Learning Nonlinear Functions via Gaussian Processes (1)

Piyush Rai

Probabilistic Machine Learning (CS772A)

October 17, 2017

Linear Models

- Consider the problem of learning to map an input $\boldsymbol{x} \in \mathbb{R}^D$ to an output y

Linear Models

- Consider the problem of learning to map an input $\boldsymbol{x} \in \mathbb{R}^D$ to an output y
- Linear models use a weighted combination of input features (i.e., $\boldsymbol{w}^\top \boldsymbol{x}$) to generate y

Linear Models

- Consider the problem of learning to map an input $\boldsymbol{x} \in \mathbb{R}^D$ to an output y
- Linear models use a weighted combination of input features (i.e., $\boldsymbol{w}^\top \boldsymbol{x}$) to generate y

$$p(y|\boldsymbol{w}, \boldsymbol{x}) = \mathcal{N}(y|\boldsymbol{w}^\top \boldsymbol{x}, \beta^{-1}) \qquad \text{(Linear Regression)}$$

Linear Models

- Consider the problem of learning to map an input $\boldsymbol{x} \in \mathbb{R}^D$ to an output y
- Linear models use a weighted combination of input features (i.e., $\boldsymbol{w}^\top \boldsymbol{x}$) to generate y

$$
\begin{aligned}
p(y|\boldsymbol{w}, \boldsymbol{x}) &= \mathcal{N}(y|\boldsymbol{w}^\top \boldsymbol{x}, \beta^{-1}) && \text{(Linear Regression)} \\
p(y|\boldsymbol{w}, \boldsymbol{x}) &= [\sigma(\boldsymbol{w}^\top \boldsymbol{x})]^y [1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x})]^{1-y} && \text{(Logistic Regression)}
\end{aligned}
$$

Linear Models

- Consider the problem of learning to map an input $\boldsymbol{x} \in \mathbb{R}^D$ to an output y
- Linear models use a weighted combination of input features (i.e., $\boldsymbol{w}^\top \boldsymbol{x}$) to generate y

$$
\begin{aligned}
p(y|\boldsymbol{w}, \boldsymbol{x}) &= \mathcal{N}(y|\boldsymbol{w}^\top \boldsymbol{x}, \beta^{-1}) && \text{(Linear Regression)} \\
p(y|\boldsymbol{w}, \boldsymbol{x}) &= [\sigma(\boldsymbol{w}^\top \boldsymbol{x})]^y [1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x})]^{1-y} && \text{(Logistic Regression)}
\end{aligned}
$$

- This can also be extended to generalized linear models (to model other types of y)

$$
p(y|\boldsymbol{w}, \boldsymbol{x}) = \text{GLM}(y|\eta) \qquad \text{where } \eta = \boldsymbol{w}^\top \boldsymbol{x}
$$

Linear Models

- Consider the problem of learning to map an input $\boldsymbol{x} \in \mathbb{R}^D$ to an output y
- Linear models use a weighted combination of input features (i.e., $\boldsymbol{w}^\top \boldsymbol{x}$) to generate y

$$
\begin{aligned}
p(y|\boldsymbol{w}, \boldsymbol{x}) &= \mathcal{N}(y|\boldsymbol{w}^\top \boldsymbol{x}, \beta^{-1}) && \text{(Linear Regression)} \\
p(y|\boldsymbol{w}, \boldsymbol{x}) &= [\sigma(\boldsymbol{w}^\top \boldsymbol{x})]^y [1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x})]^{1-y} && \text{(Logistic Regression)}
\end{aligned}
$$

- This can also be extended to generalized linear models (to model other types of y)

$$
p(y|\boldsymbol{w}, \boldsymbol{x}) = \text{GLM}(y|\eta) \qquad \text{where } \eta = \boldsymbol{w}^\top \boldsymbol{x}
$$

- The weights $\boldsymbol{w}$ can be learned using MLE, MAP, or fully Bayesian inference

Linear Models

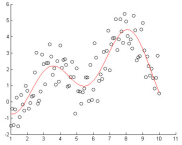
- Consider the problem of learning to map an input $\boldsymbol{x} \in \mathbb{R}^D$ to an output y
- Linear models use a weighted combination of input features (i.e., $\boldsymbol{w}^\top \boldsymbol{x}$) to generate y

$$
\begin{aligned}
p(y|\boldsymbol{w}, \boldsymbol{x}) &= \mathcal{N}(y|\boldsymbol{w}^\top \boldsymbol{x}, \beta^{-1}) && \text{(Linear Regression)} \\
p(y|\boldsymbol{w}, \boldsymbol{x}) &= [\sigma(\boldsymbol{w}^\top \boldsymbol{x})]^y [1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x})]^{1-y} && \text{(Logistic Regression)}
\end{aligned}
$$

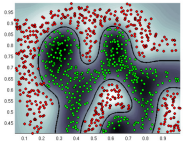
- This can also be extended to generalized linear models (to model other types of y)

$$p(y|\boldsymbol{w}, \boldsymbol{x}) = \text{GLM}(y|\eta) \qquad \text{where } \eta = \boldsymbol{w}^\top \boldsymbol{x}$$

- The weights $\boldsymbol{w}$ can be learned using MLE, MAP, or fully Bayesian inference
- However, linear models have limited expressive power. Fail to learn highly nonlinear patterns.



