) \end{matrix}\right)$$

Standard GP Regression Model: Predictions (2)

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{matrix} \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I} & \mathbf{k}(\mathbf{X},\mathbf{x}_*) \\ \mathbf{k}(\mathbf{x}_*,\mathbf{X}) & \kappa(\mathbf{x}_*\mathbf{x}_*) \end{matrix}\right)$$

Denoting $\mathbf{k}_* = \mathbf{K}(\mathbf{X},\mathbf{x}_*)$ and $\widetilde{\mathbf{K}} = \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I}$

Standard GP Regression Model: Predictions (2)

$$\begin{bmatrix} \mathbf{y} \\ \mathrm{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{matrix} \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I} & \mathbf{k}(\mathbf{X},\mathbf{x}_*) \\ \mathbf{k}(\mathbf{x}_*,\mathbf{X}) & \kappa(\mathbf{x}_*\mathbf{x}_*) \end{matrix}\right)$$

Denoting $\mathbf{k}_* = \mathbf{K}(\mathbf{X},\mathbf{x}_*)$ and $\widetilde{\mathbf{K}} = \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I}$ then:

$$\begin{aligned} \mathrm{f}_*|\mathbf{X},\mathbf{y},\mathbf{x}_* &\sim \mathcal{N}(\mathbb{E}[\mathrm{f}_*],\mathbb{V}[\mathrm{f}_*]), \\ \mathbb{E}[\mathrm{f}_*] &= \mathbf{k}_*^\mathsf{T}\widetilde{\mathbf{K}}^{-1}\mathbf{y}, \\ \mathbb{V}[\mathrm{f}_*] &= \kappa(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}_*^\mathsf{T}\widetilde{\mathbf{K}}^{-1}\mathbf{k}_*. \end{aligned}$$

Standard GP Regression Model: Predictions (2)

$$\begin{bmatrix} \mathbf{y} \\ \mathrm{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{matrix} \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I} & \mathbf{k}(\mathbf{X},\mathbf{x}_*) \\ \mathbf{k}(\mathbf{x}_*,\mathbf{X}) & \kappa(\mathbf{x}_*\mathbf{x}_*) \end{matrix}\right)$$

Denoting $\mathbf{k}_* = \mathbf{K}(\mathbf{X},\mathbf{x}_*)$ and $\widetilde{\mathbf{K}} = \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I}$ then:

$$\begin{aligned} \mathrm{f}_*|\mathbf{X},\mathbf{y},\mathbf{x}_* &\sim \mathcal{N}(\mathbb{E}[\mathrm{f}_*],\mathbb{V}[\mathrm{f}_*]), \\ \mathbb{E}[\mathrm{f}_*] &= \mathbf{k}_*^\top\widetilde{\mathbf{K}}^{-1}\mathbf{y}, \\ \mathbb{V}[\mathrm{f}_*] &= \kappa(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}_*^\top\widetilde{\mathbf{K}}^{-1}\mathbf{k}_*. \end{aligned}$$

- $\mathbb{E}[\mathrm{f}_*]$: Linear combination of N observations, i.e. linear predictor

Standard GP Regression Model: Predictions (2)

$$\begin{bmatrix} \mathbf{y} \\ \mathrm{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{matrix} \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I} & \mathbf{k}(\mathbf{X},\mathbf{x}_*) \\ \mathbf{k}(\mathbf{x}_*,\mathbf{X}) & \kappa(\mathbf{x}_*\mathbf{x}_*) \end{matrix}\right)$$

Denoting $\mathbf{k}_* = \mathbf{K}(\mathbf{X},\mathbf{x}_*)$ and $\widetilde{\mathbf{K}} = \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I}$ then:

$$\begin{aligned} \mathrm{f}_*|\mathbf{X},\mathbf{y},\mathbf{x}_* &\sim \mathcal{N}(\mathbb{E}[\mathrm{f}_*],\mathbb{V}[\mathrm{f}_*]), \\ \mathbb{E}[\mathrm{f}_*] &= \mathbf{k}_*^\top \widetilde{\mathbf{K}}^{-1}\mathbf{y}, \\ \mathbb{V}[\mathrm{f}_*] &= \kappa(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}_*^\top \widetilde{\mathbf{K}}^{-1}\mathbf{k}_*. \end{aligned}$$

- $\mathbb{E}[\mathrm{f}_*]$: Linear combination of N observations, i.e. linear predictor
- Say $\boldsymbol{\alpha} = (\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y}$ then: $\mathbb{E}[\mathrm{f}_*] = \sum_{i=1}^{N} \alpha_i \kappa(\mathbf{x}_i,\mathbf{x}_*)$ is a linear combination of N kernel functions: Representer theorem

Standard GP Regression Model: Predictions (2)

$$\begin{bmatrix} \mathbf{y} \\ \mathrm{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{matrix} \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I} & \mathbf{k}(\mathbf{X},\mathbf{x}_*) \\ \mathbf{k}(\mathbf{x}_*,\mathbf{X}) & \kappa(\mathbf{x}_*\mathbf{x}_*) \end{matrix}\right)$$

Denoting $\mathbf{k}_* = \mathbf{K}(\mathbf{X},\mathbf{x}_*)$ and $\widetilde{\mathbf{K}} = \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I}$ then:

$$\begin{aligned} \mathrm{f}_*|\mathbf{X},\mathbf{y},\mathbf{x}_* &\sim \mathcal{N}(\mathbb{E}[\mathrm{f}_*],\mathbb{V}[\mathrm{f}_*]), \\ \mathbb{E}[\mathrm{f}_*] &= \mathbf{k}_*^\top \widetilde{\mathbf{K}}^{-1}\mathbf{y}, \\ \mathbb{V}[\mathrm{f}_*] &= \kappa(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}_*^\top \widetilde{\mathbf{K}}^{-1}\mathbf{k}_*. \end{aligned}$$

- $\mathbb{E}[\mathrm{f}_*]$: Linear combination of N observations, i.e. linear predictor
- Say $\boldsymbol{\alpha} = (\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y}$ then: $\mathbb{E}[\mathrm{f}_*] = \sum_{i=1}^{N} \alpha_i \kappa(\mathbf{x}_i,\mathbf{x}_*)$ is a linear combination of N kernel functions: Representer theorem
- We encountered this predictive distribution before ▸ Go to Linear Model

Standard GP Regression Model: Predictions (2)

$$\begin{bmatrix} \mathbf{y} \\ \mathrm{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{matrix} \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I} & \mathbf{k}(\mathbf{X},\mathbf{x}_*) \\ \mathbf{k}(\mathbf{x}_*,\mathbf{X}) & \kappa(\mathbf{x}_*\mathbf{x}_*) \end{matrix}\right)$$

Denoting $\mathbf{k}_* = \mathbf{K}(\mathbf{X},\mathbf{x}_*)$ and $\widetilde{\mathbf{K}} = \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I}$ then:

$$\begin{aligned} \mathrm{f}_*|\mathbf{X},\mathbf{y},\mathbf{x}_* &\sim \mathcal{N}(\mathbb{E}[\mathrm{f}_*],\mathbb{V}[\mathrm{f}_*]), \\ \mathbb{E}[\mathrm{f}_*] &= \mathbf{k}_*^\top \widetilde{\mathbf{K}}^{-1}\mathbf{y}, \\ \mathbb{V}[\mathrm{f}_*] &= \kappa(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}_*^\top \widetilde{\mathbf{K}}^{-1}\mathbf{k}_*. \end{aligned}$$

- $\mathbb{E}[\mathrm{f}_*]$: Linear combination of N observations, i.e. linear predictor
- Say $\boldsymbol{\alpha} = (\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y}$ then: $\mathbb{E}[\mathrm{f}_*] = \sum_{i=1}^{N} \alpha_i \kappa(\mathbf{x}_i,\mathbf{x}_*)$ is a linear combination of N kernel functions: Representer theorem
- We encountered this predictive distribution before ▸ Go to Linear Model
- $\mathbb{V}[\mathrm{f}_*]$ does not depend on $\mathbf{y}$

Standard GP Regression Model: Predictions (2)

$$\begin{bmatrix} \mathbf{y} \\ \mathrm{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{matrix} \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I} & \mathbf{k}(\mathbf{X},\mathbf{x}_*) \\ \mathbf{k}(\mathbf{x}_*,\mathbf{X}) & \kappa(\mathbf{x}_*\mathbf{x}_*) \end{matrix}\right)$$

Denoting $\mathbf{k}_* = \mathbf{K}(\mathbf{X},\mathbf{x}_*)$ and $\widetilde{\mathbf{K}} = \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I}$ then:

$$\begin{aligned} \mathrm{f}_*|\mathbf{X},\mathbf{y},\mathbf{x}_* &\sim \mathcal{N}(\mathbb{E}[\mathrm{f}_*],\mathbb{V}[\mathrm{f}_*]), \\ \mathbb{E}[\mathrm{f}_*] &= \mathbf{k}_*^\top \widetilde{\mathbf{K}}^{-1}\mathbf{y}, \\ \mathbb{V}[\mathrm{f}_*] &= \kappa(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}_*^\top \widetilde{\mathbf{K}}^{-1}\mathbf{k}_*. \end{aligned}$$

- $\mathbb{E}[\mathrm{f}_*]$: Linear combination of N observations, i.e. linear predictor
- Say $\boldsymbol{\alpha} = (\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y}$ then: $\mathbb{E}[\mathrm{f}_*] = \sum_{i=1}^{N} \alpha_i \kappa(\mathbf{x}_i,\mathbf{x}_*)$ is a linear combination of N kernel functions: Representer theorem
- We encountered this predictive distribution before ▸ Go to Linear Model
- $\mathbb{V}[\mathrm{f}_*]$ does not depend on $\mathbf{y}$
- In fact we have a Gaussian posterior process

