Gaussian Processes for Nonlinear Regression and Nonlinear Dimensionality Reduction

Piyush Rai
IIT Kanpur

Probabilistic Machine Learning (CS772A)

Feb 10, 2016

Gaussian Process

- A Gaussian Process (GP) is a distribution over functions
- A random draw from a GP thus gives a function f

$$f \sim \text{GP}(\mu, \kappa)$$

where $\boldsymbol{\mu}$ is the mean function and κ is the covariance/kernel function (the cov. function controls f's shape/smoothness)

- Note: μ and κ can be chosen or learned from data

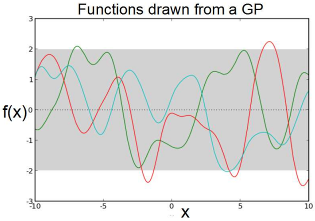
Gaussian Process

- A Gaussian Process (GP) is a distribution over functions
- A random draw from a GP thus gives a function f

$$f \sim \text{GP}(\mu, \kappa)$$

  where $\boldsymbol{\mu}$ is the mean function and κ is the covariance/kernel function (the cov. function controls f's shape/smoothness)

- Note: μ and κ can be chosen or learned from data



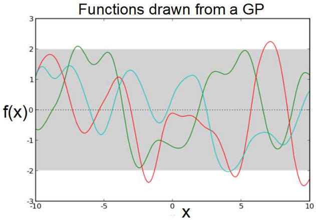
Gaussian Process

- A Gaussian Process (GP) is a distribution over functions
- A random draw from a GP thus gives a function f

$$f \sim \mathrm{GP}(\mu, \kappa)$$

where $\boldsymbol{\mu}$ is the mean function and κ is the covariance/kernel function (the cov. function controls f's shape/smoothness)

- Note: μ and κ can be chosen or learned from data



- GP can be used as a nonparametric prior distribution for such functions

Gaussian Process

- A function f is said to be drawn from $\mathrm{GP}(\mu, \kappa)$ if

$$\begin{bmatrix} f(\boldsymbol{x}_1) \\ f(\boldsymbol{x}_2) \\ \vdots \\ f(\boldsymbol{x}_N) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\boldsymbol{x}_1) \\ \mu(\boldsymbol{x}_2) \\ \vdots \\ \mu(\boldsymbol{x}_N) \end{bmatrix}, \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) \ldots \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \kappa(\boldsymbol{x}_2, \boldsymbol{x}_1) \ldots \kappa(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) \ldots \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix} \right)$$

Gaussian Process

- A function f is said to be drawn from $\text{GP}(\mu, \kappa)$ if

$$\begin{bmatrix} f(\boldsymbol{x}_1) \\ f(\boldsymbol{x}_2) \\ \vdots \\ f(\boldsymbol{x}_N) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\boldsymbol{x}_1) \\ \mu(\boldsymbol{x}_2) \\ \vdots \\ \mu(\boldsymbol{x}_N) \end{bmatrix}, \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \kappa(\boldsymbol{x}_2, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots \quad\quad \ddots \quad\quad \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix} \right)$$

- Thus, if f is drawn from a GP then the joint distribution of f's evaluations at a finite set of points $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\}$ is a multivariate normal

Gaussian Process

- Let's define

$$\mathbf{f} = \begin{bmatrix} f(\boldsymbol{x}_1) \\ f(\boldsymbol{x}_2) \\ \vdots \\ f(\boldsymbol{x}_N) \end{bmatrix}, \boldsymbol{\mu} = \begin{bmatrix} \mu(\boldsymbol{x}_1) \\ \mu(\boldsymbol{x}_2) \\ \vdots \\ \mu(\boldsymbol{x}_N) \end{bmatrix}, \mathbf{K} = \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \kappa(\boldsymbol{x}_2, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix}$$

Note: $\mathbf{K}$ is also called the kernel matrix. $K_{nm} = \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m)$

Gaussian Process

- Let's define

$$\mathbf{f} = \begin{bmatrix} f(\boldsymbol{x}_1) \\ f(\boldsymbol{x}_2) \\ \vdots \\ f(\boldsymbol{x}_N) \end{bmatrix}, \boldsymbol{\mu} = \begin{bmatrix} \mu(\boldsymbol{x}_1) \\ \mu(\boldsymbol{x}_2) \\ \vdots \\ \mu(\boldsymbol{x}_N) \end{bmatrix}, \mathbf{K} = \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \kappa(\boldsymbol{x}_2, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix}$$

  Note: $\mathbf{K}$ is also called the kernel matrix. $K_{nm} = \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m)$

- Thus we have

$$\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$$

Gaussian Process

- Let's define

$$\mathbf{f} = \begin{bmatrix} f(\boldsymbol{x}_1) \\ f(\boldsymbol{x}_2) \\ \vdots \\ f(\boldsymbol{x}_N) \end{bmatrix}, \boldsymbol{\mu} = \begin{bmatrix} \mu(\boldsymbol{x}_1) \\ \mu(\boldsymbol{x}_2) \\ \vdots \\ \mu(\boldsymbol{x}_N) \end{bmatrix}, \mathbf{K} = \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \kappa(\boldsymbol{x}_2, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix}$$

  Note: $\mathbf{K}$ is also called the kernel matrix. $K_{nm} = \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m)$

- Thus we have

$$\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$$

- Often, we assume the mean function to be zero. Thus $\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$

Gaussian Process

- Let's define

$$\mathbf{f} = \begin{bmatrix} f(\boldsymbol{x}_1) \\ f(\boldsymbol{x}_2) \\ \vdots \\ f(\boldsymbol{x}_N) \end{bmatrix}, \boldsymbol{\mu} = \begin{bmatrix} \mu(\boldsymbol{x}_1) \\ \mu(\boldsymbol{x}_2) \\ \vdots \\ \mu(\boldsymbol{x}_N) \end{bmatrix}, \mathbf{K} = \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \kappa(\boldsymbol{x}_2, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix}$$

  Note: $\mathbf{K}$ is also called the kernel matrix. $K_{nm} = \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m)$

- Thus we have

$$\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$$

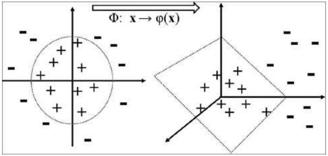
- Often, we assume the mean function to be zero. Thus $\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$
- Covariance/kernel function κ measures similarity between two inputs
  - $\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\frac{||\boldsymbol{x}_n - \boldsymbol{x}_m||^2}{\gamma}\right)$: RBF kernel
  - $\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) = v_0 \exp\left\{-\left(\frac{|\boldsymbol{x}_n - \boldsymbol{x}_m|}{r}\right)^\alpha\right\} + v_1 + v_2\delta_{nm}$

Kernel Functions

- Covariance/kernel function κ measures similarity between two inputs
- Corresponds to implicitly mapping data to a higher dimensional space via a feature mapping ϕ ($\boldsymbol{x} \rightarrow \phi(\boldsymbol{x})$) and computing the dot product that space

$$\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) = \phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m)$$

- Popularly known as the kernel trick (used in kernel methods for nonlinear regression/classification/clustering/dimensionality reduction, etc.)
- Allows extending linear models to nonlinear problems



Today's Plan

Gaussian Processes for two problems

- Nonlinear Regression: Gaussian Process Regression
- Nonlinear Dimensionality Reduction: Gaussian Process Latent Variable Models (GPLVM)

Gaussian Process Regression

Gaussian Process Regression

- Training data $\mathcal{D}$: $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$. $\boldsymbol{x}_n \in \mathbb{R}^D$, $y_n \in \mathbb{R}$
- Assume the responses to be a noisy function of the inputs

$$y_n = f(\boldsymbol{x}_n) + \epsilon_n = f_n + \epsilon_n$$

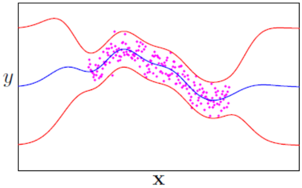
- Don't *a priori* know the form of f (linear/polynomial/something else?)