Nonlinear Regression



Nonlinear Classification

Linear Regression

- Let's first consider the (probabilistic) linear regression model

$$\begin{aligned} p(\boldsymbol{w}) &= \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu_0}, \Sigma_0) \qquad \text{(Prior)} \\ p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) &= \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}_n, \beta^{-1}) \qquad \text{(Likelihood w.r.t. one obs.)} \end{aligned}$$

Linear Regression

- Let's first consider the (probabilistic) linear regression model

$$
\begin{aligned}
p(\boldsymbol{w}) &= \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu_0}, \Sigma_0) && \text{(Prior)} \\
p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) &= \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}_n, \beta^{-1}) && \text{(Likelihood w.r.t. one obs.)} \\
p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) &= \mathcal{N}(\mathbf{X}\boldsymbol{w}, \beta^{-1}\mathbf{I}_N) && \text{(Likelihood w.r.t. } N \text{ obs.)}
\end{aligned}
$$

Linear Regression

- Let's first consider the (probabilistic) linear regression model

$$
\begin{aligned}
p(\boldsymbol{w}) &= \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu_0}, \Sigma_0) && \text{(Prior)} \\
p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) &= \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}_n, \beta^{-1}) && \text{(Likelihood w.r.t. one obs.)} \\
p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) &= \mathcal{N}(\mathbf{X}\boldsymbol{w}, \beta^{-1}\mathbf{I}_N) && \text{(Likelihood w.r.t. } N \text{ obs.)} \\
p(\boldsymbol{y}|\mathbf{X}) &= \int p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) p(\boldsymbol{w}) d\boldsymbol{w} = \mathcal{N}(\mathbf{X}\boldsymbol{\mu_0}, \beta^{-1}\mathbf{I}_N + \mathbf{X}\boldsymbol{\Sigma_0}\mathbf{X}^\top) && \text{(Marginal likelihood)}
\end{aligned}
$$

Linear Regression

- Let's first consider the (probabilistic) linear regression model

$$
\begin{aligned}
p(\boldsymbol{w}) &= \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu_0}, \Sigma_0) && \text{(Prior)} \\
p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) &= \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}_n, \beta^{-1}) && \text{(Likelihood w.r.t. one obs.)} \\
p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) &= \mathcal{N}(\mathbf{X}\boldsymbol{w}, \beta^{-1}\mathbf{I}_N) && \text{(Likelihood w.r.t. } N \text{ obs.)} \\
p(\boldsymbol{y}|\mathbf{X}) &= \int p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) p(\boldsymbol{w}) d\boldsymbol{w} = \mathcal{N}(\mathbf{X}\boldsymbol{\mu_0}, \beta^{-1}\mathbf{I}_N + \mathbf{X}\boldsymbol{\Sigma_0}\mathbf{X}^\top) && \text{(Marginal likelihood)} \\
p(\boldsymbol{y}|\mathbf{X}) &= \mathcal{N}(\mathbf{0}, \beta^{-1}\mathbf{I}_N + \mathbf{X}\mathbf{X}^\top) && \text{(if } \boldsymbol{\mu}_0 = 0 \text{ and } \boldsymbol{\Sigma}_0 = \mathbf{I}\text{)}
\end{aligned}
$$

Linear Regression

- Let's first consider the (probabilistic) linear regression model

$$
\begin{aligned}
p(\boldsymbol{w}) &= \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu_0}, \Sigma_0) && \text{(Prior)} \\
p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) &= \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}_n, \beta^{-1}) && \text{(Likelihood w.r.t. one obs.)} \\
p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) &= \mathcal{N}(\mathbf{X}\boldsymbol{w}, \beta^{-1}\mathbf{I}_N) && \text{(Likelihood w.r.t. } N \text{ obs.)} \\
p(\boldsymbol{y}|\mathbf{X}) &= \int p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) p(\boldsymbol{w}) d\boldsymbol{w} = \mathcal{N}(\mathbf{X}\boldsymbol{\mu_0}, \beta^{-1}\mathbf{I}_N + \mathbf{X}\boldsymbol{\Sigma_0}\mathbf{X}^\top) && \text{(Marginal likelihood)} \\
p(\boldsymbol{y}|\mathbf{X}) &= \mathcal{N}(\mathbf{0}, \beta^{-1}\mathbf{I}_N + \mathbf{X}\mathbf{X}^\top) && \text{(if } \boldsymbol{\mu}_0 = 0 \text{ and } \boldsymbol{\Sigma}_0 = \mathbf{I}) \\
p(\boldsymbol{y}|\mathbf{X}) &= \mathcal{N}(\mathbf{0}, \mathbf{X}\mathbf{X}^\top) && \text{(if } \beta = \infty, \text{ i.e., zero noise)}
\end{aligned}
$$