The Graphical Model for GPs

The Graphical Model for GPs

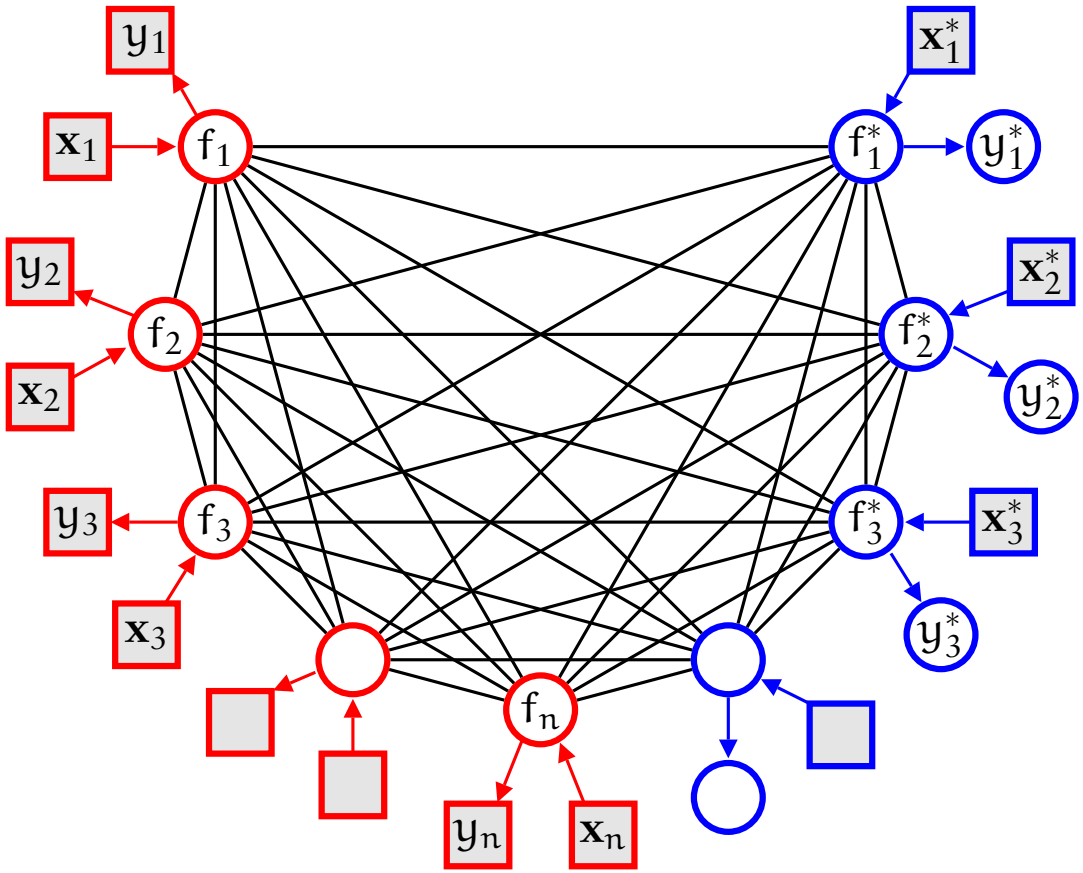


Figure from Carl Rasmussen's slides

The Graphical Model for GPs

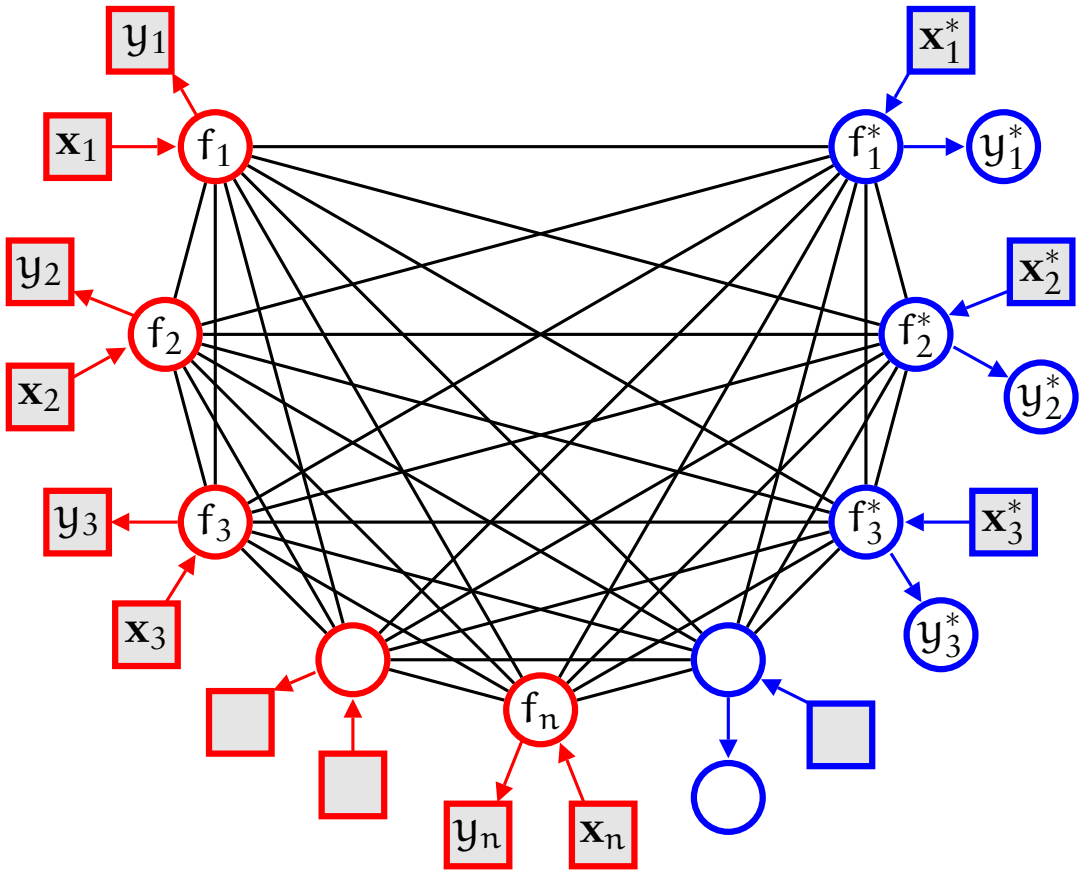


Figure from Carl Rasmussen's slides

- Observations y depend on their corresponding latent function f

The Graphical Model for GPs

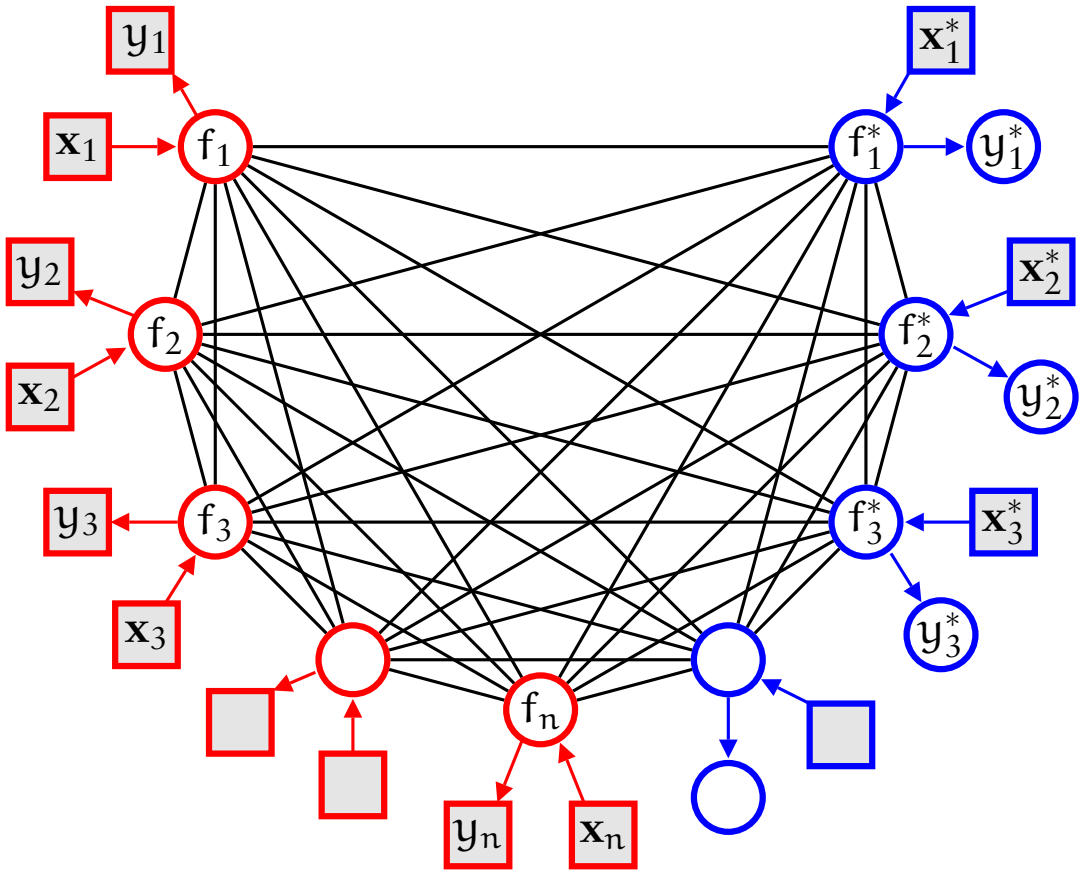


Figure from Carl Rasmussen's slides

- Observations y depend on their corresponding latent function f
- The marginalization property implies that adding a new x_i^*, f_i^*, y_i^* does not affect the distribution

Model Selection

- It includes the discrete choice of the functional form for the covariance function and the values for the hyper-parameters.

Model Selection

- It includes the discrete choice of the functional form for the covariance function and the values for the hyper-parameters.
- E.g. for the SE: $\kappa(\mathbf{x}, \mathbf{x}') = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^\mathsf{T} \mathbf{C} (\mathbf{x} - \mathbf{x}')^\mathsf{T}\right)$ the parameters are σ_s^2 and the parameters of $\mathbf{C}$

Model Selection

- It includes the discrete choice of the functional form for the covariance function and the values for the hyper-parameters.
- E.g. for the SE: $\kappa(\mathbf{x}, \mathbf{x}') = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^\mathsf{T}\mathbf{C}(\mathbf{x} - \mathbf{x}')^\mathsf{T}\right)$ the parameters are σ_s^2 and the parameters of $\mathbf{C}$
- However, we will refer to the set of hyper-parameters $\boldsymbol{\theta}$ as the parameters of the covariance and the noise variance σ_n^2

Model Selection

- It includes the discrete choice of the functional form for the covariance function and the values for the hyper-parameters.
- E.g. for the SE: $\kappa(\mathbf{x}, \mathbf{x}') = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^\mathsf{T}\mathbf{C}(\mathbf{x} - \mathbf{x}')^\mathsf{T}\right)$ the parameters are σ_s^2 and the parameters of $\mathbf{C}$
- However, we will refer to the set of hyper-parameters $\boldsymbol{\theta}$ as the parameters of the covariance and the noise variance σ_n^2
- We can do cross-validation (potential problems?)

Model Selection

- It includes the discrete choice of the functional form for the covariance function and the values for the hyper-parameters.
- E.g. for the SE: $\kappa(\mathbf{x}, \mathbf{x}') = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^\mathsf{T}\mathbf{C}(\mathbf{x} - \mathbf{x}')^\mathsf{T}\right)$ the parameters are σ_s^2 and the parameters of $\mathbf{C}$
- However, we will refer to the set of hyper-parameters $\boldsymbol{\theta}$ as the parameters of the covariance and the noise variance σ_n^2
- We can do cross-validation (potential problems?)
- We focus here on the so-called type II maximum likelihood, i.e. we want to maximize the marginal likelihood.

Model Selection

- It includes the discrete choice of the functional form for the covariance function and the values for the hyper-parameters.
- E.g. for the SE: $\kappa(\mathbf{x}, \mathbf{x}') = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^\mathsf{T}\mathbf{C}(\mathbf{x} - \mathbf{x}')^\mathsf{T}\right)$ the parameters are σ_s^2 and the parameters of $\mathbf{C}$
- However, we will refer to the set of hyper-parameters $\boldsymbol{\theta}$ as the parameters of the covariance and the noise variance σ_n^2
- We can do cross-validation (potential problems?)
- We focus here on the so-called type II maximum likelihood, i.e. we want to maximize the marginal likelihood.
- Integrate out the "parameters" of the GP: (which parameters?)

Model Selection

- It includes the discrete choice of the functional form for the covariance function and the values for the hyper-parameters.
- E.g. for the SE: $\kappa(\mathbf{x}, \mathbf{x}') = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^\mathsf{T}\mathbf{C}(\mathbf{x} - \mathbf{x}')^\mathsf{T}\right)$ the parameters are σ_s^2 and the parameters of $\mathbf{C}$
- However, we will refer to the set of hyper-parameters θ as the parameters of the covariance and the noise variance σ_n^2
- We can do cross-validation (potential problems?)
- We focus here on the so-called type II maximum likelihood, i.e. we want to maximize the marginal likelihood.
- Integrate out the "parameters" of the GP: (which parameters?)

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}, \theta) &= \int p(\mathbf{y}|\mathbf{f}, \mathbf{X}, \theta) p(\mathbf{f}|\mathbf{X}, \theta) d\mathbf{f} \\ &= \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K} + \sigma^2 \mathbf{I}) \end{aligned}$$

Log Marginal Likelihood

$$\mathcal{L} = \log p(\mathbf{y}|\mathbf{X}, \theta)$$

$$= \underbrace{-\frac{1}{2}\mathbf{y}^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1}\mathbf{y}}_{\text{data-fit}} - \underbrace{\frac{1}{2}\log|\mathbf{K} + \sigma_n^2 \mathbf{I}|}_{\text{complexity}} - \underbrace{\frac{N}{2}\log 2\pi}_{normaliz.}$$

Log Marginal Likelihood

$$\mathcal{L} = \log p(\mathbf{y}|\mathbf{X}, \theta)$$

$$= \underbrace{-\frac{1}{2}\mathbf{y}^\top(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y}}_{\text{data-fit}} - \underbrace{\frac{1}{2}\log|\mathbf{K} + \sigma_n^2\mathbf{I}|}_{\text{complexity}} - \underbrace{\frac{N}{2}\log 2\pi}_{normaliz.}$$

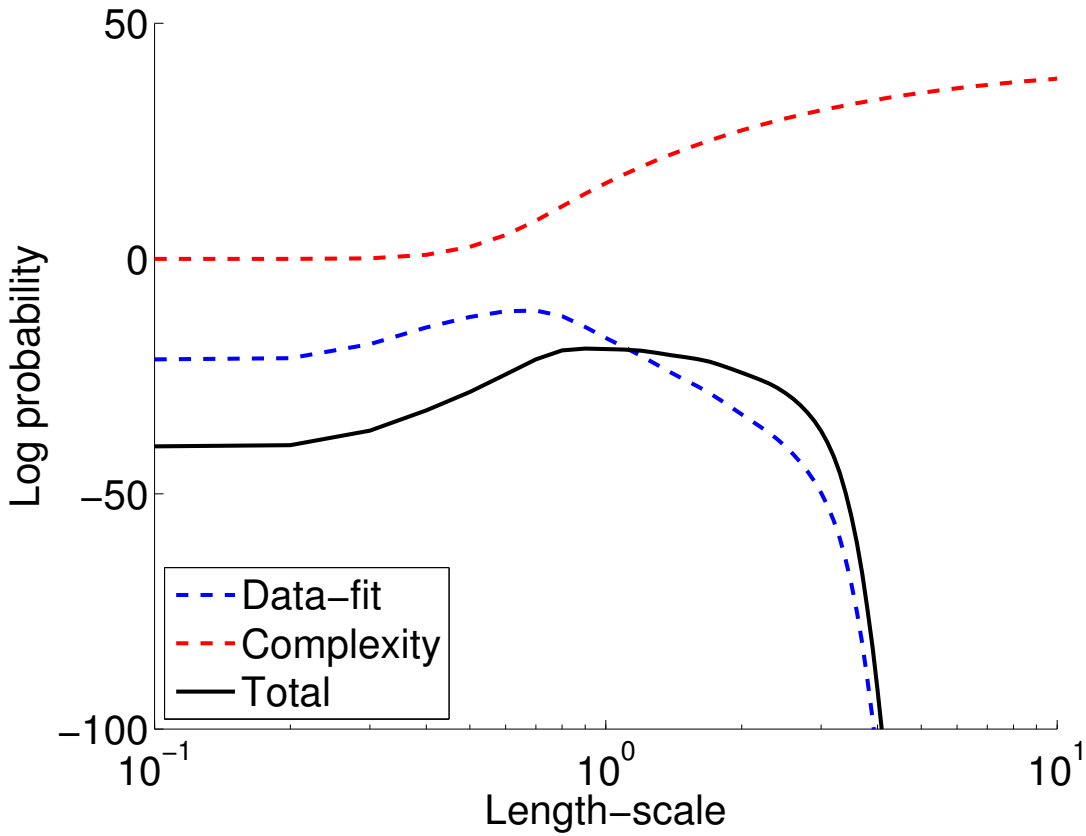
- Isotropic SE
- $\sigma_s^2 = 1,\ \sigma_n^2 = 0.01$
- $\ell = 1$
- $N = 20$

Log Marginal Likelihood

$$\mathcal{L} = \log p(\mathbf{y}|\mathbf{X}, \theta)$$

$$= \underbrace{-\frac{1}{2}\mathbf{y}^\top(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y}}_{\text{data-fit}} - \underbrace{\frac{1}{2}\log|\mathbf{K} + \sigma_n^2\mathbf{I}|}_{\text{complexity}} - \underbrace{\frac{N}{2}\log 2\pi}_{normaliz.}$$

- Isotropic SE
- $\sigma_s^2 = 1,\ \sigma_n^2 = 0.01$
- $\ell = 1$
- $N = 20$



Hyper-parameter Learning

Let $\widetilde{\mathbf{K}} = \mathbf{K} + \sigma_n^2 \mathbf{I}$:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{1}{2}\mathbf{y}^\top \widetilde{\mathbf{K}}^{-1} \frac{\partial \widetilde{\mathbf{K}}}{\partial \theta_i} \widetilde{\mathbf{K}}^{-1}\mathbf{y} - \frac{1}{2}\ \mathrm{tr}\ \left(\widetilde{\mathbf{K}}^{-1} \frac{\partial \widetilde{\mathbf{K}}}{\partial \theta_i} \right)$$

$$= \frac{1}{2}\ \mathrm{tr}\ \left((\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \widetilde{\mathbf{K}}^{-1}) \frac{\partial \widetilde{\mathbf{K}}}{\partial \theta_i} \right)$$

where $\boldsymbol{\alpha} = \widetilde{\mathbf{K}}^{-1}\mathbf{y}$.