Gaussian Process Regression

- Training data $\mathcal{D}$: $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$. $\boldsymbol{x}_n \in \mathbb{R}^D$, $y_n \in \mathbb{R}$
- Assume the responses to be a noisy function of the inputs

$$y_n = f(\boldsymbol{x}_n) + \epsilon_n = f_n + \epsilon_n$$

- Don't *a priori* know the form of f (linear/polynomial/something else?)
- Want to learn f with error bars

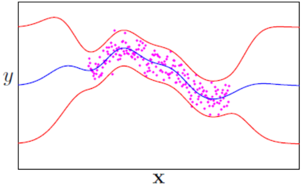


Gaussian Process Regression

- Training data $\mathcal{D}$: $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$. $\boldsymbol{x}_n \in \mathbb{R}^D$, $y_n \in \mathbb{R}$
- Assume the responses to be a noisy function of the inputs

$$y_n = f(\boldsymbol{x}_n) + \epsilon_n = f_n + \epsilon_n$$

- Don't *a priori* know the form of f (linear/polynomial/something else?)
- Want to learn f with error bars



- We'll use GP prior on f and use Bayes rule to get the posterior on f

$$p(f|\mathcal{D}) = \frac{p(f)p(\mathcal{D}|f)}{p(\mathcal{D})}$$

Gaussian Process Regression

- Training data: $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$. $\boldsymbol{x}_n \in \mathbb{R}^D$, $y_n \in \mathbb{R}$
- Assume the responses to be a noisy function of the inputs

$$y_n = f(\boldsymbol{x}_n) + \epsilon_n = f_n + \epsilon_n$$

- Assume a zero-mean Gaussian error: $\epsilon_n \sim \mathcal{N}(\epsilon_n | 0, \sigma^2)$

Gaussian Process Regression

- Training data: $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$. $\boldsymbol{x}_n \in \mathbb{R}^D$, $y_n \in \mathbb{R}$
- Assume the responses to be a noisy function of the inputs

$$y_n = f(\boldsymbol{x}_n) + \epsilon_n = f_n + \epsilon_n$$

- Assume a zero-mean Gaussian error: $\epsilon_n \sim \mathcal{N}(\epsilon_n|0, \sigma^2)$
- Thus the likelihood model

$$p(y_n|f_n) = \mathcal{N}(y_n|f_n, \sigma^2)$$

Gaussian Process Regression

- Training data: $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$. $\boldsymbol{x}_n \in \mathbb{R}^D$, $y_n \in \mathbb{R}$
- Assume the responses to be a noisy function of the inputs

$$y_n = f(\boldsymbol{x}_n) + \epsilon_n = f_n + \epsilon_n$$

- Assume a zero-mean Gaussian error: $\epsilon_n \sim \mathcal{N}(\epsilon_n | 0, \sigma^2)$
- Thus the likelihood model

$$p(y_n | f_n) = \mathcal{N}(y_n | f_n, \sigma^2)$$

- For N i.i.d. responses, the joint likelihood can be written as

$$p(\boldsymbol{y} | \mathbf{f}) = \mathcal{N}(\boldsymbol{y} | \mathbf{f}, \sigma^2 \mathbf{I}_N)$$

Gaussian Process Regression

- Training data: $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$. $\boldsymbol{x}_n \in \mathbb{R}^D$, $y_n \in \mathbb{R}$
- Assume the responses to be a noisy function of the inputs

$$y_n = f(\boldsymbol{x}_n) + \epsilon_n = f_n + \epsilon_n$$

- Assume a zero-mean Gaussian error: $\epsilon_n \sim \mathcal{N}(\epsilon_n|0, \sigma^2)$
- Thus the likelihood model

$$p(y_n|f_n) = \mathcal{N}(y_n|f_n, \sigma^2)$$

- For N i.i.d. responses, the joint likelihood can be written as

$$p(\boldsymbol{y}|\mathbf{f}) = \mathcal{N}(\boldsymbol{y}|\mathbf{f}, \sigma^2\mathbf{I}_N)$$

- We will assume a zero mean Gaussian Process prior on f, which means:

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$

Gaussian Process Regression

- The likelihood model

$$p(\boldsymbol{y}|\mathbf{f}) = \mathcal{N}(\boldsymbol{y}|\mathbf{f}, \sigma^2\mathbf{I}_N)$$

- The prior distribution

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$

Gaussian Process Regression

- The likelihood model
$$p(\boldsymbol{y}|\mathbf{f}) = \mathcal{N}(\boldsymbol{y}|\mathbf{f}, \sigma^2 \mathbf{I}_N)$$
- The prior distribution
$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$
- Note: We don't actually need to compute the posterior $p(\mathbf{f}|\boldsymbol{y})$ here

Gaussian Process Regression

- The likelihood model

$$p(\boldsymbol{y}|\mathbf{f}) = \mathcal{N}(\boldsymbol{y}|\mathbf{f}, \sigma^2\mathbf{I}_N)$$

- The prior distribution

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$

- Note: We don't actually need to compute the posterior $p(\mathbf{f}|\boldsymbol{y})$ here
- The marginal distribution of the training data responses $\boldsymbol{y}$

$$p(\boldsymbol{y}) = \int p(\boldsymbol{y}|\mathbf{f})p(\mathbf{f})d\mathbf{f} = \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \mathbf{K} + \sigma^2\mathbf{I}_N) = \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \mathbf{C}_N)$$

Gaussian Process Regression

- The likelihood model

$$p(\boldsymbol{y}|\mathbf{f}) = \mathcal{N}(\boldsymbol{y}|\mathbf{f}, \sigma^2\mathbf{I}_N)$$

- The prior distribution

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$

- Note: We don't actually need to compute the posterior $p(\mathbf{f}|\boldsymbol{y})$ here
- The marginal distribution of the training data responses $\boldsymbol{y}$

$$p(\boldsymbol{y}) = \int p(\boldsymbol{y}|\mathbf{f})p(\mathbf{f})d\mathbf{f} = \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \mathbf{K} + \sigma^2\mathbf{I}_N) = \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \mathbf{C}_N)$$

- What will be the prediction y_* for a new test example $\boldsymbol{x}_*$?

Gaussian Process Regression

- The likelihood model

$$p(\boldsymbol{y}|\mathbf{f}) = \mathcal{N}(\boldsymbol{y}|\mathbf{f}, \sigma^2\mathbf{I}_N)$$

- The prior distribution

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$

- Note: We don't actually need to compute the posterior $p(\mathbf{f}|\boldsymbol{y})$ here
- The marginal distribution of the training data responses $\boldsymbol{y}$

$$p(\boldsymbol{y}) = \int p(\boldsymbol{y}|\mathbf{f})p(\mathbf{f})d\mathbf{f} = \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \mathbf{K} + \sigma^2\mathbf{I}_N) = \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \mathbf{C}_N)$$

- What will be the prediction y_* for a new test example $\boldsymbol{x}_*$?
- Well, we know that the marginal distribution of y_* will be

$$p(y_*) = \mathcal{N}(y_*|0, \kappa(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma^2)$$

Gaussian Process Regression

- The likelihood model

$$p(\boldsymbol{y}|\mathbf{f}) = \mathcal{N}(\boldsymbol{y}|\mathbf{f}, \sigma^2\mathbf{I}_N)$$

- The prior distribution

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$

- Note: We don't actually need to compute the posterior $p(\mathbf{f}|\boldsymbol{y})$ here
- The marginal distribution of the training data responses $\boldsymbol{y}$

$$p(\boldsymbol{y}) = \int p(\boldsymbol{y}|\mathbf{f})p(\mathbf{f})d\mathbf{f} = \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \mathbf{K} + \sigma^2\mathbf{I}_N) = \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \mathbf{C}_N)$$

- What will be the prediction y_* for a new test example $\boldsymbol{x}_*$?
- Well, we know that the marginal distribution of y_* will be

$$p(y_*) = \mathcal{N}(y_*|0, \kappa(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma^2)$$

- But what we actually want is the predictive distribution $p(y_*|\boldsymbol{y})$