Linear Regression

- Let's first consider the (probabilistic) linear regression model

$$
\begin{aligned}
p(\boldsymbol{w}) &= \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu_0}, \Sigma_0) && \text{(Prior)} \\
p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) &= \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}_n, \beta^{-1}) && \text{(Likelihood w.r.t. one obs.)} \\
p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) &= \mathcal{N}(\mathbf{X}\boldsymbol{w}, \beta^{-1}\mathbf{I}_N) && \text{(Likelihood w.r.t. } N \text{ obs.)} \\
p(\boldsymbol{y}|\mathbf{X}) &= \int p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) p(\boldsymbol{w}) d\boldsymbol{w} = \mathcal{N}(\mathbf{X}\boldsymbol{\mu_0}, \beta^{-1}\mathbf{I}_N + \mathbf{X}\boldsymbol{\Sigma_0}\mathbf{X}^\top) && \text{(Marginal likelihood)} \\
p(\boldsymbol{y}|\mathbf{X}) &= \mathcal{N}(\mathbf{0}, \beta^{-1}\mathbf{I}_N + \mathbf{X}\mathbf{X}^\top) && (\text{if } \boldsymbol{\mu}_0 = 0 \text{ and } \boldsymbol{\Sigma}_0 = \mathbf{I}) \\
p(\boldsymbol{y}|\mathbf{X}) &= \mathcal{N}(\mathbf{0}, \mathbf{X}\mathbf{X}^\top) && (\text{if } \beta = \infty, \text{ i.e., zero noise})
\end{aligned}
$$

- Thus the joint marginal distr. of $\boldsymbol{y}$ conditioned on $\mathbf{X}$ is the following multivariate normal

$$
\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \boldsymbol{x}_1^\top \boldsymbol{x}_1 & \dots & \boldsymbol{x}_1^\top \boldsymbol{x}_N \\ \boldsymbol{x}_2^\top \boldsymbol{x}_1 & \dots & \boldsymbol{x}_2^\top \boldsymbol{x}_N \\ \vdots & \ddots & \vdots \\ \boldsymbol{x}_N^\top \boldsymbol{x}_1 & \dots & \boldsymbol{x}_N^\top \boldsymbol{x}_N \end{bmatrix} \right)
$$

Linear Regression

- Let's first consider the (probabilistic) linear regression model

$$
\begin{aligned}
p(\boldsymbol{w}) &= \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu_0}, \Sigma_0) && \text{(Prior)} \\
p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) &= \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}_n, \beta^{-1}) && \text{(Likelihood w.r.t. one obs.)} \\
p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) &= \mathcal{N}(\mathbf{X}\boldsymbol{w}, \beta^{-1}\mathbf{I}_N) && \text{(Likelihood w.r.t. } N \text{ obs.)} \\
p(\boldsymbol{y}|\mathbf{X}) &= \int p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) p(\boldsymbol{w}) d\boldsymbol{w} = \mathcal{N}(\mathbf{X}\boldsymbol{\mu_0}, \beta^{-1}\mathbf{I}_N + \mathbf{X}\boldsymbol{\Sigma_0}\mathbf{X}^\top) && \text{(Marginal likelihood)} \\
p(\boldsymbol{y}|\mathbf{X}) &= \mathcal{N}(\mathbf{0}, \beta^{-1}\mathbf{I}_N + \mathbf{X}\mathbf{X}^\top) && (\text{if } \boldsymbol{\mu}_0 = 0 \text{ and } \boldsymbol{\Sigma}_0 = \mathbf{I}) \\
p(\boldsymbol{y}|\mathbf{X}) &= \mathcal{N}(\mathbf{0}, \mathbf{X}\mathbf{X}^\top) && (\text{if } \beta = \infty, \text{ i.e., zero noise})
\end{aligned}
$$

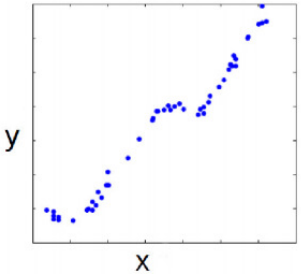
- Thus the joint marginal distr. of $\boldsymbol{y}$ conditioned on $\mathbf{X}$ is the following multivariate normal

$$
\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \boldsymbol{x}_1^\top \boldsymbol{x}_1 \ldots \boldsymbol{x}_1^\top \boldsymbol{x}_N \\ \boldsymbol{x}_2^\top \boldsymbol{x}_1 \ldots \boldsymbol{x}_2^\top \boldsymbol{x}_N \\ \vdots \quad \ddots \quad \vdots \\ \boldsymbol{x}_N^\top \boldsymbol{x}_1 \ldots \boldsymbol{x}_N^\top \boldsymbol{x}_N \end{bmatrix} \right)
$$

- Note that the cov. matrix above tells us how the different outputs "co-vary" w.r.t. each other

Nonlinear Regression

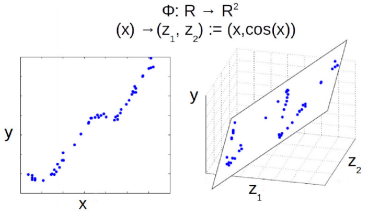
- Consider the following dataset for regression



- Input-output relationship is NOT linear
- A linear model would do badly on such data!