Hyper-parameter Learning

Let $\widetilde{\mathbf{K}} = \mathbf{K} + \sigma_n^2 \mathbf{I}$:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{1}{2}\mathbf{y}^\top \widetilde{\mathbf{K}}^{-1} \frac{\partial \widetilde{\mathbf{K}}}{\partial \theta_i} \widetilde{\mathbf{K}}^{-1}\mathbf{y} - \frac{1}{2}\ \mathrm{tr}\ \left(\widetilde{\mathbf{K}}^{-1} \frac{\partial \widetilde{\mathbf{K}}}{\partial \theta_i} \right)$$

$$= \frac{1}{2}\ \mathrm{tr}\ \left((\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \widetilde{\mathbf{K}}^{-1}) \frac{\partial \widetilde{\mathbf{K}}}{\partial \theta_i} \right)$$

where $\boldsymbol{\alpha} = \widetilde{\mathbf{K}}^{-1}\mathbf{y}$.

- Can use gradient-based optimization

Hyper-parameter Learning

Let $\widetilde{\mathbf{K}} = \mathbf{K} + \sigma_n^2 \mathbf{I}$:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{1}{2}\mathbf{y}^\top \widetilde{\mathbf{K}}^{-1} \frac{\partial \widetilde{\mathbf{K}}}{\partial \theta_i} \widetilde{\mathbf{K}}^{-1}\mathbf{y} - \frac{1}{2}\ \mathrm{tr}\ \left(\widetilde{\mathbf{K}}^{-1} \frac{\partial \widetilde{\mathbf{K}}}{\partial \theta_i} \right)$$

$$= \frac{1}{2}\ \mathrm{tr}\ \left((\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \widetilde{\mathbf{K}}^{-1}) \frac{\partial \widetilde{\mathbf{K}}}{\partial \theta_i} \right)$$

where $\boldsymbol{\alpha} = \widetilde{\mathbf{K}}^{-1}\mathbf{y}$.

- Can use gradient-based optimization
- General approach and only needs derivatives of the covariance

Hyper-parameter Learning

Let $\widetilde{\mathbf{K}} = \mathbf{K} + \sigma_n^2 \mathbf{I}$:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{1}{2}\mathbf{y}^\top \widetilde{\mathbf{K}}^{-1} \frac{\partial \widetilde{\mathbf{K}}}{\partial \theta_i} \widetilde{\mathbf{K}}^{-1}\mathbf{y} - \frac{1}{2}\,\mathrm{tr}\left(\widetilde{\mathbf{K}}^{-1}\frac{\partial \widetilde{\mathbf{K}}}{\partial \theta_i}\right)$$

$$= \frac{1}{2}\,\mathrm{tr}\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \widetilde{\mathbf{K}}^{-1})\frac{\partial \widetilde{\mathbf{K}}}{\partial \theta_i}\right)$$

where $\boldsymbol{\alpha} = \widetilde{\mathbf{K}}^{-1}\mathbf{y}$.

- Can use gradient-based optimization
- General approach and only needs derivatives of the covariance
- Such principled "kernel" learning does not exist in standard SVM

Hyper-parameter Learning

Let $\widetilde{\mathbf{K}} = \mathbf{K} + \sigma_n^2 \mathbf{I}$:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{1}{2}\mathbf{y}^\top \widetilde{\mathbf{K}}^{-1} \frac{\partial \widetilde{\mathbf{K}}}{\partial \theta_i} \widetilde{\mathbf{K}}^{-1}\mathbf{y} - \frac{1}{2}\ \mathrm{tr}\ \left(\widetilde{\mathbf{K}}^{-1}\frac{\partial \widetilde{\mathbf{K}}}{\partial \theta_i}\right)$$

$$= \frac{1}{2}\ \mathrm{tr}\ \left((\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \widetilde{\mathbf{K}}^{-1})\frac{\partial \widetilde{\mathbf{K}}}{\partial \theta_i}\right)$$

where $\boldsymbol{\alpha} = \widetilde{\mathbf{K}}^{-1}\mathbf{y}$.

- Can use gradient-based optimization
- General approach and only needs derivatives of the covariance
- Such principled "kernel" learning does not exist in standard SVM
- Non-convex optimization

Hyper-parameter Learning

Let $\widetilde{\mathbf{K}} = \mathbf{K} + \sigma_n^2 \mathbf{I}$:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{1}{2}\mathbf{y}^\top \widetilde{\mathbf{K}}^{-1} \frac{\partial \widetilde{\mathbf{K}}}{\partial \theta_i} \widetilde{\mathbf{K}}^{-1}\mathbf{y} - \frac{1}{2}\ \mathrm{tr}\left(\widetilde{\mathbf{K}}^{-1}\frac{\partial \widetilde{\mathbf{K}}}{\partial \theta_i}\right)$$

$$= \frac{1}{2}\ \mathrm{tr}\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \widetilde{\mathbf{K}}^{-1})\frac{\partial \widetilde{\mathbf{K}}}{\partial \theta_i}\right)$$

where $\boldsymbol{\alpha} = \widetilde{\mathbf{K}}^{-1}\mathbf{y}$.

- Can use gradient-based optimization
- General approach and only needs derivatives of the covariance
- Such principled "kernel" learning does not exist in standard SVM
- Non-convex optimization
- Multiple local optima correspond to different explanations of the data

Hyper-parameter Learning

Let $\widetilde{\mathbf{K}} = \mathbf{K} + \sigma_n^2 \mathbf{I}$:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{1}{2}\mathbf{y}^\top \widetilde{\mathbf{K}}^{-1} \frac{\partial \widetilde{\mathbf{K}}}{\partial \theta_i} \widetilde{\mathbf{K}}^{-1}\mathbf{y} - \frac{1}{2}\ \mathrm{tr}\left(\widetilde{\mathbf{K}}^{-1}\frac{\partial \widetilde{\mathbf{K}}}{\partial \theta_i}\right)$$

$$= \frac{1}{2}\ \mathrm{tr}\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \widetilde{\mathbf{K}}^{-1})\frac{\partial \widetilde{\mathbf{K}}}{\partial \theta_i}\right)$$

where $\boldsymbol{\alpha} = \widetilde{\mathbf{K}}^{-1}\mathbf{y}$.

- Can use gradient-based optimization
- General approach and only needs derivatives of the covariance
- Such principled "kernel" learning does not exist in standard SVM
- Non-convex optimization
- Multiple local optima correspond to different explanations of the data
- Computational Requirements?

Automatic Relevance Determination (ARD)

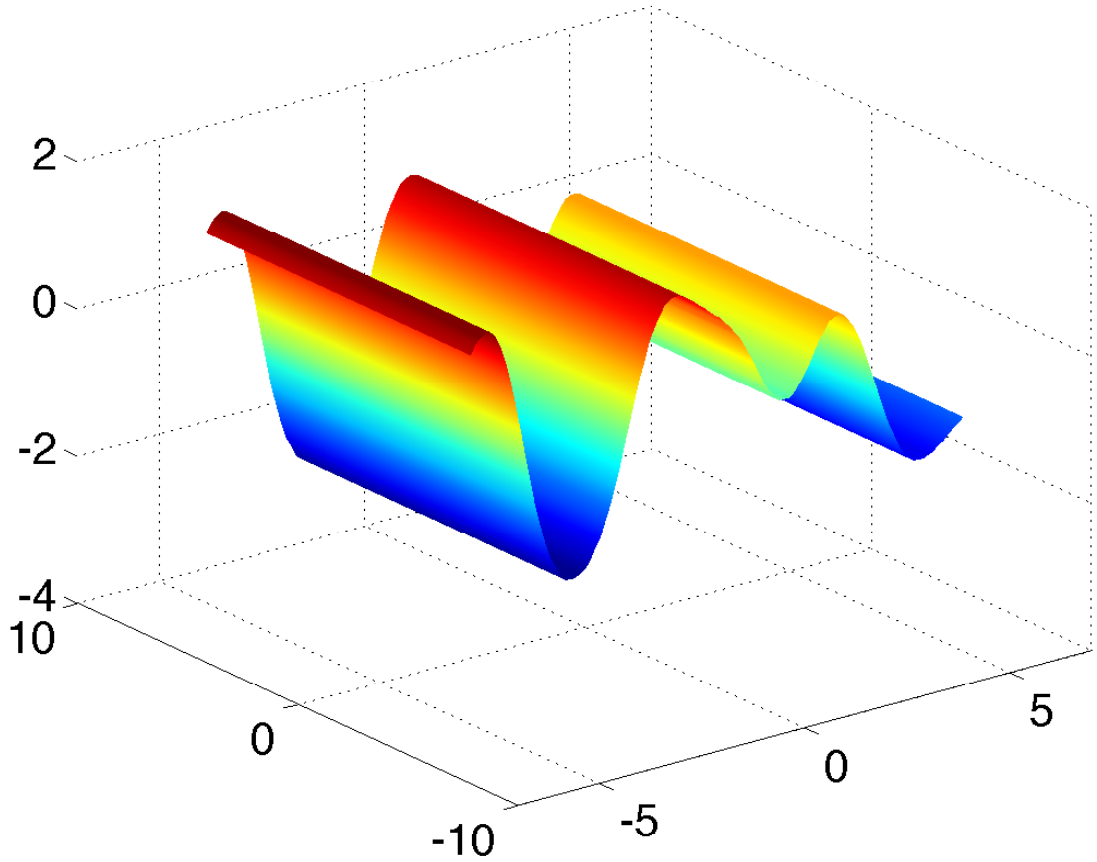
- Inverse of the length-scale determines the relevance of the dimension.

Automatic Relevance Determination (ARD)

- Inverse of the length-scale determines the relevance of the dimension.
- The larger the length-scale the more irrelevant the corresponding input is.

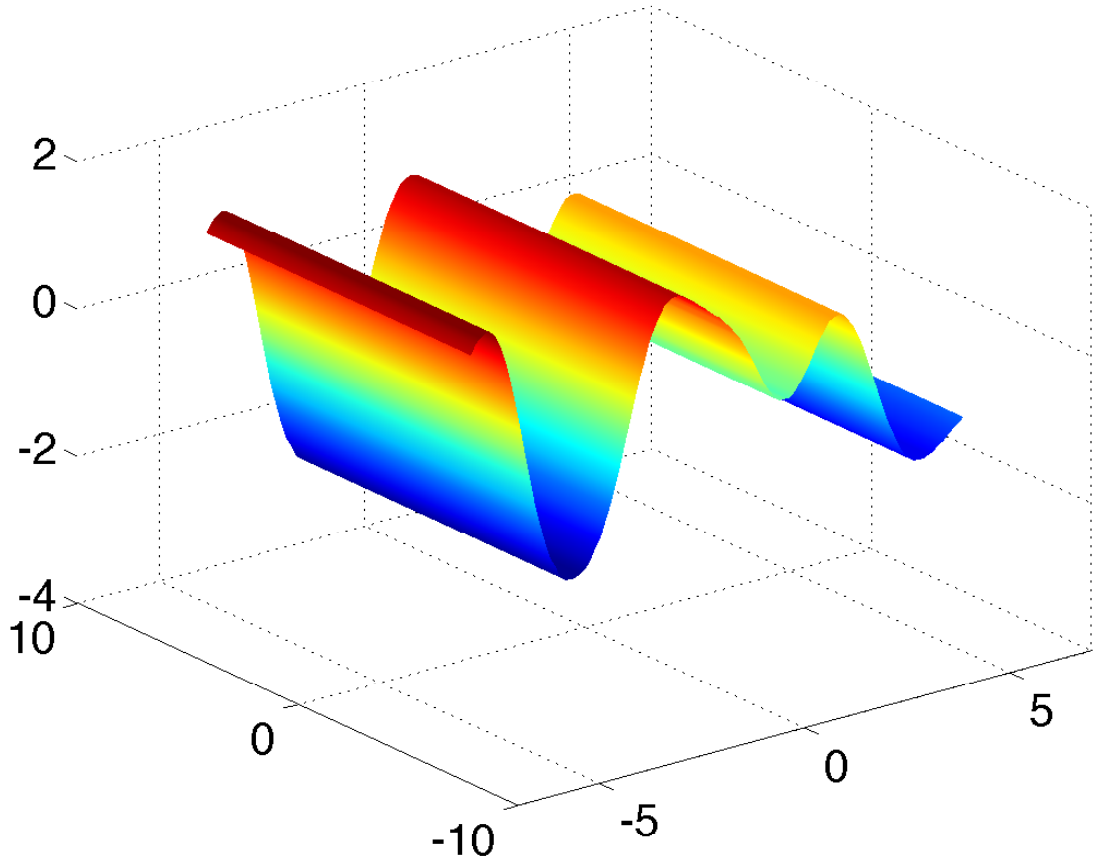
Automatic Relevance Determination (ARD)

- Inverse of the length-scale determines the relevance of the dimension.
- The larger the length-scale the more irrelevant the corresponding input is.



Automatic Relevance Determination (ARD)

- Inverse of the length-scale determines the relevance of the dimension.
- The larger the length-scale the more irrelevant the corresponding input is.