Making Predictions

- Let's consider the joint distr. of N training responses $\boldsymbol{y}$ and test response y_*

$$p\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix} \middle| \begin{bmatrix} \boldsymbol{0} \\ 0 \end{bmatrix}, \mathbf{C}_{N+1}\right)$$

where the $(N+1) \times (N+1)$ matrix $\mathbf{C}_{N+1}$ is given by

$$\mathbf{C}_{N+1} = \begin{bmatrix} \mathbf{C}_N & \mathbf{k}_* \\ \mathbf{k}_*^\top & c \end{bmatrix}$$

Making Predictions

- Let's consider the joint distr. of N training responses $\boldsymbol{y}$ and test response y_*

$$p\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \mathbf{C}_{N+1}\right)$$

  where the $(N+1) \times (N+1)$ matrix $\mathbf{C}_{N+1}$ is given by

$$\mathbf{C}_{N+1} = \begin{bmatrix} \mathbf{C}_N & \mathbf{k}_* \\ \mathbf{k}_*^\top & c \end{bmatrix}$$

  and $\mathbf{k}_* = [k(\boldsymbol{x}_*, \boldsymbol{x}_1), \ldots, k(\boldsymbol{x}_*, \boldsymbol{x}_N)]^\top$, $c = k(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma^2$

Making Predictions

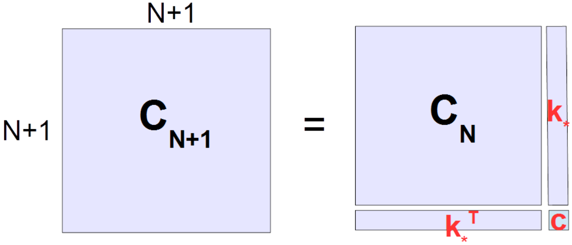
- Let's consider the joint distr. of N training responses $\boldsymbol{y}$ and test response y_*

$$p\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \mathbf{C}_{N+1}\right)$$

where the $(N+1) \times (N+1)$ matrix $\mathbf{C}_{N+1}$ is given by

$$\mathbf{C}_{N+1} = \begin{bmatrix} \mathbf{C}_N & \mathbf{k}_* \\ \mathbf{k}_*^\top & c \end{bmatrix}$$

and $\mathbf{k}_* = [k(\boldsymbol{x}_*, \boldsymbol{x}_1), \ldots, k(\boldsymbol{x}_*, \boldsymbol{x}_N)]^\top$, $c = k(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma^2$



Making Predictions

- Given the jointly Gaussian distribution

$$p\left(\begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{C}_N & \mathbf{k}_* \\ \mathbf{k}_*^\top & c \end{bmatrix}\right)$$

- The predictive distribution will be

$$\begin{aligned} p(y_*|\mathbf{y}) &= \mathcal{N}(y_*|\mu_*, \sigma_*^2) \\ \mu_* &= \mathbf{k}_*^\top \mathbf{C}_N^{-1} \mathbf{y} \\ \sigma_*^2 &= k(\mathbf{x}_*, \mathbf{x}_*) + \sigma^2 - \mathbf{k}_*^\top \mathbf{C}_N^{-1} \mathbf{k}_* \end{aligned}$$

- Follows readily from property of Gaussians (lecture 2 and PRML 2.94-2.96)

Making Predictions

- Given the jointly Gaussian distribution

$$p\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{C}_N & \mathbf{k}_* \\ \mathbf{k}_*^\top & c \end{bmatrix}\right)$$

- The predictive distribution will be

$$\begin{aligned} p(y_*|\boldsymbol{y}) &= \mathcal{N}(y_*|\mu_*, \sigma_*^2) \\ \mu_* &= \mathbf{k}_*^\top \mathbf{C}_N^{-1} \boldsymbol{y} \\ \sigma_*^2 &= k(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma^2 - \mathbf{k}_*^\top \mathbf{C}_N^{-1} \mathbf{k}_* \end{aligned}$$

- Follows readily from property of Gaussians (lecture 2 and PRML 2.94-2.96)

- Note: Instead of explicitly inverting, often Cholesky decomposition $\mathbf{C}_N = \mathbf{L}\mathbf{L}^\top$ is used (for better numerical stability)

Making Predictions

- Given the jointly Gaussian distribution

$$p\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{C}_N & \mathbf{k}_* \\ \mathbf{k}_*{}^\top & c \end{bmatrix}\right)$$

- The predictive distribution will be

$$\begin{aligned} p(y_*|\boldsymbol{y}) &= \mathcal{N}(y_*|\mu_*, \sigma_*^2) \\ \mu_* &= \mathbf{k}_*{}^\top \mathbf{C}_N^{-1} \boldsymbol{y} \\ \sigma_*^2 &= k(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma^2 - \mathbf{k}_*{}^\top \mathbf{C}_N^{-1} \mathbf{k}_* \end{aligned}$$

- Follows readily from property of Gaussians (lecture 2 and PRML 2.94-2.96)
- Note: Instead of explicitly inverting, often Cholesky decomposition $\mathbf{C}_N = \mathbf{L}\mathbf{L}^\top$ is used (for better numerical stability)
- Test time cost

Making Predictions

- Given the jointly Gaussian distribution

$$p\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{C}_N & \mathbf{k}_* \\ \mathbf{k}_*^\top & c \end{bmatrix}\right)$$

- The predictive distribution will be

$$\begin{aligned} p(y_*|\boldsymbol{y}) &= \mathcal{N}(y_*|\mu_*, \sigma_*^2) \\ \mu_* &= \mathbf{k}_*^\top \mathbf{C}_N^{-1} \boldsymbol{y} \\ \sigma_*^2 &= k(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma^2 - \mathbf{k}_*^\top \mathbf{C}_N^{-1} \mathbf{k}_* \end{aligned}$$

- Follows readily from property of Gaussians (lecture 2 and PRML 2.94-2.96)
- Note: Instead of explicitly inverting, often Cholesky decomposition $\mathbf{C}_N = \mathbf{L}\mathbf{L}^\top$ is used (for better numerical stability)
- Test time cost is $\mathcal{O}(N)$

Making Predictions

- Given the jointly Gaussian distribution

$$p\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{C}_N & \mathbf{k}_* \\ \mathbf{k}_*^\top & c \end{bmatrix}\right)$$

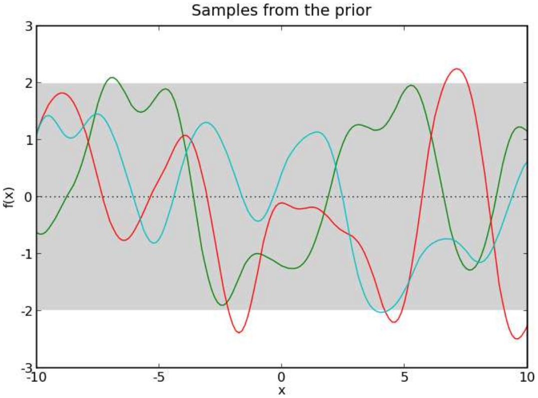
- The predictive distribution will be

$$\begin{aligned} p(y_*|\boldsymbol{y}) &= \mathcal{N}(y_*|\mu_*, \sigma_*^2) \\ \mu_* &= \mathbf{k}_*^\top \mathbf{C}_N^{-1} \boldsymbol{y} \\ \sigma_*^2 &= k(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma^2 - \mathbf{k}_*^\top \mathbf{C}_N^{-1} \mathbf{k}_* \end{aligned}$$

- Follows readily from property of Gaussians (lecture 2 and PRML 2.94-2.96)
- Note: Instead of explicitly inverting, often Cholesky decomposition $\mathbf{C}_N = \mathbf{L}\mathbf{L}^\top$ is used (for better numerical stability)
- Test time cost is $\mathcal{O}(N)$: linear in the number of training examples (just like kernel SVM or nearest neighbor methods)