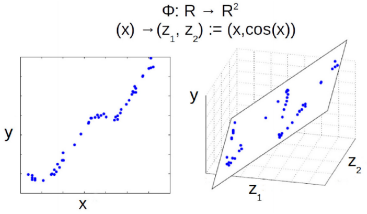
Nonlinear Regression via Mapping Inputs to Higher Dimensions

- Suppose we replace each input $\boldsymbol{x}_n$ by a higher dimensional mapping $\phi(\boldsymbol{x}_n)$



Nonlinear Regression via Mapping Inputs to Higher Dimensions

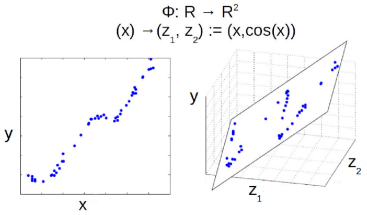
- Suppose we replace each input $\boldsymbol{x}_n$ by a higher dimensional mapping $\phi(\boldsymbol{x}_n)$



- Now the nonlinear regression problem becomes a linear reg. problem in a higher dim. space

Nonlinear Regression via Mapping Inputs to Higher Dimensions

- Suppose we replace each input $\boldsymbol{x}_n$ by a higher dimensional mapping $\phi(\boldsymbol{x}_n)$

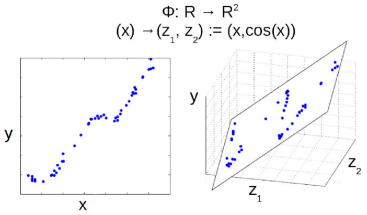


- Now the nonlinear regression problem becomes a linear reg. problem in a higher dim. space
- The joint distribution $\boldsymbol{y}$ given $\mathbf{X}$ will be the same as the linear case, except that each entry $\boldsymbol{x}_n^\top \boldsymbol{x}_m$ in the covariance matrix is replaced by $\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m)$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \phi(\boldsymbol{x}_1)^\top \phi(\boldsymbol{x}_1) & \dots & \phi(\boldsymbol{x}_1)^\top \phi(\boldsymbol{x}_N) \\ \phi(\boldsymbol{x}_2)^\top \phi(\boldsymbol{x}_1) & \dots & \phi(\boldsymbol{x}_2)^\top \phi(\boldsymbol{x}_N) \\ \vdots & \ddots & \vdots \\ \phi(\boldsymbol{x}_N)^\top \phi(\boldsymbol{x}_1) & \dots & \phi(\boldsymbol{x}_N)^\top \phi(\boldsymbol{x}_N) \end{bmatrix} \right)$$

Nonlinear Regression via Mapping Inputs to Higher Dimensions

- Suppose we replace each input $\boldsymbol{x}_n$ by a higher dimensional mapping $\phi(\boldsymbol{x}_n)$



- Now the nonlinear regression problem becomes a linear reg. problem in a higher dim. space
- The joint distribution $\boldsymbol{y}$ given $\mathbf{X}$ will be the same as the linear case, except that each entry $\boldsymbol{x}_n^\top \boldsymbol{x}_m$ in the covariance matrix is replaced by $\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m)$

$$\begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ . \\ . \\ 0 \end{bmatrix}, \begin{bmatrix} \phi(\boldsymbol{x}_1)^\top \phi(\boldsymbol{x}_1) \dots \phi(\boldsymbol{x}_1)^\top \phi(\boldsymbol{x}_N) \\ \phi(\boldsymbol{x}_2)^\top \phi(\boldsymbol{x}_1) \dots \phi(\boldsymbol{x}_2)^\top \phi(\boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \phi(\boldsymbol{x}_N)^\top \phi(\boldsymbol{x}_1) \dots \phi(\boldsymbol{x}_N)^\top \phi(\boldsymbol{x}_N) \end{bmatrix} \right)$$

- Note that $\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m)$ is the Euclidean similarity between $\boldsymbol{x}_n$ and $\boldsymbol{x}_n$ in the high-dim space

A Notion of Similarity: Kernel Functions

- Consider a covariance/kernel function κ that measures similarity between two inputs

A Notion of Similarity: Kernel Functions

- Consider a covariance/kernel function κ that measures similarity between two inputs
- Suppose κ 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 (Euclidean) similarity in that space