Learned lengh-scale for irrelevant dimension: 1.0557×10^5

Other Covariance Functions: Matérn Covariance

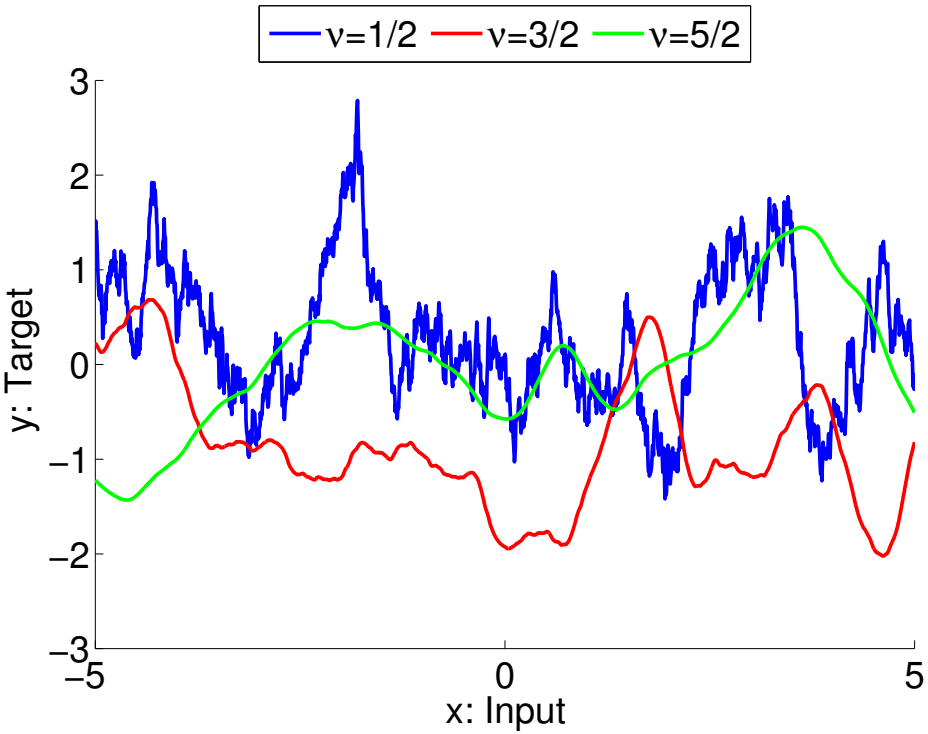
$$\kappa(\mathbf{x}, \mathbf{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\|\mathbf{x} - \mathbf{x}'\|}{\ell} \right)^{\nu} \mathcal{K}_{\nu} \left(\frac{\sqrt{2\nu}\|\mathbf{x} - \mathbf{x}'\|}{\ell} \right)$$

where $\mathcal{K}_{\nu}$ is a modified Bessel function and $\nu > 0$, $\ell > 0$.

Other Covariance Functions: Matérn Covariance

$$\kappa(\mathbf{x}, \mathbf{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\|\mathbf{x} - \mathbf{x}'\|}{\ell} \right)^{\nu} \mathcal{K}_{\nu} \left(\frac{\sqrt{2\nu}\|\mathbf{x} - \mathbf{x}'\|}{\ell} \right)$$

where $\mathcal{K}_{\nu}$ is a modified Bessel function and $\nu > 0$, $\ell > 0$.

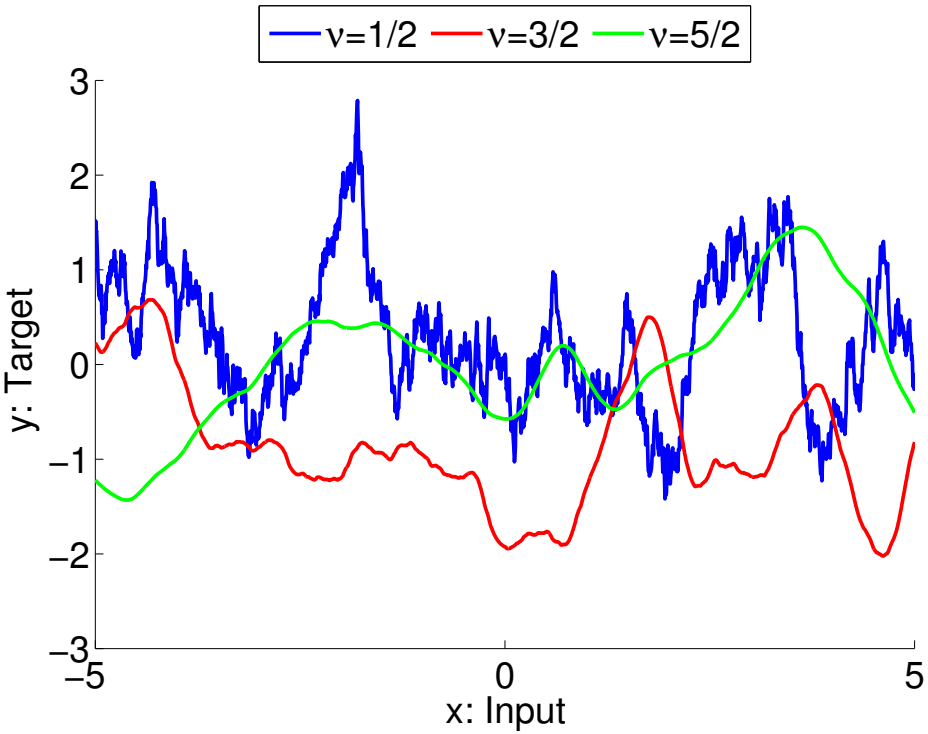


$\ell = 1$

Other Covariance Functions: Matérn Covariance

$$\kappa(\mathbf{x}, \mathbf{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\|\mathbf{x} - \mathbf{x}'\|}{\ell} \right)^{\nu} \mathcal{K}_{\nu} \left(\frac{\sqrt{2\nu}\|\mathbf{x} - \mathbf{x}'\|}{\ell} \right)$$

where $\mathcal{K}_{\nu}$ is a modified Bessel function and $\nu > 0$, $\ell > 0$.



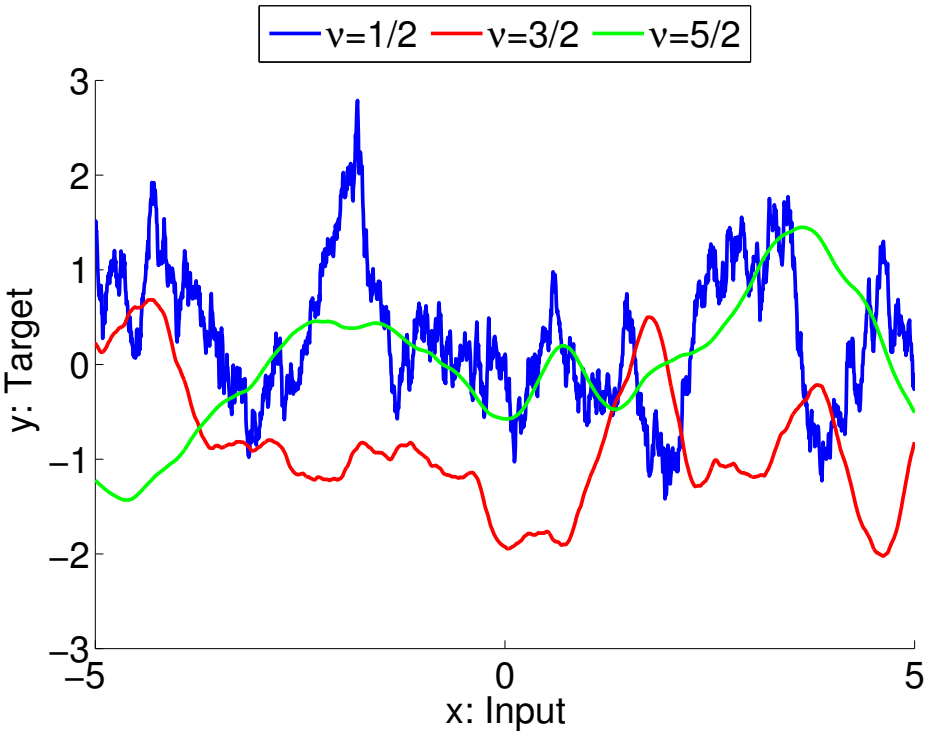
$\ell = 1$

- Stationary, Isotropic

Other Covariance Functions: Matérn Covariance

$$\kappa(\mathbf{x}, \mathbf{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\|\mathbf{x} - \mathbf{x}'\|}{\ell} \right)^{\nu} \mathcal{K}_{\nu} \left(\frac{\sqrt{2\nu}\|\mathbf{x} - \mathbf{x}'\|}{\ell} \right)$$

where $\mathcal{K}_{\nu}$ is a modified Bessel function and $\nu > 0$, $\ell > 0$.



$\ell = 1$

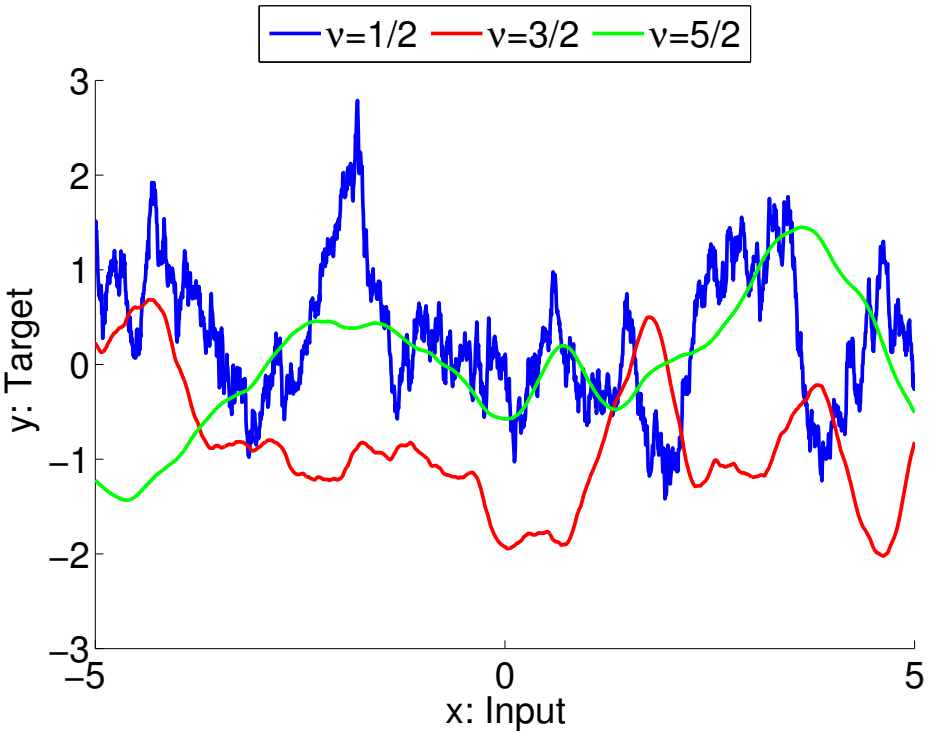
- Stationary, Isotropic
- $\nu = 1/2$:

  $\kappa(\mathbf{x}, \mathbf{x}') = \exp(-\frac{|\mathbf{x}-\mathbf{x}'|}{\ell})$
  - Very rough process
  - Brownian motion
  - Ornstein-Uhlenbeck (D=1)

Other Covariance Functions: Matérn Covariance

$$\kappa(\mathbf{x}, \mathbf{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\|\mathbf{x} - \mathbf{x}'\|}{\ell} \right)^{\nu} \mathcal{K}_{\nu} \left(\frac{\sqrt{2\nu}\|\mathbf{x} - \mathbf{x}'\|}{\ell} \right)$$

where $\mathcal{K}_{\nu}$ is a modified Bessel function and $\nu > 0$, $\ell > 0$.



$\ell = 1$

- Stationary, Isotropic
- $\nu = 1/2$:

  $\kappa(\mathbf{x}, \mathbf{x}') = \exp(-\frac{|\mathbf{x}-\mathbf{x}'|}{\ell})$
  - Very rough process
  - Brownian motion
  - Ornstein-Uhlenbeck (D=1)
- $\nu \to \infty$: SE covariance

Other Covariance Functions: Rational Quadratic

$$\kappa(\mathbf{x}, \mathbf{x}') = \left(1 + \frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\alpha\ell^2}\right)^{-\alpha}$$

with $\alpha > 0$, $\ell > 0$.

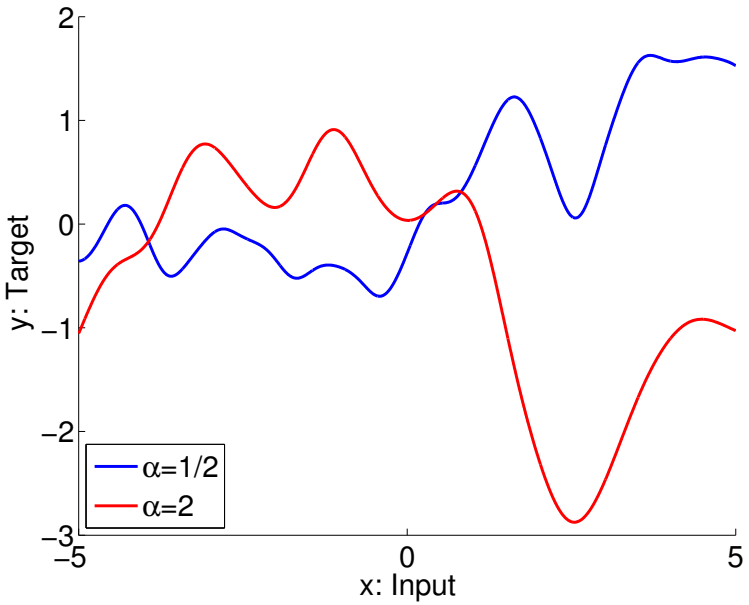
can be seen as an infinite sum of squared exponential (SE) covariance functions with different characteristic length-scales.

Other Covariance Functions: Rational Quadratic

$$\kappa(\mathbf{x}, \mathbf{x}') = \left(1 + \frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\alpha\ell^2}\right)^{-\alpha}$$

with $\alpha > 0$, $\ell > 0$.

can be seen as an infinite sum of squared exponential (SE) covariance functions with different characteristic length-scales.

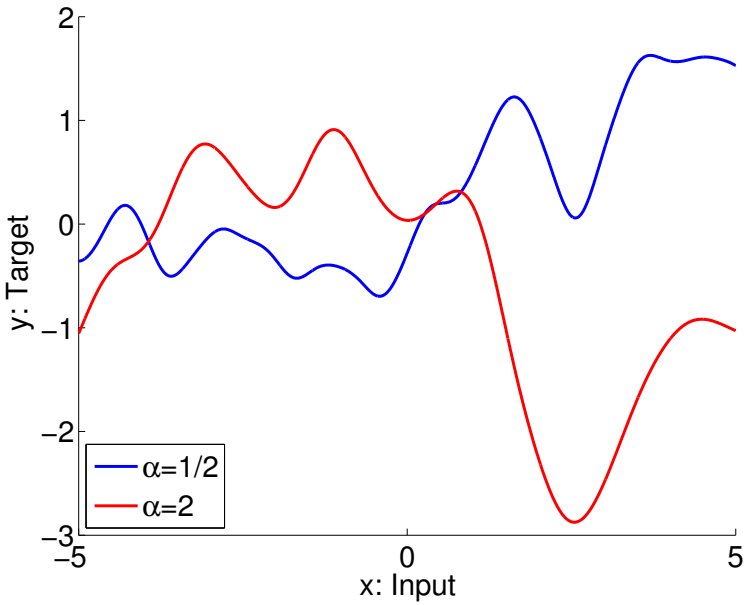


Other Covariance Functions: Rational Quadratic

$$\kappa(\mathbf{x}, \mathbf{x}') = \left(1 + \frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\alpha\ell^2}\right)^{-\alpha}$$

with $\alpha > 0$, $\ell > 0$.

can be seen as an infinite sum of squared exponential (SE) covariance functions with different characteristic length-scales.