GP Regression: Pictorially

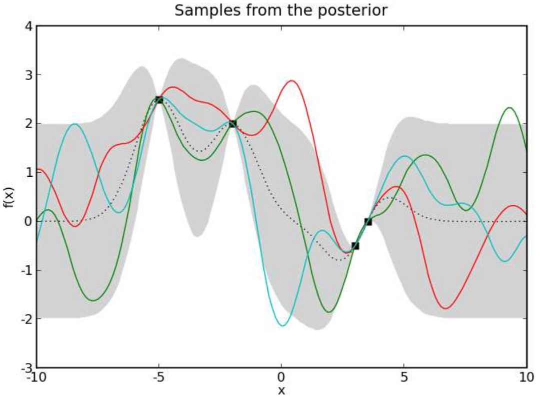
A GP with squared-exponential kernel function



Picture courtesy: https://pythonhosted.org/infpy/gps.html

GP Regression: Pictorially

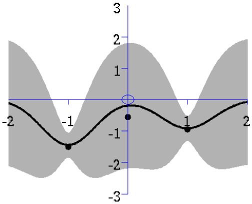
A GP with squared-exponential kernel function



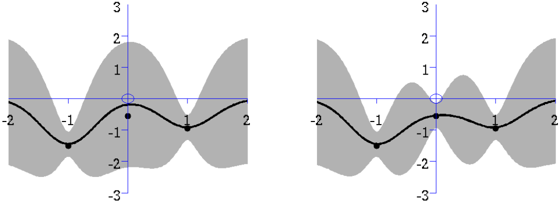
Shaded area denotes twice the standard deviation at each input

Picture courtesy: https://pythonhosted.org/infpy/gps.html

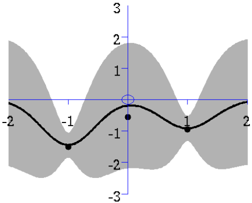
GP Regression: Pictorially

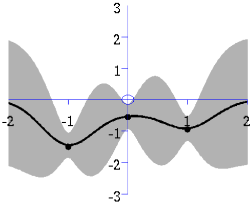


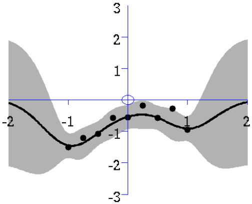
GP Regression: Pictorially



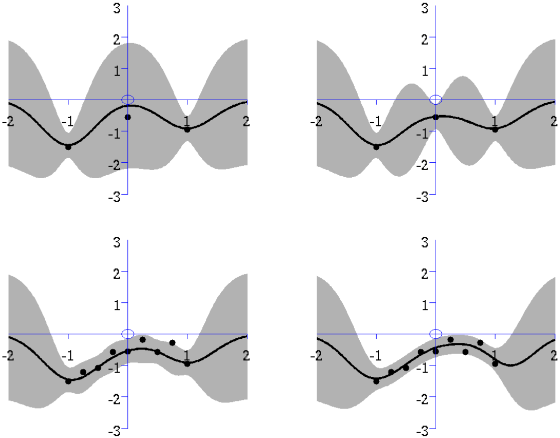
GP Regression: Pictorially







GP Regression: Pictorially



Interpreting GP predictions..

- Let's look at the predictions made by GP regression

$$
\begin{aligned}
p(y_*|\boldsymbol{y}) &= \mathcal{N}(y_*|\mu_*, \sigma_*^2) \\
\mu_* &= \mathbf{k}_*{}^\top \mathbf{C}_N^{-1}\boldsymbol{y} \\
\sigma_*^2 &= k(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma^2 - \mathbf{k}_*{}^\top \mathbf{C}_N^{-1}\mathbf{k}_*
\end{aligned}
$$

Interpreting GP predictions..

- Let's look at the predictions made by GP regression

$$
\begin{aligned}
p(y_*|\boldsymbol{y}) &= \mathcal{N}(y_*|\mu_*, \sigma_*^2) \\
\mu_* &= \mathbf{k}_*^\top \mathbf{C}_N^{-1} \boldsymbol{y} \\
\sigma_*^2 &= k(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma^2 - \mathbf{k}_*^\top \mathbf{C}_N^{-1} \mathbf{k}_*
\end{aligned}
$$

- Two interpretations for the mean prediction μ_*

Interpreting GP predictions..

- Let's look at the predictions made by GP regression

$$\begin{aligned} p(y_*|\boldsymbol{y}) &= \mathcal{N}(y_*|\mu_*, \sigma_*^2) \\ \mu_* &= \mathbf{k}_*{}^\top \mathbf{C}_N^{-1} \boldsymbol{y} \\ \sigma_*^2 &= k(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma^2 - \mathbf{k}_*{}^\top \mathbf{C}_N^{-1} \mathbf{k}_* \end{aligned}$$

- Two interpretations for the mean prediction μ_*
  - An SVM like interpretation

$$\mu_* = \mathbf{k}_*{}^\top \mathbf{C}_N^{-1} \boldsymbol{y} = \mathbf{k}_*{}^\top \boldsymbol{\alpha} = \sum_{n=1}^{N} k(\boldsymbol{x}_*, \boldsymbol{x}_n)\alpha_n$$

  where $\boldsymbol{\alpha}$ is akin to the weights of support vectors

Interpreting GP predictions..

- Let's look at the predictions made by GP regression

$$
\begin{aligned}
p(y_*|\boldsymbol{y}) &= \mathcal{N}(y_*|\mu_*, \sigma_*^2) \\
\mu_* &= \mathbf{k}_*{}^\top \mathbf{C}_N^{-1}\boldsymbol{y} \\
\sigma_*^2 &= k(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma^2 - \mathbf{k}_*{}^\top \mathbf{C}_N^{-1}\mathbf{k}_*
\end{aligned}
$$

- Two interpretations for the mean prediction μ_*
  - An SVM like interpretation

$$
\mu_* = \mathbf{k}_*{}^\top \mathbf{C}_N^{-1}\boldsymbol{y} = \mathbf{k}_*{}^\top \boldsymbol{\alpha} = \sum_{n=1}^{N} k(\boldsymbol{x}_*, \boldsymbol{x}_n)\alpha_n
$$

    where $\boldsymbol{\alpha}$ is akin to the weights of support vectors

  - A nearest neighbors interpretation

$$
\mu_* = \mathbf{k}_*{}^\top \mathbf{C}_N^{-1}\boldsymbol{y} = \boldsymbol{w}^\top \boldsymbol{y} = \sum_{n=1}^{N} w_n y_n
$$

    where $\boldsymbol{w}$ is akin to the weights of the neighbors

Inferring Hyperparameters

- There are two hyperparameters in GP regression models
  - Variance of the Gaussian noise σ^2
  - Hyperparameters θ of the covariance function κ, e.g.,

$$\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\frac{||\boldsymbol{x}_n - \boldsymbol{x}_m||^2}{\gamma}\right) \qquad \text{(RBF kernel)}$$

$$\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\sum_{d=1}^{D}\frac{(\boldsymbol{x}_{nd} - \boldsymbol{x}_{md})^2}{\gamma_d}\right) \qquad \text{(ARD kernel)}$$

Inferring Hyperparameters

- There are two hyperparameters in GP regression models
  - Variance of the Gaussian noise σ^2
  - Hyperparameters θ of the covariance function κ, e.g.,

$$\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\frac{||\boldsymbol{x}_n - \boldsymbol{x}_m||^2}{\gamma}\right) \qquad \text{(RBF kernel)}$$

$$\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\sum_{d=1}^{D}\frac{(x_{nd} - x_{md})^2}{\gamma_d}\right) \qquad \text{(ARD kernel)}$$

- These can be learned from data by maximizing the marginal likelihood

$$p(\boldsymbol{y}|\sigma^2, \theta) = \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \sigma^2\mathbf{I}_N + \mathbf{K}_\theta)$$

Inferring Hyperparameters

- There are two hyperparameters in GP regression models
  - Variance of the Gaussian noise σ^2
  - Hyperparameters θ of the covariance function κ, e.g.,

$$\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\frac{||\boldsymbol{x}_n - \boldsymbol{x}_m||^2}{\gamma}\right) \qquad \text{(RBF kernel)}$$

$$\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\sum_{d=1}^{D} \frac{(\boldsymbol{x}_{nd} - \boldsymbol{x}_{md})^2}{\gamma_d}\right) \qquad \text{(ARD kernel)}$$