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

A Notion of Similarity: Kernel Functions

- Consider a covariance/kernel function κ that measures similarity between two inputs
- Suppose κ 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 (Euclidean) similarity in that space

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

- An example: Radial basis function (RBF) kernel: $\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\frac{||\boldsymbol{x}_n - \boldsymbol{x}_m||^2}{\gamma}\right)$ where ϕ_n corresponds to an infinite dimensional mapping. Another example: Polynomial kernel

A Notion of Similarity: Kernel Functions

- Consider a covariance/kernel function κ that measures similarity between two inputs
- Suppose κ 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 (Euclidean) similarity in that space

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

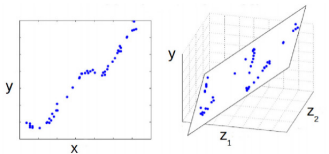
- An example: Radial basis function (RBF) kernel: $\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\frac{||\boldsymbol{x}_n - \boldsymbol{x}_m||^2}{\gamma}\right)$ where ϕ_n corresponds to an infinite dimensional mapping. Another example: Polynomial kernel
- Allows extending linear models to nonlinear models without explicitly computing $\phi(\boldsymbol{x}_n)$

A Notion of Similarity: Kernel Functions

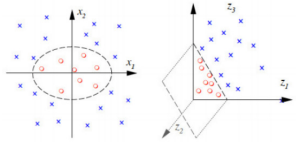
- Consider a covariance/kernel function κ that measures similarity between two inputs
- Suppose κ corresponds to implicitly mapping data to a higher dimensional space via a feature mapping ϕ ($\boldsymbol{x} \rightarrow \phi(\boldsymbol{x})$) and computing the dot product (Euclidean) similarity in that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

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

- An example: Radial basis function (RBF) kernel: $\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\frac{||\boldsymbol{x}_n - \boldsymbol{x}_m||^2}{\gamma}\right)$ where ϕ_n corresponds to an infinite dimensional mapping. Another example: Polynomial kernel
- Allows extending linear models to nonlinear models without explicitly computing $\phi(\boldsymbol{x}_n)$



Nonlinear Regression



Nonlinear Classification

Nonlinear Regression

- We saw that, for a nonlin. reg. model with each $\boldsymbol{x}_n$ replaced by a high-dim map $\phi(\boldsymbol{x}_n)$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \phi(\boldsymbol{x}_1)^\top \phi(\boldsymbol{x}_1) \dots \phi(\boldsymbol{x}_1)^\top \phi(\boldsymbol{x}_N) \\ \phi(\boldsymbol{x}_2)^\top \phi(\boldsymbol{x}_1) \dots \phi(\boldsymbol{x}_2)^\top \phi(\boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \phi(\boldsymbol{x}_N)^\top \phi(\boldsymbol{x}_1) \dots \phi(\boldsymbol{x}_N)^\top \phi(\boldsymbol{x}_N) \end{bmatrix} \right)$$

Nonlinear Regression

- We saw that, for a nonlin. reg. model with each $\boldsymbol{x}_n$ replaced by a high-dim map $\phi(\boldsymbol{x}_n)$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \phi(\boldsymbol{x}_1)^\top \phi(\boldsymbol{x}_1) \dots \phi(\boldsymbol{x}_1)^\top \phi(\boldsymbol{x}_N) \\ \phi(\boldsymbol{x}_2)^\top \phi(\boldsymbol{x}_1) \dots \phi(\boldsymbol{x}_2)^\top \phi(\boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \phi(\boldsymbol{x}_N)^\top \phi(\boldsymbol{x}_1) \dots \phi(\boldsymbol{x}_N)^\top \phi(\boldsymbol{x}_N) \end{bmatrix} \right)$$

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

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \kappa(\boldsymbol{x}_2, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix} \right)$$

Nonlinear Regression

- We saw that, for a nonlin. reg. model with each $\boldsymbol{x}_n$ replaced by a high-dim map $\phi(\boldsymbol{x}_n)$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \phi(\boldsymbol{x}_1)^\top \phi(\boldsymbol{x}_1) \dots \phi(\boldsymbol{x}_1)^\top \phi(\boldsymbol{x}_N) \\ \phi(\boldsymbol{x}_2)^\top \phi(\boldsymbol{x}_1) \dots \phi(\boldsymbol{x}_2)^\top \phi(\boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \phi(\boldsymbol{x}_N)^\top \phi(\boldsymbol{x}_1) \dots \phi(\boldsymbol{x}_N)^\top \phi(\boldsymbol{x}_N) \end{bmatrix} \right)$$

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

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \kappa(\boldsymbol{x}_2, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix} \right)$$

- More compactly, we can write $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{0}, \mathbf{C})$ where $C_{nm} = \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) = \phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m)$