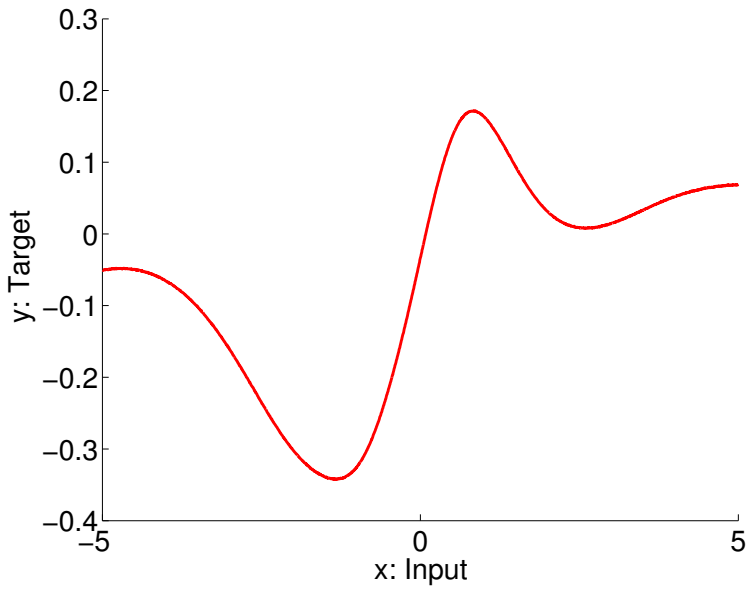


with $\alpha \to \infty$ is the SE covariance with length-scale ℓ.

Other Covariance Functions: Neural Network Covariance

- Consider a neural network with one hidden layer and $\mathrm{N_H}$ hidden units.
- Under certain assumptions the corresponding stochastic process will converge to a Gaussian Process as $\mathrm{N_H} \to \infty$.
- For a specific settings of the transfer function of the neural net:

$$\kappa(\mathbf{x}, \mathbf{x}') = \frac{2}{\pi} \sin^{-1}\left(\frac{2\tilde{\mathbf{x}}^{\mathsf{T}}\boldsymbol{\Sigma}\tilde{\mathbf{x}}'}{\sqrt{(1+2\tilde{\mathbf{x}}^{\mathsf{T}}\boldsymbol{\Sigma}\tilde{\mathbf{x}})(1+2\tilde{\mathbf{x}}'^{\mathsf{T}}\boldsymbol{\Sigma}\tilde{\mathbf{x}}')}}\right)$$



Other Covariance Functions: Periodic, Smooth Functions

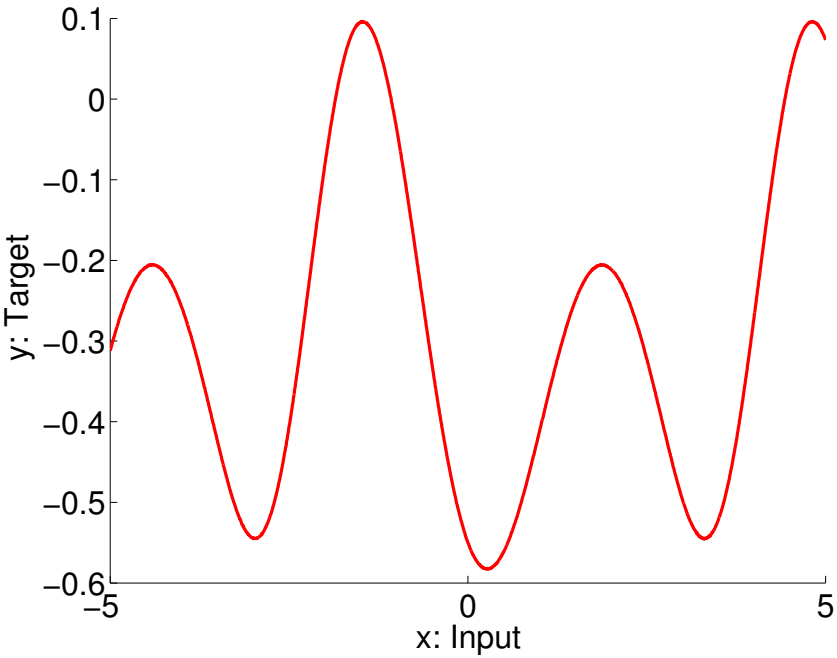
We can create a distribution over periodic functions of x by using the mapping $\mathbf{u}(x) = (\cos(x), \sin(x))$ and then use the SE covariance on $\mathbf{u}$ space. This gives rise to:

$$\kappa(x, x') = \exp\left(-\frac{2\sin^2(\frac{x-x'}{2})}{\ell^2}\right)$$

Other Covariance Functions: Periodic, Smooth Functions

We can create a distribution over periodic functions of x by using the mapping $\mathbf{u}(x) = (\cos(x), \sin(x))$ and then use the SE covariance on $\mathbf{u}$ space. This gives rise to:

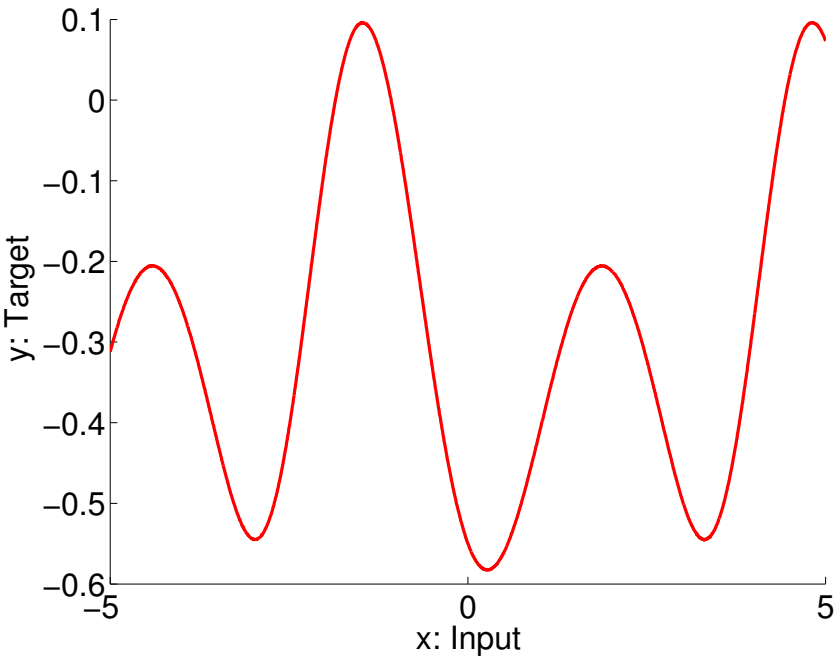
$$\kappa(x, x') = \exp\left(-\frac{2\sin^2\left(\frac{x-x'}{2}\right)}{\ell^2}\right)$$



Other Covariance Functions: Periodic, Smooth Functions

We can create a distribution over periodic functions of x by using the mapping $\mathbf{u}(x) = (\cos(x), \sin(x))$ and then use the SE covariance on $\mathbf{u}$ space. This gives rise to:

$$\kappa(x, x') = \exp\left(-\frac{2\sin^2\left(\frac{x-x'}{2}\right)}{\ell^2}\right)$$



This is called warping and can also be used to introduce non-stationarity.

Gaussian Process Classification: Introduction

- Targets are discrete

Gaussian Process Classification: Introduction

- Targets are discrete
- Examples: face recognition, digit recognition

Gaussian Process Classification: Introduction

- Targets are discrete
- Examples: face recognition, digit recognition
- We want probabilistic classifications

Gaussian Process Classification: Introduction

- Targets are discrete
- Examples: face recognition, digit recognition
- We want probabilistic classifications
- Can use decision theory for point prediction and e.g. zero-one loss

Gaussian Process Classification: Introduction

- Targets are discrete
- Examples: face recognition, digit recognition
- We want probabilistic classifications
- Can use decision theory for point prediction and e.g. zero-one loss
- Unlike the regression setting, in GP classification the non-Gaussian likelihood makes things analytically intractable

Gaussian Process Classification: Introduction

- Targets are discrete
- Examples: face recognition, digit recognition
- We want probabilistic classifications
- Can use decision theory for point prediction and e.g. zero-one loss
- Unlike the regression setting, in GP classification the non-Gaussian likelihood makes things analytically intractable
- Need approximations to the posterior, e.g. Laplace

Gaussian Process Classification: Introduction

- Targets are discrete
- Examples: face recognition, digit recognition
- We want probabilistic classifications
- Can use decision theory for point prediction and e.g. zero-one loss
- Unlike the regression setting, in GP classification the non-Gaussian likelihood makes things analytically intractable
- Need approximations to the posterior, e.g. Laplace
- Generative v discriminative $+$ and $-$?

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

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

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

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

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

Two popular approaches:

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

Linear Models for Classification

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

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

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

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

Two popular approaches:

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

MAP Approach

As in Bayesian linear regression we can use the prior:

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

MAP Approach

As in Bayesian linear regression we can use the prior:

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

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

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

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

MAP Approach

As in Bayesian linear regression we can use the prior:

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

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

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

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:

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

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

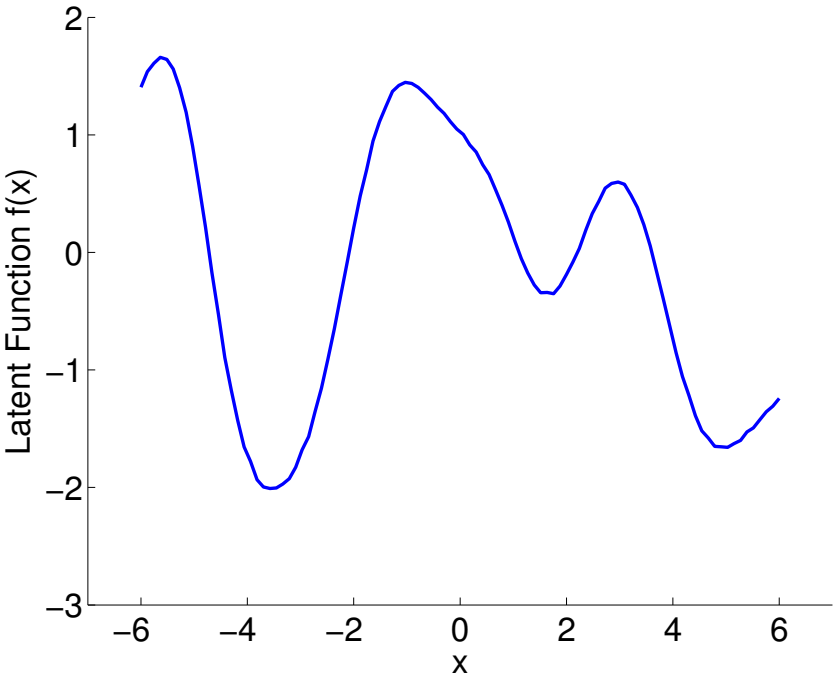
$$\mathcal{L}^{\text{MAP}} = \sum_{i=1}^{N} \log \sigma(y_i f_i) - \frac{1}{2}\mathbf{w}^T \boldsymbol{\Sigma}_w^{-1} \mathbf{w},$$

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)

Multi-class case is addressed with a softmax function.

Gaussian Process Classification (GPC)

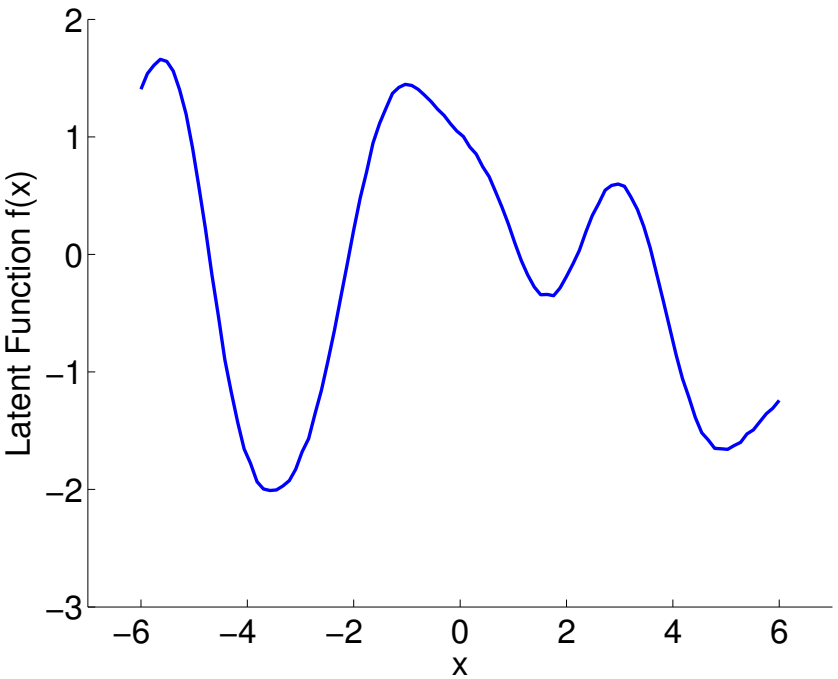
1. Place prior over the latent functions $\mathbf{f}(\mathbf{x})$