- These can be learned from data by maximizing the marginal likelihood

$$p(\boldsymbol{y}|\sigma^2, \theta) = \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \sigma^2\mathbf{I}_N + \mathbf{K}_\theta)$$

- Can maximize the (log) marginal likelihood w.r.t. σ^2 and the kernel hyperparameters θ and get point estimates of the hyperparameters

$$\log p(\boldsymbol{y}|\sigma^2, \theta) = -\frac{1}{2}\log|\sigma^2\mathbf{I}_N + \mathbf{K}_\theta| - \frac{1}{2}\boldsymbol{y}^\top(\sigma^2\mathbf{I}_N + \mathbf{K}_\theta)^{-1}\boldsymbol{y} + \text{const}$$

Inferring Hyperparameters

- The (log) marginal likelihood

$$\log p(\boldsymbol{y}|\sigma^2, \theta) = -\frac{1}{2}\log|\sigma^2\mathbf{I}_N + \mathbf{K}_\theta| - \frac{1}{2}\boldsymbol{y}^\top(\sigma^2\mathbf{I}_N + \mathbf{K}_\theta)^{-1}\boldsymbol{y} + \text{const}$$

Inferring Hyperparameters

- The (log) marginal likelihood

$$\log p(\boldsymbol{y}|\sigma^2, \theta) = -\frac{1}{2}\log|\sigma^2\mathbf{I}_N + \mathbf{K}_\theta| - \frac{1}{2}\boldsymbol{y}^\top(\sigma^2\mathbf{I}_N + \mathbf{K}_\theta)^{-1}\boldsymbol{y} + \text{const}$$

- Defining $\mathbf{K}_y = \sigma^2\mathbf{I}_N + \mathbf{K}_\theta$ and taking derivative w.r.t. kernel hyperparams θ

$$\begin{aligned}
\frac{\partial}{\partial\theta_j}\log p(\boldsymbol{y}|\sigma^2, \theta) &= -\frac{1}{2}\text{tr}\left(\mathbf{K}_y^{-1}\frac{\partial\mathbf{K}_y}{\partial\theta_j}\right) + \frac{1}{2}\boldsymbol{y}^\top\mathbf{K}_y^{-1}\frac{\partial\mathbf{K}_y}{\partial\theta_j}\mathbf{K}_y^{-1}\boldsymbol{y} \\
&= \frac{1}{2}\text{tr}\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \mathbf{K}_y^{-1})\frac{\partial\mathbf{K}_y}{\partial\theta_j}\right)
\end{aligned}$$

where θ_j is the j^{th} hyperparam. of the kernel, and $\boldsymbol{\alpha} = \mathbf{K}_y^{-1}\boldsymbol{y}$

Inferring Hyperparameters

- The (log) marginal likelihood

$$\log p(\boldsymbol{y}|\sigma^2, \theta) = -\frac{1}{2}\log|\sigma^2\mathbf{I}_N + \mathbf{K}_\theta| - \frac{1}{2}\boldsymbol{y}^\top(\sigma^2\mathbf{I}_N + \mathbf{K}_\theta)^{-1}\boldsymbol{y} + \text{const}$$

- Defining $\mathbf{K}_y = \sigma^2\mathbf{I}_N + \mathbf{K}_\theta$ and taking derivative w.r.t. kernel hyperparams θ

$$\begin{aligned}\frac{\partial}{\partial\theta_j}\log p(\boldsymbol{y}|\sigma^2, \theta) &= -\frac{1}{2}\text{tr}\left(\mathbf{K}_y^{-1}\frac{\partial\mathbf{K}_y}{\partial\theta_j}\right) + \frac{1}{2}\boldsymbol{y}^\top\mathbf{K}_y^{-1}\frac{\partial\mathbf{K}_y}{\partial\theta_j}\mathbf{K}_y^{-1}\boldsymbol{y} \\ &= \frac{1}{2}\text{tr}\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \mathbf{K}_y^{-1})\frac{\partial\mathbf{K}_y}{\partial\theta_j}\right)\end{aligned}$$

  where θ_j is the j^{th} hyperparam. of the kernel, and $\boldsymbol{\alpha} = \mathbf{K}_y^{-1}\boldsymbol{y}$

- No closed form solution for θ_j. Gradient based methods can be used.

Inferring Hyperparameters

- The (log) marginal likelihood

$$\log p(\boldsymbol{y}|\sigma^2, \theta) = -\frac{1}{2}\log|\sigma^2\mathbf{I}_N + \mathbf{K}_\theta| - \frac{1}{2}\boldsymbol{y}^\top(\sigma^2\mathbf{I}_N + \mathbf{K}_\theta)^{-1}\boldsymbol{y} + \text{const}$$

- Defining $\mathbf{K}_y = \sigma^2\mathbf{I}_N + \mathbf{K}_\theta$ and taking derivative w.r.t. kernel hyperparams θ

$$\begin{aligned}\frac{\partial}{\partial\theta_j}\log p(\boldsymbol{y}|\sigma^2, \theta) &= -\frac{1}{2}\text{tr}\left(\mathbf{K}_y^{-1}\frac{\partial\mathbf{K}_y}{\partial\theta_j}\right) + \frac{1}{2}\boldsymbol{y}^\top\mathbf{K}_y^{-1}\frac{\partial\mathbf{K}_y}{\partial\theta_j}\mathbf{K}_y^{-1}\boldsymbol{y} \\ &= \frac{1}{2}\text{tr}\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \mathbf{K}_y^{-1})\frac{\partial\mathbf{K}_y}{\partial\theta_j}\right)\end{aligned}$$

  where θ_j is the j^{th} hyperparam. of the kernel, and $\boldsymbol{\alpha} = \mathbf{K}_y^{-1}\boldsymbol{y}$
- No closed form solution for θ_j. Gradient based methods can be used.
- Note: Computing $\mathbf{K}_y^{-1}$ itself takes $\mathcal{O}(N^3)$ time (faster approximations exist though). Then each gradient computation takes $\mathcal{O}(N^2)$ time

Inferring Hyperparameters

- The (log) marginal likelihood

$$\log p(\boldsymbol{y}|\sigma^2, \theta) = -\frac{1}{2}\log|\sigma^2\mathbf{I}_N + \mathbf{K}_\theta| - \frac{1}{2}\boldsymbol{y}^\top(\sigma^2\mathbf{I}_N + \mathbf{K}_\theta)^{-1}\boldsymbol{y} + \text{const}$$

- Defining $\mathbf{K}_y = \sigma^2\mathbf{I}_N + \mathbf{K}_\theta$ and taking derivative w.r.t. kernel hyperparams θ

$$\begin{aligned}\frac{\partial}{\partial\theta_j}\log p(\boldsymbol{y}|\sigma^2, \theta) &= -\frac{1}{2}\text{tr}\left(\mathbf{K}_y^{-1}\frac{\partial\mathbf{K}_y}{\partial\theta_j}\right) + \frac{1}{2}\boldsymbol{y}^\top\mathbf{K}_y^{-1}\frac{\partial\mathbf{K}_y}{\partial\theta_j}\mathbf{K}_y^{-1}\boldsymbol{y} \\ &= \frac{1}{2}\text{tr}\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \mathbf{K}_y^{-1})\frac{\partial\mathbf{K}_y}{\partial\theta_j}\right)\end{aligned}$$

  where θ_j is the j^{th} hyperparam. of the kernel, and $\boldsymbol{\alpha} = \mathbf{K}_y^{-1}\boldsymbol{y}$
- No closed form solution for θ_j. Gradient based methods can be used.
- Note: Computing $\mathbf{K}_y^{-1}$ itself takes $\mathcal{O}(N^3)$ time (faster approximations exist though). Then each gradient computation takes $\mathcal{O}(N^2)$ time
- Form of $\frac{\partial\mathbf{K}_y}{\partial\theta_j}$ depends on the covariance/kernel function κ

Inferring Hyperparameters

- The (log) marginal likelihood

$$\log p(\boldsymbol{y}|\sigma^2, \theta) = -\frac{1}{2}\log|\sigma^2\mathbf{I}_N + \mathbf{K}_\theta| - \frac{1}{2}\boldsymbol{y}^\top(\sigma^2\mathbf{I}_N + \mathbf{K}_\theta)^{-1}\boldsymbol{y} + \text{const}$$