Nonlinear Regression

- We saw that, for a nonlin. reg. model with each $\boldsymbol{x}_n$ replaced by a high-dim map $\phi(\boldsymbol{x}_n)$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \phi(\boldsymbol{x}_1)^\top \phi(\boldsymbol{x}_1) \dots \phi(\boldsymbol{x}_1)^\top \phi(\boldsymbol{x}_N) \\ \phi(\boldsymbol{x}_2)^\top \phi(\boldsymbol{x}_1) \dots \phi(\boldsymbol{x}_2)^\top \phi(\boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \phi(\boldsymbol{x}_N)^\top \phi(\boldsymbol{x}_1) \dots \phi(\boldsymbol{x}_N)^\top \phi(\boldsymbol{x}_N) \end{bmatrix} \right)$$

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

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \kappa(\boldsymbol{x}_2, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix} \right)$$

- More compactly, we can write $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{0}, \mathbf{C})$ where $C_{nm} = \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) = \phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m)$
- Note: In the noisy case, we only need to add β^{-1} $(=\sigma^2)$ to diagonals of $\mathbf{C}$

Nonlinear Regression

- Assume noisy case. Consider joint distr. of N training outputs $\boldsymbol{y}$ and test output y_* (input $\boldsymbol{x}_*$)

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

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

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

Nonlinear Regression

- Assume noisy case. Consider joint distr. of N training outputs $\boldsymbol{y}$ and test output y_* (input $\boldsymbol{x}_*$)

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

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

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

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

Nonlinear Regression

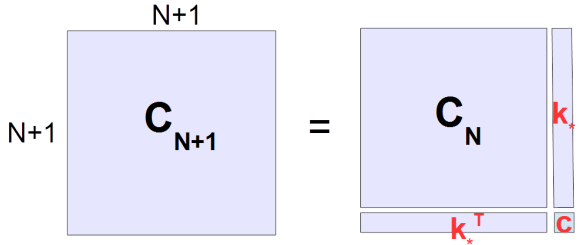
- Assume noisy case. Consider joint distr. of N training outputs $\boldsymbol{y}$ and test output y_* (input $\boldsymbol{x}_*$)

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

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

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

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



Nonlinear Regression

- Given the jointly Gaussian distribution

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

Nonlinear Regression

- Given the jointly Gaussian distribution

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

- The desired predictive posterior will be (using conditional from joint property of Gaussian)

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

Nonlinear Regression

- Given the jointly Gaussian distribution

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

- The desired predictive posterior will be (using conditional from joint property of Gaussian)

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

- How does this compare with linear model's posterior predictive?

Nonlinear Regression

- Given the jointly Gaussian distribution

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

- The desired predictive posterior will be (using conditional from joint property of Gaussian)

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

- How does this compare with linear model's posterior predictive?
  - One key difference is that the model's prediction now depends explicitly on all the training data

Nonlinear Regression

- 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 desired predictive posterior will be (using conditional from joint property of Gaussian)

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

- How does this compare with linear model's posterior predictive?
  - One key difference is that the model's prediction now depends explicitly on all the training data
  - Such models are called "nonparametric" models (model's complexity depends on the data)

Nonlinear Regression

- Given the jointly Gaussian distribution

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

- The desired predictive posterior will be (using conditional from joint property of Gaussian)

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

- How does this compare with linear model's posterior predictive?
  - One key difference is that the model's prediction now depends explicitly on all the training data
  - Such models are called "nonparametric" models (model's complexity depends on the data)
  - A downside: Making predictions gets slower

Learning Nonlinear Functions via Gaussian Processes (what we just saw, but in a more formal way)

Gaussian Process

- Recall that our kernel version of probabilistic linear regression (noiseless case) had

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \kappa(\boldsymbol{x}_2, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix} \right)$$

Gaussian Process

- Recall that our kernel version of probabilistic linear regression (noiseless case) had

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \kappa(\boldsymbol{x}_2, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix}\right)$$

- Note: A slightly general version would be with nonzero mean

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\boldsymbol{x}_1) \\ \mu(\boldsymbol{x}_2) \\ \vdots \\ \mu(\boldsymbol{x}_N) \end{bmatrix}, \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \kappa(\boldsymbol{x}_2, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix}\right)$$

  where μ is a "mean function"

Gaussian Process

- Recall that our kernel version of probabilistic linear regression (noiseless case) had

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \kappa(\boldsymbol{x}_2, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix}\right)$$

- Note: A slightly general version would be with nonzero mean

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\boldsymbol{x}_1) \\ \mu(\boldsymbol{x}_2) \\ \vdots \\ \mu(\boldsymbol{x}_N) \end{bmatrix}, \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \kappa(\boldsymbol{x}_2, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix}\right)$$

  where μ is a "mean function"

- Suppose we define $y_n = f(\boldsymbol{x}_n)$ where f is the (possibly nonlinear) function that maps $\boldsymbol{x}$ to y

Gaussian Process

- Recall that our kernel version of probabilistic linear regression (noiseless case) had