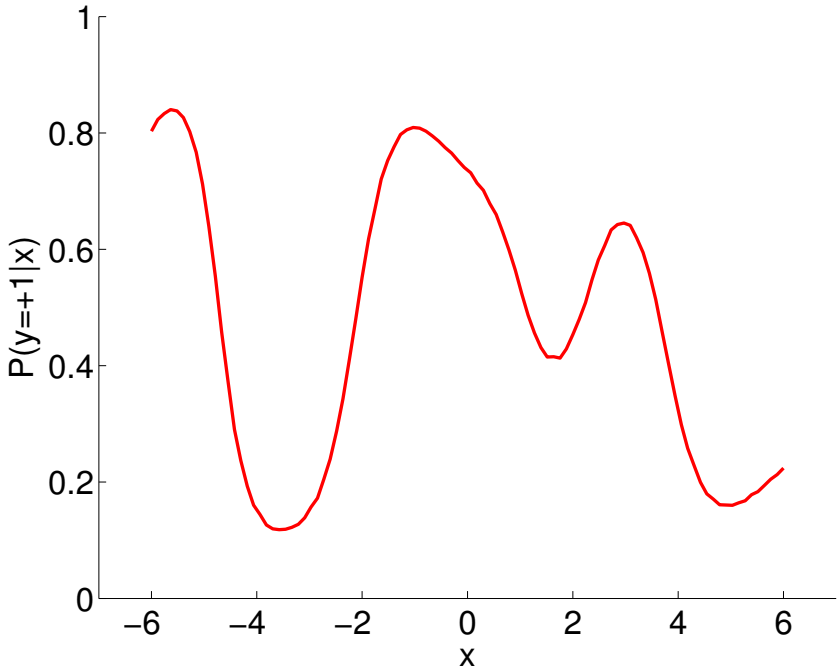
Sample from a GP

Gaussian Process Classification (GPC)

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



Sample from a GP



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

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:

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

   - 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:

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

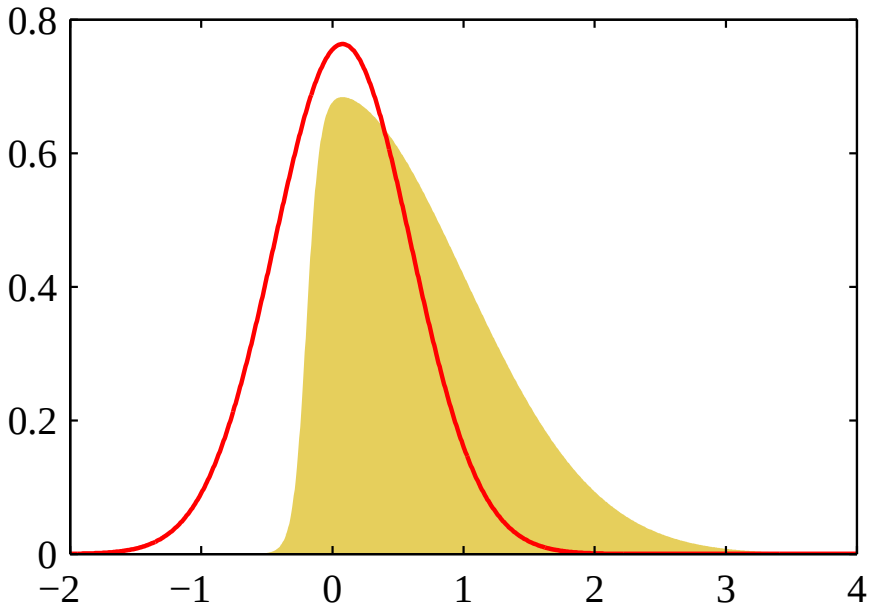
   - Analytic solution for the probit model
   - Require numerical approximations (1D) integral for other sigmoid functions

The Laplace Approximation

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

The Laplace Approximation

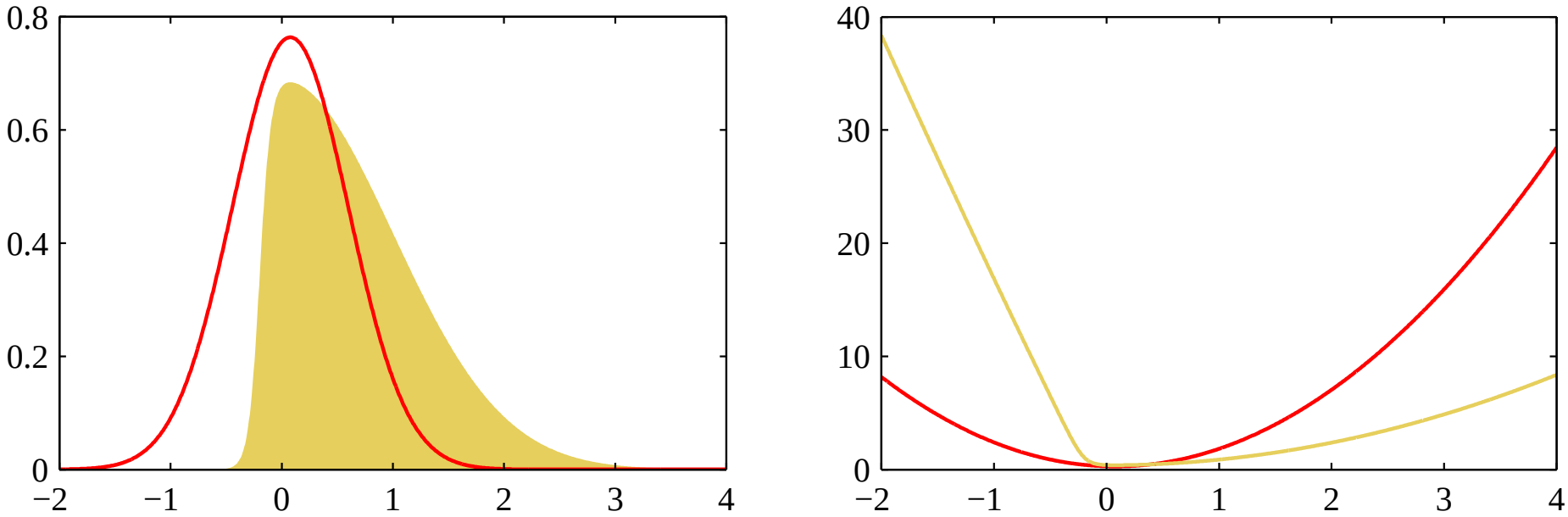
Idea: Find a Gaussian approximation to $p(z) = \frac{1}{Z} f(z)$, where Z is unknown. We centre the Gaussian approximation at the mode of $p(z)$.



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

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

where: $\hat{\mathbf{f}} = \text{argmax}_{\mathbf{f}}\, p(\mathbf{f}|\mathcal{D},\theta) = \text{argmax}_{\mathbf{f}}\; p(\mathcal{D}|\mathbf{f},\theta)p(\mathbf{f}|\theta)$ 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

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

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

Hence we focus on the maximization of:

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

The Laplace Approximation to the GP Binary Classifier

Gaussian approximation

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

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

Hence we focus on the maximization of:

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

Using Newton's method we obtain the following update:

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

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

The Laplace Approximation to the GP Binary Classifier

Gaussian approximation

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

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

Hence we focus on the maximization of:

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

Using Newton's method we obtain the following update:

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

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

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:

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

When is this approximation a good/bad idea?

Posterior and Predictive Distributions

Recalling the posterior distribution:

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

Posterior and Predictive Distributions

Recalling the posterior distribution:

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

Hence, under Laplace approximation:

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

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

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:

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

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

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:

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

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

- 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:

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

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

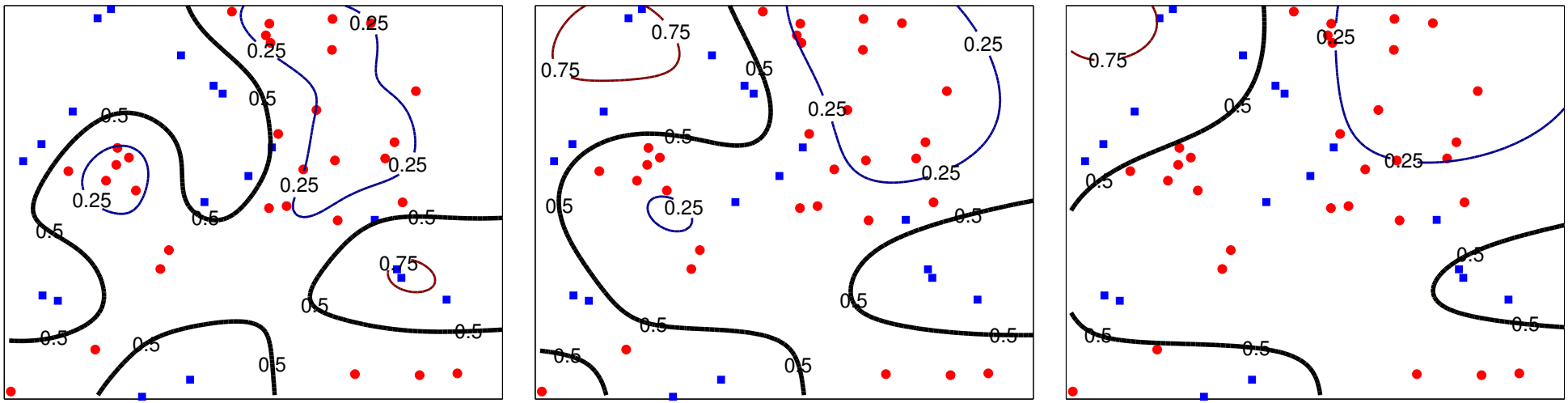
- They provide the same prediction when concerned with most probable classification
- Full distribution is required if we are concerned with confidence in the predictions (e.g. reject options)

Marginal Likelihood and hyper-parameter learning

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

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

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?

GPR's Computational Complexity

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

GPR's 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: $\widetilde{\mathbf{K}}\mathbf{b} = \mathbf{y}$

GPR's 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):

GPR's 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

GPR's 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: $\mathcal{O}(IN^2)$

GPR's 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

GPR's 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:

GPR's 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 $\tilde{N}$ inducing points)

GPR's 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 $\tilde{N}$ inducing points)
  - Apply matrix computational tricks (e.g. block inverses, Woodbury's formula)

GPR's 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 $\tilde{N}$ inducing points)
  - Apply matrix computational tricks (e.g. block inverses, Woodbury's formula)
  - Computations are usually $\mathcal{O}(\tilde{N}^2 N)$

GPR's 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 $\tilde{N}$ inducing points)
  - Apply matrix computational tricks (e.g. block inverses, Woodbury's formula)
  - Computations are usually $\mathcal{O}(\tilde{N}^2 N)$
  - 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 $\tilde{\mathrm{N}} \rightarrow \mathcal{O}(\tilde{\mathrm{N}}^3)$

Subset of Data-points (SD)

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

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

where ν_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 $\tilde{N} \rightarrow \mathcal{O}(\tilde{N}^3)$
- $\tilde{N}$ 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:

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

where ν_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)

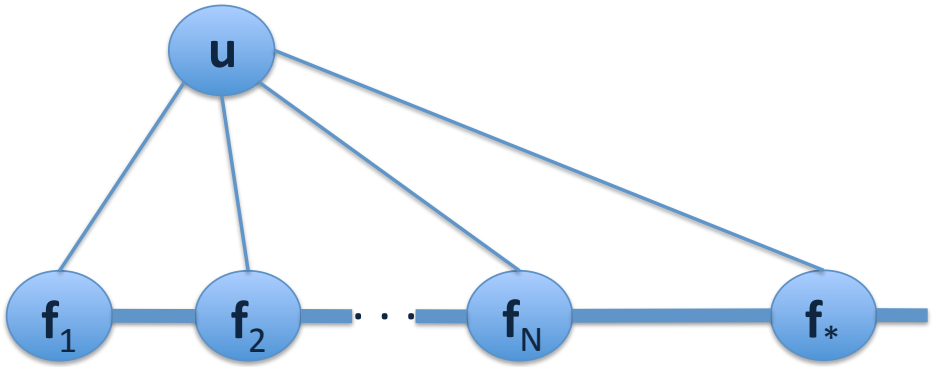
- Simplest approach: throw data away
- Keep the GP predictor on smaller set of size $\tilde{N} \rightarrow \mathcal{O}(\tilde{N}^3)$
- $\tilde{N}$ data-points can be selected at random
- Alternatively, they can be selected in a greedy fashion in order to optimize an objective function.
- Lawrence et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

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

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

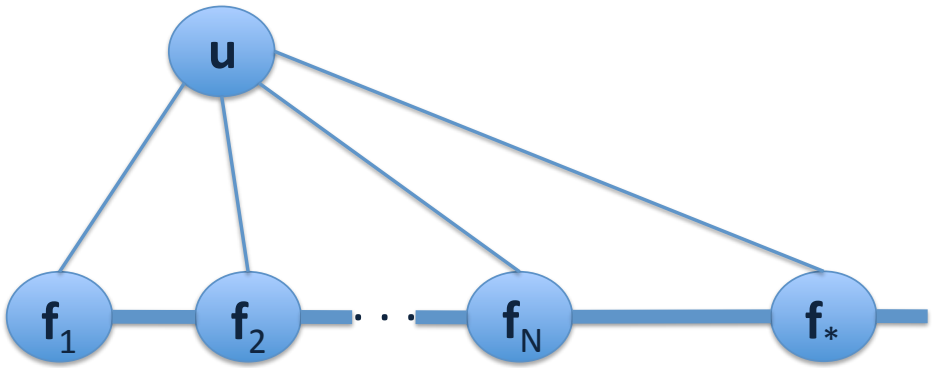
  - Simply choose the site with largest variance!
  - Overall complexity: $\mathcal{O}(\tilde{N}^2 N)$

GP Approximations: A Unifying Framework

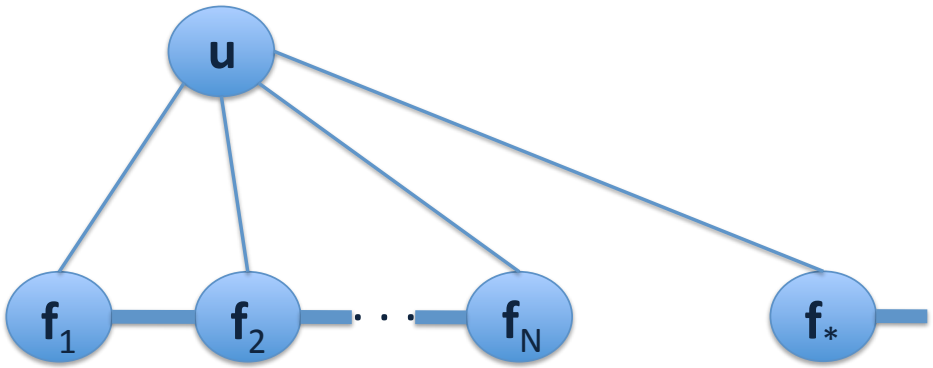


Full GP (no approximations). All latent functions are fully connected.