- Defining $\mathbf{K}_y = \sigma^2\mathbf{I}_N + \mathbf{K}_\theta$ and taking derivative w.r.t. kernel hyperparams θ

$$\begin{aligned}
\frac{\partial}{\partial\theta_j}\log p(\boldsymbol{y}|\sigma^2, \theta) &= -\frac{1}{2}\text{tr}\left(\mathbf{K}_y^{-1}\frac{\partial\mathbf{K}_y}{\partial\theta_j}\right) + \frac{1}{2}\boldsymbol{y}^\top\mathbf{K}_y^{-1}\frac{\partial\mathbf{K}_y}{\partial\theta_j}\mathbf{K}_y^{-1}\boldsymbol{y} \\
&= \frac{1}{2}\text{tr}\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \mathbf{K}_y^{-1})\frac{\partial\mathbf{K}_y}{\partial\theta_j}\right)
\end{aligned}$$

  where θ_j is the j^{th} hyperparam. of the kernel, and $\boldsymbol{\alpha} = \mathbf{K}_y^{-1}\boldsymbol{y}$

- No closed form solution for θ_j. Gradient based methods can be used.
- Note: Computing $\mathbf{K}_y^{-1}$ itself takes $\mathcal{O}(N^3)$ time (faster approximations exist though). Then each gradient computation takes $\mathcal{O}(N^2)$ time
- Form of $\frac{\partial\mathbf{K}_y}{\partial\theta_j}$ depends on the covariance/kernel function κ
- Noise variance σ^2 can also be estimated likewise

Gaussian Processes with GLMs

- GP regression is only one example of supervised learning with GP

Gaussian Processes with GLMs

- GP regression is only one example of supervised learning with GP
- GP can be combined with other types of likelihood functions to handle other types of responses (e.g., binary, categorical, counts, etc.) by replacing the Gaussian likelihood for responses by a generalized linear model

Gaussian Processes with GLMs

- GP regression is only one example of supervised learning with GP
- GP can be combined with other types of likelihood functions to handle other types of responses (e.g., binary, categorical, counts, etc.) by replacing the Gaussian likelihood for responses by a generalized linear model
- Inference however becomes more tricky because such likelihoods may no longer be conjugate to GP prior. Approximate inference needed in such cases.

Gaussian Processes with GLMs

- GP regression is only one example of supervised learning with GP
- GP can be combined with other types of likelihood functions to handle other types of responses (e.g., binary, categorical, counts, etc.) by replacing the Gaussian likelihood for responses by a generalized linear model
- Inference however becomes more tricky because such likelihoods may no longer be conjugate to GP prior. Approximate inference needed in such cases.
- We will revisit one such example (GP for binary classification) later during the semester

GP vs (Kernel) SVM

- The objective function of a soft-margin SVM looks like

$$\frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{n=1}^{N}(1 - y_n f_n)_+$$

where $f_n = \boldsymbol{w}^\top \boldsymbol{x}_n$ and y_n is the true label for $\boldsymbol{x}_n$

GP vs (Kernel) SVM

- The objective function of a soft-margin SVM looks like

$$\frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{n=1}^{N}(1 - y_n f_n)_+$$

  where $f_n = \boldsymbol{w}^\top \boldsymbol{x}_n$ and y_n is the true label for $\boldsymbol{x}_n$

- Kernel SVM: $f_n = \sum_{m=1}^{N} \alpha_m k(\boldsymbol{x}_n, \boldsymbol{x}_m)$. Denote $\mathbf{f} = [f_1, \ldots, f_N]^\top$

GP vs (Kernel) SVM

- The objective function of a soft-margin SVM looks like

$$\frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{n=1}^{N}(1 - y_n f_n)_+$$

where $f_n = \boldsymbol{w}^\top \boldsymbol{x}_n$ and y_n is the true label for $\boldsymbol{x}_n$

- Kernel SVM: $f_n = \sum_{m=1}^{N} \alpha_m k(\boldsymbol{x}_n, \boldsymbol{x}_m)$. Denote $\mathbf{f} = [f_1, \ldots, f_N]^\top$
- We can write $\frac{||\boldsymbol{w}||^2}{2} = \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha} = \mathbf{f}^\top \mathbf{K}^{-1} \mathbf{f}$, and kernel SVM objective becomes

$$\frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f} + C\sum_{n=1}^{N}(1 - y_n f_n)_+$$

GP vs (Kernel) SVM

- The objective function of a soft-margin SVM looks like

$$\frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{n=1}^{N}(1 - y_n f_n)_+$$

where $f_n = \boldsymbol{w}^\top \boldsymbol{x}_n$ and y_n is the true label for $\boldsymbol{x}_n$

- Kernel SVM: $f_n = \sum_{m=1}^{N} \alpha_m k(\boldsymbol{x}_n, \boldsymbol{x}_m)$. Denote $\mathbf{f} = [f_1, \dots, f_N]^\top$
- We can write $\frac{||\boldsymbol{w}||^2}{2} = \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha} = \mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f}$, and kernel SVM objective becomes

$$\frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f} + C\sum_{n=1}^{N}(1 - y_n f_n)_+$$

- Negative log-posterior $\log p(\boldsymbol{y}|\mathbf{f})p(\mathbf{f})$ of a GP can be written as

$$\frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f} - \sum_{n=1}^{N} \log p(y_n|f_n) + \text{const}$$

GP vs (Kernel) SVM

- Thus GPs can be interpreted as a Bayesian analogue of kernel SVMs

GP vs (Kernel) SVM

- Thus GPs can be interpreted as a Bayesian analogue of kernel SVMs
- Both GP and SVM need dealing with (storing/inverting) large kernel matrices
  - Various approximations proposed to address this issue (applicable to both)

GP vs (Kernel) SVM

- Thus GPs can be interpreted as a Bayesian analogue of kernel SVMs
- Both GP and SVM need dealing with (storing/inverting) large kernel matrices
  - Various approximations proposed to address this issue (applicable to both)
- Ability to learn the kernel hyperparameters in GP is very useful, e.g.,

GP vs (Kernel) SVM

- Thus GPs can be interpreted as a Bayesian analogue of kernel SVMs
- Both GP and SVM need dealing with (storing/inverting) large kernel matrices
  - Various approximations proposed to address this issue (applicable to both)
- Ability to learn the kernel hyperparameters in GP is very useful, e.g.,
  - Learning the kernel bandwidth for Gaussian kernels

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\frac{||\boldsymbol{x}_n - \boldsymbol{x}_m||^2}{\gamma}\right)$$

GP vs (Kernel) SVM

- Thus GPs can be interpreted as a Bayesian analogue of kernel SVMs
- Both GP and SVM need dealing with (storing/inverting) large kernel matrices
  - Various approximations proposed to address this issue (applicable to both)
- Ability to learn the kernel hyperparameters in GP is very useful, e.g.,
  - Learning the kernel bandwidth for Gaussian kernels

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\frac{||\boldsymbol{x}_n - \boldsymbol{x}_m||^2}{\gamma}\right)$$

  - Doing feature selection (via Automatic Relevance Determination)

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\sum_{d=1}^{D}\frac{(\boldsymbol{x}_{nd} - \boldsymbol{x}_{md})^2}{\gamma_d}\right)$$

GP vs (Kernel) SVM

- Thus GPs can be interpreted as a Bayesian analogue of kernel SVMs
- Both GP and SVM need dealing with (storing/inverting) large kernel matrices
  - Various approximations proposed to address this issue (applicable to both)
- Ability to learn the kernel hyperparameters in GP is very useful, e.g.,
  - Learning the kernel bandwidth for Gaussian kernels

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\frac{||\boldsymbol{x}_n - \boldsymbol{x}_m||^2}{\gamma}\right)$$

  - Doing feature selection (via Automatic Relevance Determination)

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\sum_{d=1}^{D}\frac{(\boldsymbol{x}_{nd} - \boldsymbol{x}_{md})^2}{\gamma_d}\right)$$

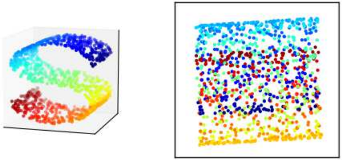
  - Learning compositions of kernels for more flexible modeling

$$\mathbf{K} = \mathbf{K}_{\theta_1} + \mathbf{K}_{\theta_2} + \ldots$$

Nonlinear Dimensionality Reduction using Gaussian Process (GPLVM)

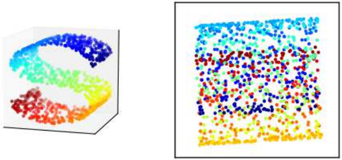
Why Nonlinear Dimensionality Reduction?