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \kappa(\boldsymbol{x}_2, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix}\right)$$

- Note: A slightly general version would be with nonzero mean

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\boldsymbol{x}_1) \\ \mu(\boldsymbol{x}_2) \\ \vdots \\ \mu(\boldsymbol{x}_N) \end{bmatrix}, \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \kappa(\boldsymbol{x}_2, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix}\right)$$

  where μ is a "mean function"

- Suppose we define $y_n = f(\boldsymbol{x}_n)$ where f is the (possibly nonlinear) function that maps $\boldsymbol{x}$ to y
- Gaussian Process (GP) formally defines a distribution over nonlinear functions and help us perform Bayesian inference on such functions

Gaussian Process

- Recall that our kernel version of probabilistic linear regression (noiseless case) had

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \kappa(\boldsymbol{x}_2, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix} \right)$$

- Note: A slightly general version would be with nonzero mean

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\boldsymbol{x}_1) \\ \mu(\boldsymbol{x}_2) \\ \vdots \\ \mu(\boldsymbol{x}_N) \end{bmatrix}, \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \kappa(\boldsymbol{x}_2, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots \quad \ddots \quad \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) \dots \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix} \right)$$

  where μ is a "mean function"

- Suppose we define $y_n = f(\boldsymbol{x}_n)$ where f is the (possibly nonlinear) function that maps $\boldsymbol{x}$ to y

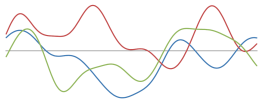
- Gaussian Process (GP) formally defines a distribution over nonlinear functions and help us perform Bayesian inference on such functions (important: not limited to only regression problems)

Gaussian Process

- A Gaussian Process, denoted as $\mathcal{GP}(\mu, \kappa)$, defines a distribution over functions
  - The GP is defined by mean function μ and covariance function κ

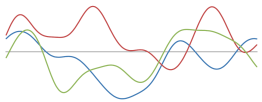
Gaussian Process

- A Gaussian Process, denoted as $\mathcal{GP}(\mu, \kappa)$, defines a distribution over functions
  - The GP is defined by mean function μ and covariance function κ
- Draw from a $\mathcal{GP}(\mu, \kappa)$ will give us a random function f (imagine it as an infinite dim. vector)



Gaussian Process

- A Gaussian Process, denoted as $\mathcal{GP}(\mu, \kappa)$, defines a distribution over functions
  - The GP is defined by mean function μ and covariance function κ
- Draw from a $\mathcal{GP}(\mu, \kappa)$ will give us a random function f (imagine it as an infinite dim. vector)

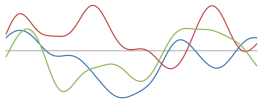


- Mean function μ models the "average" function f from $\mathcal{GP}(\mu, \kappa)$

$$\mu(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})]$$

Gaussian Process

- A Gaussian Process, denoted as $\mathcal{GP}(\mu, \kappa)$, defines a distribution over functions
  - The GP is defined by mean function μ and covariance function κ
- Draw from a $\mathcal{GP}(\mu, \kappa)$ will give us a random function f (imagine it as an infinite dim. vector)



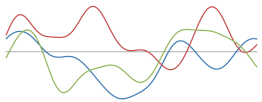
- Mean function μ models the "average" function f from $\mathcal{GP}(\mu, \kappa)$

$$\mu(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})]$$

- Cov. function κ models "shape/smoothness" of these functions

Gaussian Process

- A Gaussian Process, denoted as $\mathcal{GP}(\mu, \kappa)$, defines a distribution over functions
  - The GP is defined by mean function μ and covariance function κ
- Draw from a $\mathcal{GP}(\mu, \kappa)$ will give us a random function f (imagine it as an infinite dim. vector)



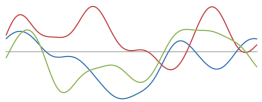
- Mean function μ models the "average" function f from $\mathcal{GP}(\mu, \kappa)$

$$\mu(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})]$$

- Cov. function κ models "shape/smoothness" of these functions
  - $\kappa(.,.)$ is a function that computes similarity between two inputs (just like a kernel function)

Gaussian Process

- A Gaussian Process, denoted as $\mathcal{GP}(\mu, \kappa)$, defines a distribution over functions
  - The GP is defined by mean function μ and covariance function κ
- Draw from a $\mathcal{GP}(\mu, \kappa)$ will give us a random function f (imagine it as an infinite dim. vector)



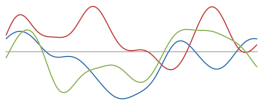
- Mean function μ models the "average" function f from $\mathcal{GP}(\mu, \kappa)$

$$\mu(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})]$$

- Cov. function κ models "shape/smoothness" of these functions
  - $\kappa(.,.)$ is a function that computes similarity between two inputs (just like a kernel function)
  - Note: $\kappa(.,.)$ needs to be positive definite (just like kernel functions)