GP Approximations: A Unifying Framework

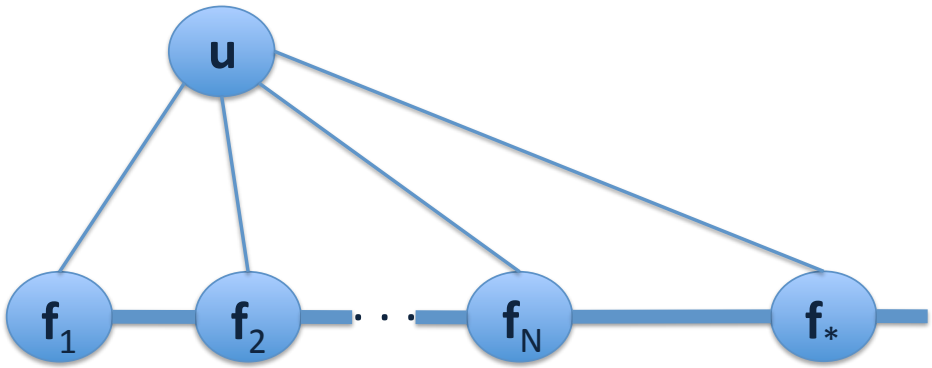


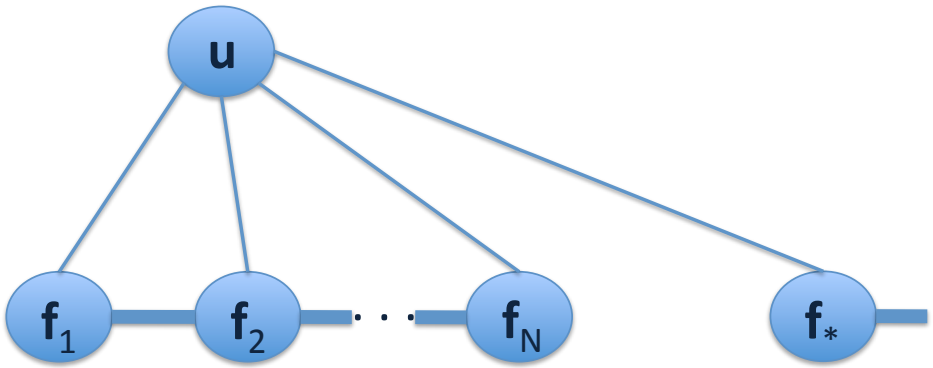
Full GP (no approximations). All latent functions are fully connected.



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

GP Approximations: A Unifying Framework





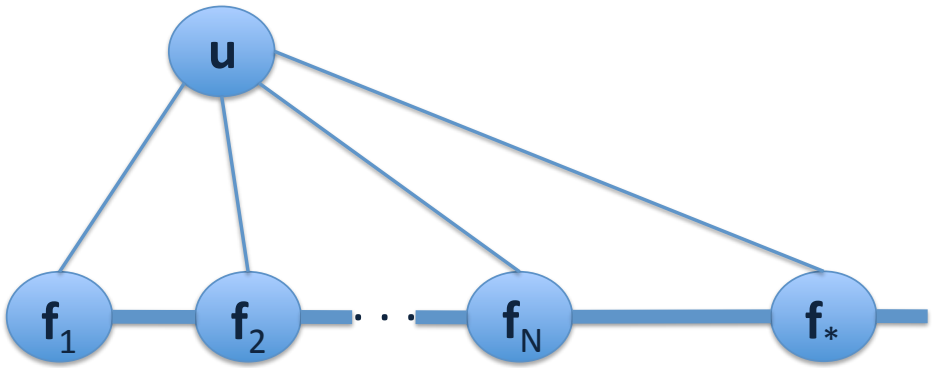
Full GP (no approximations). All latent functions are fully connected.

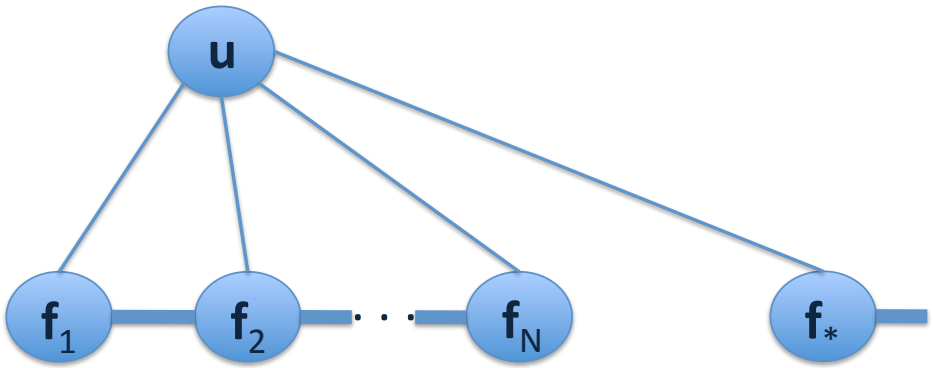
Traning and test values are conditionally independent given $\mathbf{u}$

The joint prior is modified through the inducing variables $\mathbf{u}_1, \ldots, \mathbf{u}_{\tilde{N}}$:

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

GP Approximations: A Unifying Framework





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 $\mathbf{u}_1, \ldots, \mathbf{u}_{\tilde{N}}$:

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

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

Most approximation methods can be defined by:

- Different specifications of these conditionals.
- Different $\mathbf{X}_\mathbf{u}$: 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:

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

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

Subset of Regressors (SR)

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

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

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

We can truncate the number of regressors needed:

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

Subset of Regressors (SR)

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

$$\text{Prior} \;:\; \boldsymbol{\alpha} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}^{-1})$$
$$\text{Model} \;:\; \mathrm{f}(\mathbf{x}_*) = \sum_{\mathrm{i}=1}^{\mathrm{N}} \alpha_{\mathrm{i}} \mathrm{k}(\mathbf{x}_*, \mathbf{x}_{\mathrm{i}})$$

We can truncate the number of regressors needed:

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

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

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

Subset of Regressors (SR)

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

$$\text{Prior} : \boldsymbol{\alpha} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}^{-1})$$
$$\text{Model} : \mathrm{f}(\mathbf{x}_*) = \sum_{\mathrm{i}=1}^{\mathrm{N}} \alpha_{\mathrm{i}} \mathrm{k}(\mathbf{x}_*, \mathbf{x}_{\mathrm{i}})$$

We can truncate the number of regressors needed:

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

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

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

Hence the predictive distribution is given by:

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

where $\boldsymbol{\Sigma} = \mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{K}_{\mathbf{f},\mathbf{u}} + \sigma_{\mathrm{n}}^2\mathbf{K}_{\mathbf{u},\mathbf{u}}$.

Subset of Regressors (SR)

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

$$\text{Prior} : \boldsymbol{\alpha} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}^{-1})$$
$$\text{Model} : \mathrm{f}(\mathbf{x}_*) = \sum_{\mathrm{i}=1}^{\mathrm{N}} \alpha_{\mathrm{i}} \mathrm{k}(\mathbf{x}_*, \mathbf{x}_{\mathrm{i}})$$

We can truncate the number of regressors needed:

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

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

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

Hence the predictive distribution is given by:

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

where $\boldsymbol{\Sigma} = \mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{K}_{\mathbf{f},\mathbf{u}} + \sigma_{\mathrm{n}}^2\mathbf{K}_{\mathbf{u},\mathbf{u}}$.

- 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:

$$\text{Prior} : \boldsymbol{\alpha} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}^{-1})$$
$$\text{Model} : \mathrm{f}(\mathbf{x}_*) = \sum_{\mathrm{i}=1}^{\mathrm{N}} \alpha_{\mathrm{i}} \mathrm{k}(\mathbf{x}_*, \mathbf{x}_{\mathrm{i}})$$

We can truncate the number of regressors needed:

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

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

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

Hence the predictive distribution is given by:

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

where $\boldsymbol{\Sigma} = \mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{K}_{\mathbf{f},\mathbf{u}} + \sigma_{\mathrm{n}}^2\mathbf{K}_{\mathbf{u},\mathbf{u}}$.

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

Projected Processes (PP)

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

- 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
- 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

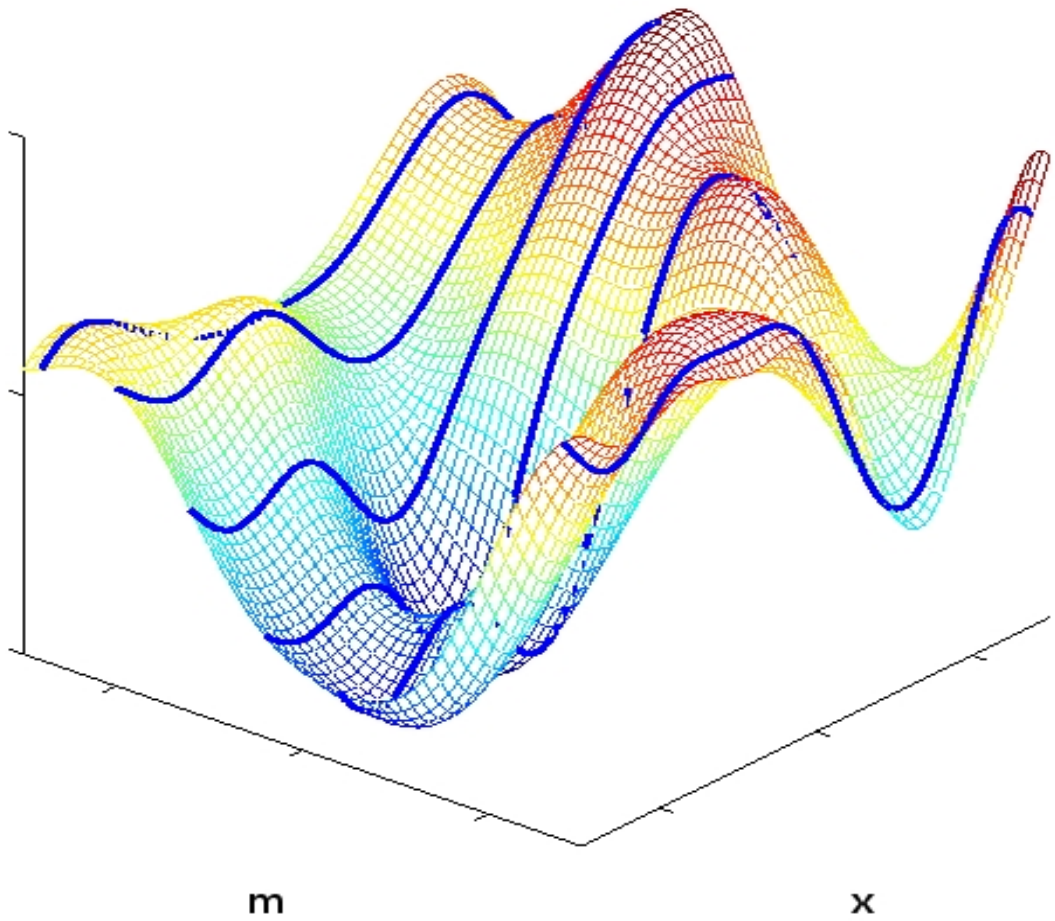
Multi-task Learning (MTL)

- 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)

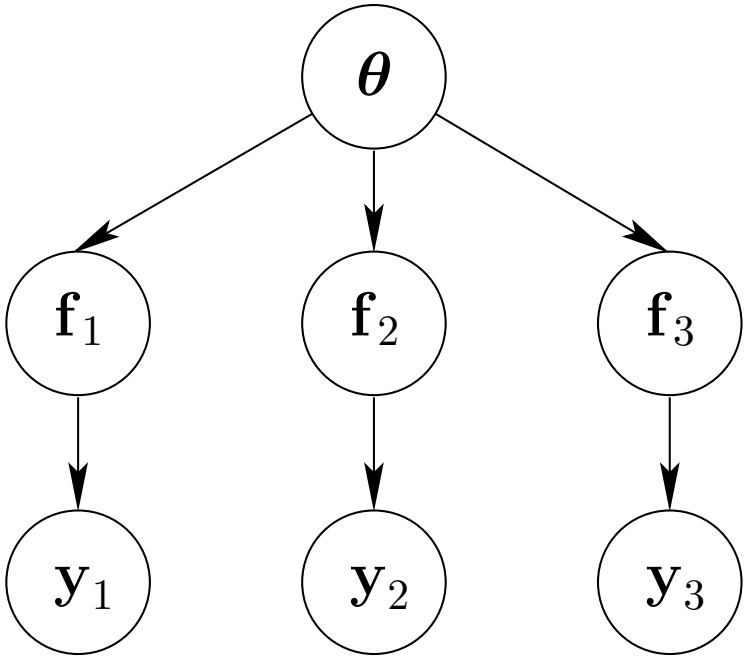
- General idea:
  - Sharing information across tasks (Caruana, 1997)
  - Very little data on test task
  - Exam score prediction, compiler performance prediction, robot inverse dynamics, multi-topic text categorisation, collaborative filtering, multi-level modelling
- Assuming task relatedness can be 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 m axis) are correlated (cf *independent* draws over sample functions)

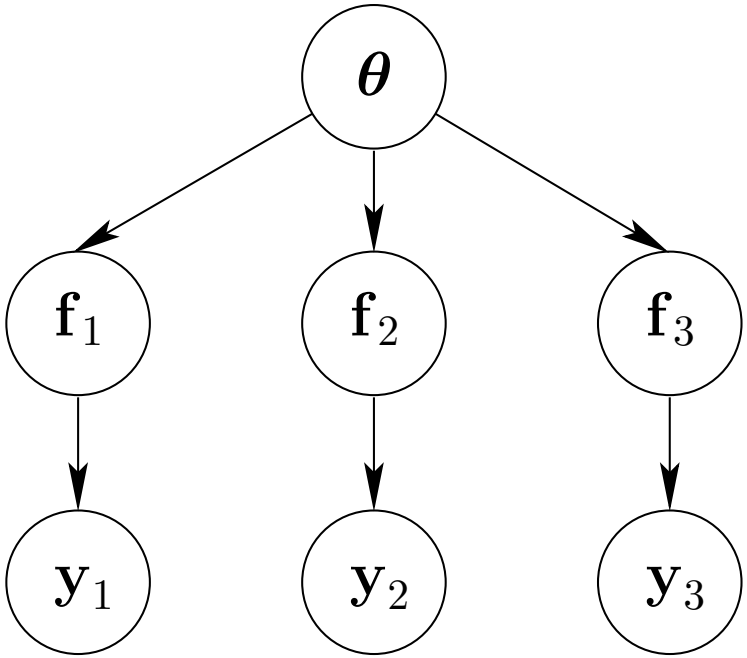
Inter-task Tying by Hyper-parameter Sharing