- Embeddings learned by PCA (left: original data, right: PCA)



Why Nonlinear Dimensionality Reduction?

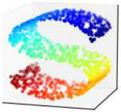
- Embeddings learned by PCA (left: original data, right: PCA)

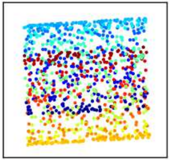


- Why PCA doesn't work in such cases?

Why Nonlinear Dimensionality Reduction?

- Embeddings learned by PCA (left: original data, right: PCA)

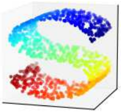


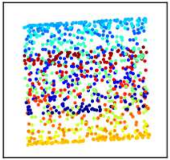


- Why PCA doesn't work in such cases?
    - Uses Euclidean distances; learns linear projections

Why Nonlinear Dimensionality Reduction?

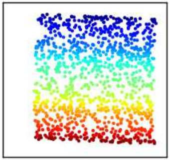
- Embeddings learned by PCA (left: original data, right: PCA)

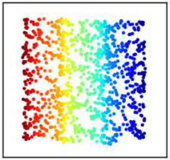




- Why PCA doesn't work in such cases?
  - Uses Euclidean distances; learns linear projections

- Embeddings learned by nonlinear dim. red. (left: LLE, right: ISOMAP)





Recap: Probabilistic PCA

- Given: $N \times D$ data matrix $\mathbf{X} = [\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top]^\top$, with $\boldsymbol{x}_n \in \mathbb{R}^D$
- Goal: Find a lower-dim. rep., an $N \times K$ matrix $\mathbf{Z} = [\boldsymbol{z}_1^\top, \ldots, \boldsymbol{z}_N^\top]^\top$, $\boldsymbol{z}_n \in \mathbb{R}^K$
- Assume the following generative model for each observation $\boldsymbol{x}_n$

$$\boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n \qquad \text{with } \mathbf{W} \in \mathbb{R}^{D \times K},\ \epsilon_n \sim \mathcal{N}(0, \sigma^2)$$

Recap: Probabilistic PCA

- Given: $N \times D$ data matrix $\mathbf{X} = [\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top]^\top$, with $\boldsymbol{x}_n \in \mathbb{R}^D$
- Goal: Find a lower-dim. rep., an $N \times K$ matrix $\mathbf{Z} = [\boldsymbol{z}_1^\top, \ldots, \boldsymbol{z}_N^\top]^\top$, $\boldsymbol{z}_n \in \mathbb{R}^K$
- Assume the following generative model for each observation $\boldsymbol{x}_n$

$$\boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n \qquad \text{with } \mathbf{W} \in \mathbb{R}^{D \times K},\ \epsilon_n \sim \mathcal{N}(0, \sigma^2)$$

- The conditional distribution

$$p(\boldsymbol{x}_n | \boldsymbol{z}_n, \mathbf{W}, \sigma^2) = \mathcal{N}(\mathbf{W}\boldsymbol{z}_n, \sigma^2 \mathbf{I}_D)$$

Recap: Probabilistic PCA

- Given: $N \times D$ data matrix $\mathbf{X} = [\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top]^\top$, with $\boldsymbol{x}_n \in \mathbb{R}^D$
- Goal: Find a lower-dim. rep., an $N \times K$ matrix $\mathbf{Z} = [\boldsymbol{z}_1^\top, \ldots, \boldsymbol{z}_N^\top]^\top$, $\boldsymbol{z}_n \in \mathbb{R}^K$
- Assume the following generative model for each observation $\boldsymbol{x}_n$

$$\boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n \qquad \text{with } \mathbf{W} \in \mathbb{R}^{D \times K},\ \epsilon_n \sim \mathcal{N}(0, \sigma^2)$$

- The conditional distribution

$$p(\boldsymbol{x}_n | \boldsymbol{z}_n, \mathbf{W}, \sigma^2) = \mathcal{N}(\mathbf{W}\boldsymbol{z}_n, \sigma^2 \mathbf{I}_D)$$

- Assume a Gaussian prior on $\boldsymbol{z}_n$: $p(\boldsymbol{z}_n) = \mathcal{N}(0, \mathbf{I}_K)$

Recap: Probabilistic PCA

- Given: $N \times D$ data matrix $\mathbf{X} = [\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top]^\top$, with $\boldsymbol{x}_n \in \mathbb{R}^D$
- Goal: Find a lower-dim. rep., an $N \times K$ matrix $\mathbf{Z} = [\boldsymbol{z}_1^\top, \ldots, \boldsymbol{z}_N^\top]^\top$, $\boldsymbol{z}_n \in \mathbb{R}^K$
- Assume the following generative model for each observation $\boldsymbol{x}_n$

$$\boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n \qquad \text{with } \mathbf{W} \in \mathbb{R}^{D \times K},\ \epsilon_n \sim \mathcal{N}(0, \sigma^2)$$

- The conditional distribution

$$p(\boldsymbol{x}_n|\boldsymbol{z}_n, \mathbf{W}, \sigma^2) = \mathcal{N}(\mathbf{W}\boldsymbol{z}_n, \sigma^2\mathbf{I}_D)$$

- Assume a Gaussian prior on $\boldsymbol{z}_n$: $p(\boldsymbol{z}_n) = \mathcal{N}(0, \mathbf{I}_K)$
- The marginal distribution of $\boldsymbol{x}_n$ (after integrating out latent variables $\boldsymbol{z}_n$)

$$\begin{aligned} p(\boldsymbol{x}_n|\mathbf{W}, \sigma^2) &= \mathcal{N}(\mathbf{0}, \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}_D) \\ p(\mathbf{X}|\mathbf{W}, \sigma^2) &= \prod_{n=1}^{N} p(\boldsymbol{x}_n|\mathbf{W}, \sigma^2) \end{aligned}$$

Gaussian Process Latent Variable Model (GPLVM)

- Consider the same model

$$\boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n \qquad \text{with } \mathbf{W} \in \mathbb{R}^{D\times K},\ \epsilon_n \sim \mathcal{N}(0, \sigma^2)$$

- Assume a prior $p(\mathbf{W}) = \prod_{d=1}^{D} \mathcal{N}(\boldsymbol{w}_d | 0, \mathbf{I}_K)$ where $\boldsymbol{w}_d$ is the d^{th} row of $\mathbf{W}$

Gaussian Process Latent Variable Model (GPLVM)

- Consider the same model

$$\boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n \qquad \text{with } \mathbf{W} \in \mathbb{R}^{D\times K},\ \epsilon_n \sim \mathcal{N}(0,\sigma^2)$$

- Assume a prior $p(\mathbf{W}) = \prod_{d=1}^{D} \mathcal{N}(\boldsymbol{w}_d|0,\mathbf{I}_K)$ where $\boldsymbol{w}_d$ is the d^{th} row of $\mathbf{W}$
- Suppose we integrate out $\mathbf{W}$ instead of $\boldsymbol{z}_n$ (treat $\boldsymbol{z}_n$'s as "parameter")

$$\begin{aligned} p(\mathbf{X}|\mathbf{Z},\sigma^2) &= \prod_{d=1}^{D} \mathcal{N}(\mathbf{X}_{:,d}|\mathbf{0},\mathbf{Z}\mathbf{Z}^\top + \sigma^2\mathbf{I}_D) \\ &= (2\pi)^{-DN/2}|\mathbf{K}_z|^{-D/2}\exp\left(-\frac{1}{2}\text{tr}(\mathbf{K}_z^{-1}\mathbf{X}\mathbf{X}^\top)\right) \end{aligned}$$

where $\mathbf{K}_z = \mathbf{Z}\mathbf{Z}^\top + \sigma^2\mathbf{I}$ and $\mathbf{X}_{:,d}$ is the d^{th} column of $N\times D$ data matrix $\mathbf{X}$