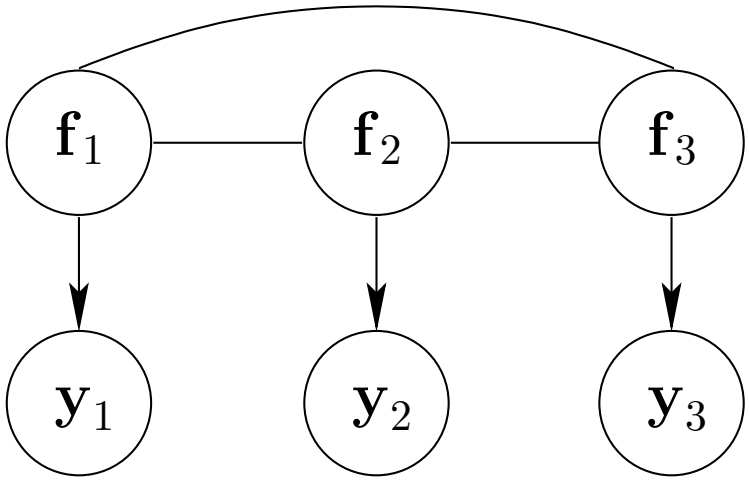
Other approaches

- Block diagonal covariance matrix, and each of the M 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



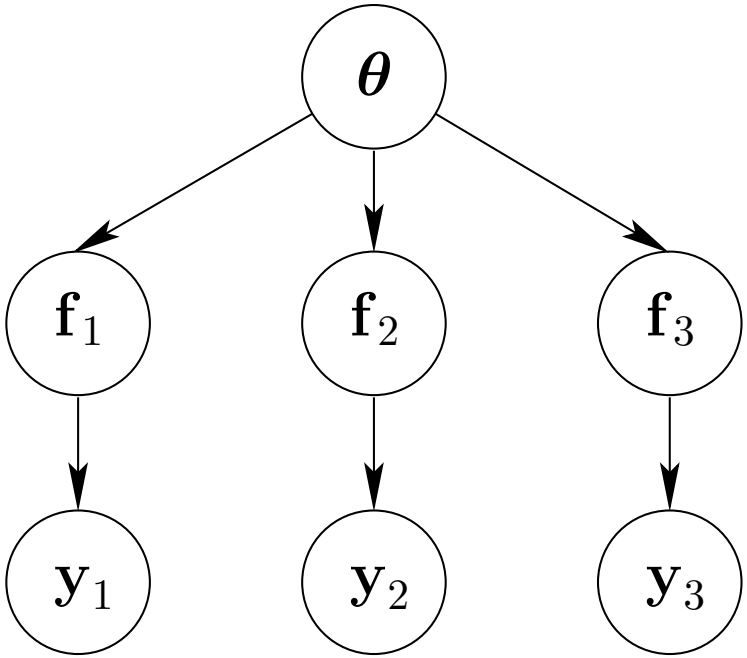
Other approaches



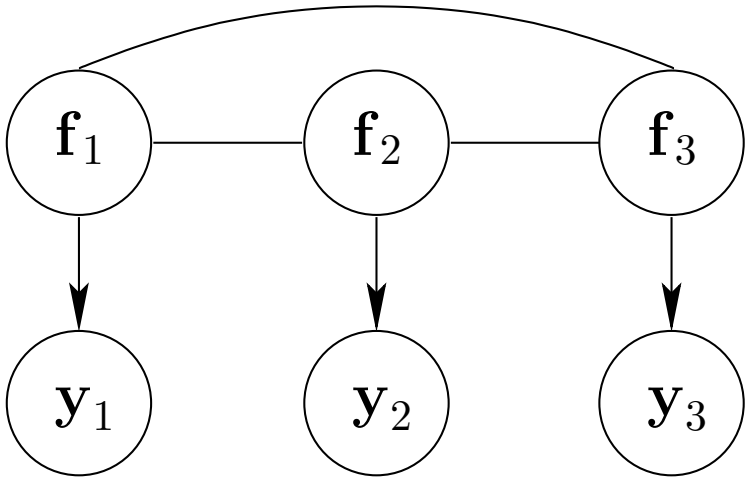
Our approach

- Block diagonal covariance matrix, and each of the M 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



Our approach

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

Multi-task GP

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

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

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. $\mathrm{K}^{\mathrm{f}} = (\mathrm{L}^{\mathrm{f}})(\mathrm{L}^{\mathrm{f}})^{\mathrm{T}}$

Multi-task GP Models

K^{f} can be:

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

Multi-task GP Models

K^{f} can be:

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

Multi-task GP Models

K^{f} can be:

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

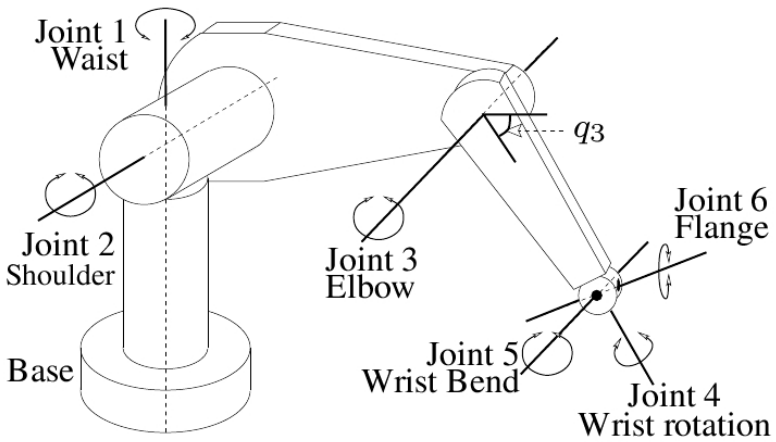
K^{f} can be:

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

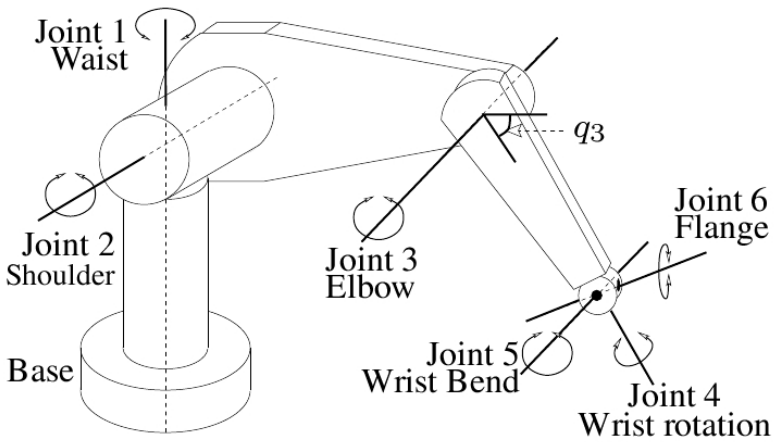
$$\mathrm{K}^{\mathrm{f}} = \begin{pmatrix} \mathrm{I} & \boldsymbol{\pi} \\ \boldsymbol{\pi}^{\mathsf{T}} & \boldsymbol{\pi}^{\mathsf{T}}\boldsymbol{\pi} \end{pmatrix},$$

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

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

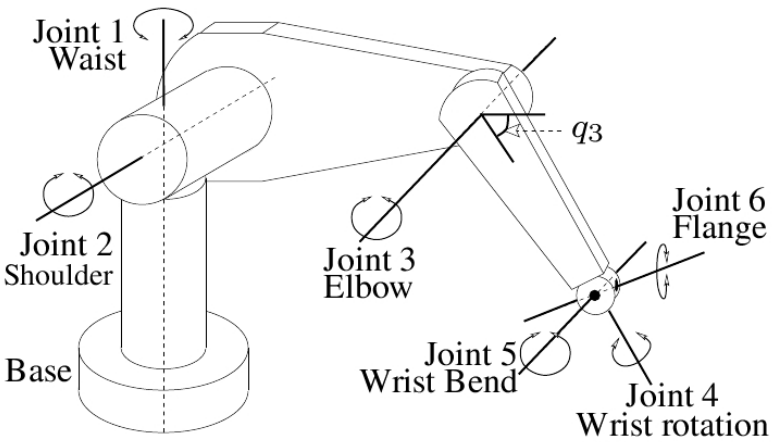


Learning Robot Inverse Dynamics (Chai et al, NIPS 2009)



- $\boldsymbol{\tau}$: 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)

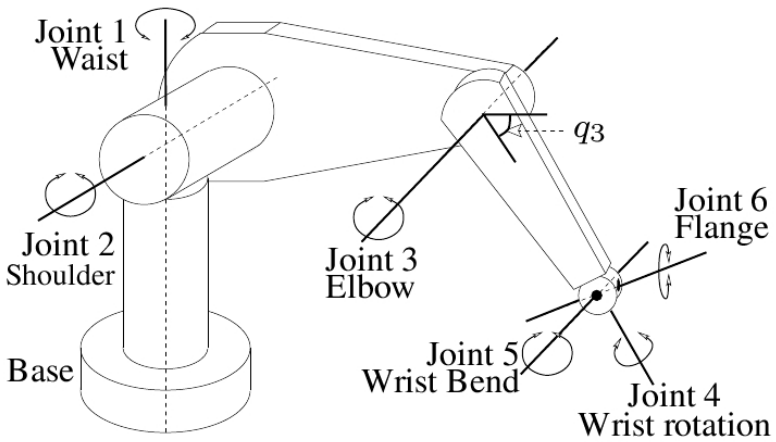
Learning Robot Inverse Dynamics (Chai et al, NIPS 2009)



- $\boldsymbol{\tau}$: 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)

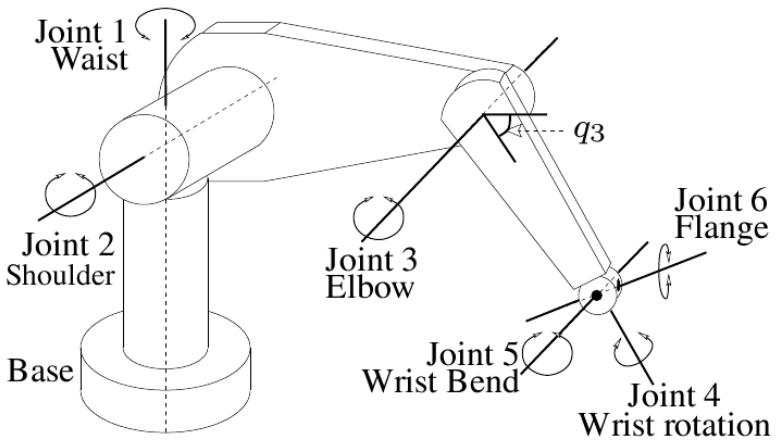


- $\boldsymbol{\tau}$: Torques needed at joints to drive a trajectory $\mathbf{x} = (\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$
- Unfeasible analytical model, e.g. friction, uncertainty in physical parameters
- Need to be controlled while having different loads (tasks)

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

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

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



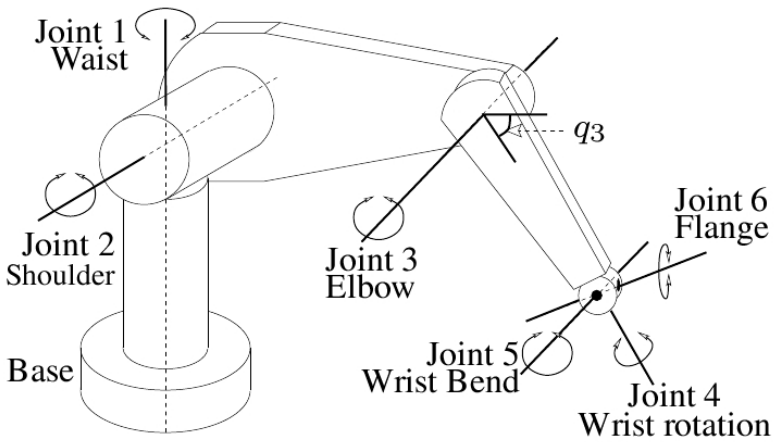
- $\boldsymbol{\tau}$: Torques needed at joints to drive a trajectory $\mathbf{x} = (\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$
- Unfeasible analytical model, e.g. friction, uncertainty in physical parameters
- Need to be controlled while having different loads (tasks)

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

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

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

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

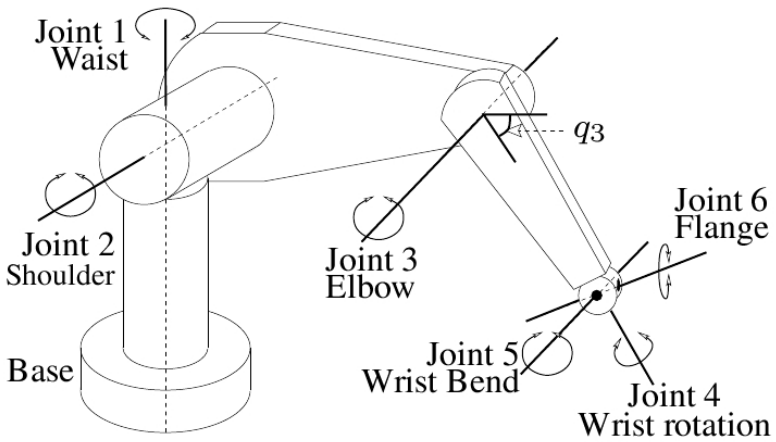
$$\tau_j^m(\mathbf{x}) = \mathbf{z}_j(\mathbf{x})^\top \boldsymbol{\rho}_j^m$$

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

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

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

The MTGP model matches the correlations between torque functions

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 (to appear)
  - 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

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.
- 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 and Future Directions

- 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

- More work on design of covariance functions needed
- Towards real large scale GPs
- Dealing with non-standard settings, e.g. preference learning and multi-task learning
- Dealing with structured data

GP Quiz