Gaussian Process Latent Variable Model (GPLVM)

- Consider the same model

$$\boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n \qquad \text{with } \mathbf{W} \in \mathbb{R}^{D\times K},\ \epsilon_n \sim \mathcal{N}(0, \sigma^2)$$

- Assume a prior $p(\mathbf{W}) = \prod_{d=1}^{D} \mathcal{N}(\boldsymbol{w}_d | 0, \mathbf{I}_K)$ where $\boldsymbol{w}_d$ is the d^{th} row of $\mathbf{W}$
- Suppose we integrate out $\mathbf{W}$ instead of $\boldsymbol{z}_n$ (treat $\boldsymbol{z}_n$'s as "parameter")

$$\begin{aligned} p(\mathbf{X}|\mathbf{Z}, \sigma^2) &= \prod_{d=1}^{D} \mathcal{N}(\mathbf{X}_{:,d} | \mathbf{0}, \mathbf{Z}\mathbf{Z}^\top + \sigma^2 \mathbf{I}_D) \\ &= (2\pi)^{-DN/2} |\mathbf{K}_z|^{-D/2} \exp\left(-\frac{1}{2}\text{tr}(\mathbf{K}_z^{-1}\mathbf{X}\mathbf{X}^\top)\right) \end{aligned}$$

  where $\mathbf{K}_z = \mathbf{Z}\mathbf{Z}^\top + \sigma^2\mathbf{I}$ and $\mathbf{X}_{:,d}$ is the d^{th} column of $N \times D$ data matrix $\mathbf{X}$
- Note that we can think of $\mathbf{X}_{:,d}$ modeled by a GP regression model

$$\mathbf{X}_{:,d} \sim \mathcal{N}(\mathbf{0}, \mathbf{Z}\mathbf{Z}^\top + \sigma^2\mathbf{I}_D)$$

- There are a total of D such GPs (one for each column of $\mathbf{X}$)

GPLVM

- $p(\mathbf{X}|\mathbf{Z},\sigma^2)$ is now a product of D GPs (one per column of data matrix $\mathbf{X}$)

$$
\begin{aligned}
p(\mathbf{X}|\mathbf{Z},\sigma^2) &= \prod_{d=1}^{D} \mathcal{N}(\mathbf{X}_{:,d}|\mathbf{0}, \mathbf{Z}\mathbf{Z}^\top + \sigma^2\mathbf{I}_D) \\
&= (2\pi)^{-DN/2}|\mathbf{K}_z|^{-D/2}\exp\left(-\frac{1}{2}\mathrm{tr}(\mathbf{K}_z^{-1}\mathbf{X}\mathbf{X}^\top)\right)
\end{aligned}
$$

- Using $\mathbf{K}_z = \mathbf{Z}\mathbf{Z}^\top + \sigma^2\mathbf{I}$ and doing MLE will give the same solution for $\mathbf{Z}$ as linear PCA (note that $\mathbf{Z}\mathbf{Z}^\top$ is a linear kernel over $\mathbf{Z}$, the low-dim rep of data)
- But with $\mathbf{K}_z = \mathbf{K} + \sigma^2\mathbf{I}$ (with $\mathbf{K}$ being some appropriately defined kernel matrix over $\mathbf{Z}$) will give nonlinear dimensionality reduction

MLE for GPLVM

- Log-likelihood is given by

$$\mathcal{L} = -\frac{D}{2}\log|\mathbf{K}_z| - \frac{1}{2}\mathrm{tr}(\mathbf{K}_z^{-1}\mathbf{X}\mathbf{X}^\top)$$

where $\mathbf{K}_z = \mathbf{K} + \sigma^2\mathbf{I}$ and $\mathbf{K}$ denotes the kernel matrix of our low-dim rep. $\mathbf{Z}$

MLE for GPLVM

- Log-likelihood is given by

$$\mathcal{L} = -\frac{D}{2}\log|\mathbf{K}_z| - \frac{1}{2}\text{tr}(\mathbf{K}_z^{-1}\mathbf{X}\mathbf{X}^\top)$$

  where $\mathbf{K}_z = \mathbf{K} + \sigma^2\mathbf{I}$ and $\mathbf{K}$ denotes the kernel matrix of our low-dim rep. $\mathbf{Z}$

- The goal is to estimate the $N \times K$ matrix $\mathbf{Z}$

MLE for GPLVM

- Log-likelihood is given by

$$\mathcal{L} = -\frac{D}{2}\log|\mathbf{K}_z| - \frac{1}{2}\mathrm{tr}(\mathbf{K}_z^{-1}\mathbf{X}\mathbf{X}^\top)$$

  where $\mathbf{K}_z = \mathbf{K} + \sigma^2\mathbf{I}$ and $\mathbf{K}$ denotes the kernel matrix of our low-dim rep. $\mathbf{Z}$

- The goal is to estimate the $N \times K$ matrix $\mathbf{Z}$

- Can't find closed form estimate of $\mathbf{Z}$. Need to use gradient-based methods, with the gradient given by

$$\frac{\partial \mathcal{L}}{\partial Z_{nk}} = \frac{\partial \mathcal{L}}{\partial \mathbf{K}_z}\frac{\partial \mathbf{K}_z}{\partial Z_{nk}}$$

  where $\frac{\partial \mathcal{L}}{\partial \mathbf{K}_z} = \mathbf{K}_z^{-1}\mathbf{X}\mathbf{X}^\top\mathbf{K}_z^{-1} - D\mathbf{K}_z^{-1}$ and $\frac{\partial \mathbf{K}_z}{\partial Z_{nk}}$ will depend on the kernel function used (note: hyperparameters of the kernel can also be learned just as we did it in the GP regression case)

MLE for GPLVM

- Log-likelihood is given by

$$\mathcal{L} = -\frac{D}{2}\log|\mathbf{K}_z| - \frac{1}{2}\text{tr}(\mathbf{K}_z^{-1}\mathbf{X}\mathbf{X}^\top)$$

  where $\mathbf{K}_z = \mathbf{K} + \sigma^2\mathbf{I}$ and $\mathbf{K}$ denotes the kernel matrix of our low-dim rep. $\mathbf{Z}$

- The goal is to estimate the $N \times K$ matrix $\mathbf{Z}$

- Can't find closed form estimate of $\mathbf{Z}$. Need to use gradient-based methods, with the gradient given by

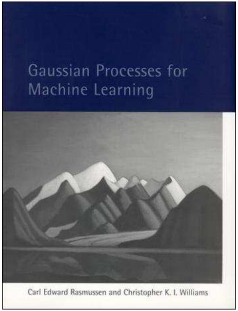
$$\frac{\partial \mathcal{L}}{\partial Z_{nk}} = \frac{\partial \mathcal{L}}{\partial \mathbf{K}_z}\frac{\partial \mathbf{K}_z}{\partial Z_{nk}}$$

  where $\frac{\partial \mathcal{L}}{\partial \mathbf{K}_z} = \mathbf{K}_z^{-1}\mathbf{X}\mathbf{X}^\top\mathbf{K}_z^{-1} - D\mathbf{K}_z^{-1}$ and $\frac{\partial \mathbf{K}_z}{\partial Z_{nk}}$ will depend on the kernel function used (note: hyperparameters of the kernel can also be learned just as we did it in the GP regression case)

- Can also impose a prior on $\mathbf{Z}$ and do MAP (or fully Bayesian) estimation

Resources on Gaussian Processes

- Book: Gaussian Processes for Machine Learning (freely available online)



- MATLAB Packages: Useful to play with, build applications, extend existing models and inference algorithms for GPs (both regression and classification)
  - GPML: `http://www.gaussianprocess.org/gpml/code/matlab/doc/`
  - GPStuff: `http://research.cs.aalto.fi/pml/software/gpstuff/`
  - GPLVM: `https://github.com/lawrennd/gplvm`