Gaussian Process

- A Gaussian Process, denoted as $\mathcal{GP}(\mu, \kappa)$, defines a distribution over functions
  - The GP is defined by mean function μ and covariance function κ
- Draw from a $\mathcal{GP}(\mu, \kappa)$ will give us a random function f (imagine it as an infinite dim. vector)



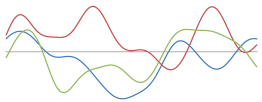
- Mean function μ models the "average" function f from $\mathcal{GP}(\mu, \kappa)$

$$\mu(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})]$$

- Cov. function κ models "shape/smoothness" of these functions
  - $\kappa(.,.)$ is a function that computes similarity between two inputs (just like a kernel function)
  - Note: $\kappa(.,.)$ needs to be positive definite (just like kernel functions)
- Can even learn μ and especially κ (makes GP very flexible to model, possibly nonlinear, functions)

Gaussian Process

- A Gaussian Process, denoted as $\mathcal{GP}(\mu, \kappa)$, defines a distribution over functions
  - The GP is defined by mean function μ and covariance function κ
- Draw from a $\mathcal{GP}(\mu, \kappa)$ will give us a random function f (imagine it as an infinite dim. vector)



- Mean function μ models the "average" function f from $\mathcal{GP}(\mu, \kappa)$

$$\mu(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})]$$

- Cov. function κ models "shape/smoothness" of these functions
  - $\kappa(.,.)$ is a function that computes similarity between two inputs (just like a kernel function)
  - Note: $\kappa(.,.)$ needs to be positive definite (just like kernel functions)
- Can even learn μ and especially κ (makes GP very flexible to model, possibly nonlinear, functions)
- GP can therefore be used as a flexible prior distribution over functions

Gaussian Process Prior

- f is said to be drawn from a $\mathcal{GP}(\mu, \kappa)$ if its finite dim. version is the following joint Gaussian

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

Gaussian Process Prior

- f is said to be drawn from a $\mathcal{GP}(\mu, \kappa)$ if its finite dim. version is the following joint Gaussian

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

- The above means that f's output at any finite set of inputs is jointly Gaussian

Gaussian Process Prior

- f is said to be drawn from a $\mathcal{GP}(\mu, \kappa)$ if its finite dim. version is the following joint Gaussian

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

- The above means that f's output at any finite set of inputs is jointly Gaussian
- Also makes intuitive sense: If $k(\boldsymbol{x}_n, \boldsymbol{x}_m)$ is large, we would expect $f(\boldsymbol{x}_n)$ and $f(\boldsymbol{x}_m)$ be the close

Gaussian Process Prior

- f is said to be drawn from a $\mathcal{GP}(\mu, \kappa)$ if its finite dim. version is the following joint Gaussian

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

- The above means that f's output at any finite set of inputs is jointly Gaussian
- Also makes intuitive sense: If $k(\boldsymbol{x}_n, \boldsymbol{x}_m)$ is large, we would expect $f(\boldsymbol{x}_n)$ and $f(\boldsymbol{x}_m)$ be the close
- We can also write the above more compactly as $\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$ where

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

Gaussian Process Prior

- f is said to be drawn from a $\mathcal{GP}(\mu, \kappa)$ if its finite dim. version is the following joint Gaussian

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

- The above means that f's output at any finite set of inputs is jointly Gaussian
- Also makes intuitive sense: If $k(\boldsymbol{x}_n, \boldsymbol{x}_m)$ is large, we would expect $f(\boldsymbol{x}_n)$ and $f(\boldsymbol{x}_m)$ be the close
- We can also write the above more compactly as $\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$ where

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

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

Gaussian Process Prior

- f is said to be drawn from a $\mathcal{GP}(\mu, \kappa)$ if its finite dim. version is the following joint Gaussian

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

- The above means that f's output at any finite set of inputs is jointly Gaussian
- Also makes intuitive sense: If $k(\boldsymbol{x}_n, \boldsymbol{x}_m)$ is large, we would expect $f(\boldsymbol{x}_n)$ and $f(\boldsymbol{x}_m)$ be the close
- We can also write the above more compactly as $\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$ where

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

  Note: $\mathbf{K}$ is also called the kernel matrix. $K_{nm} = \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m)$
- Note that $p(\mathbf{f}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$ can be seen as the finite-dimensional version of the GP prior over f

Gaussian Process Prior

- f is said to be drawn from a $\mathcal{GP}(\mu, \kappa)$ if its finite dim. version is the following joint Gaussian

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

- The above means that f's output at any finite set of inputs is jointly Gaussian
- Also makes intuitive sense: If $k(\boldsymbol{x}_n, \boldsymbol{x}_m)$ is large, we would expect $f(\boldsymbol{x}_n)$ and $f(\boldsymbol{x}_m)$ be the close
- We can also write the above more compactly as $\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$ where

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

  Note: $\mathbf{K}$ is also called the kernel matrix. $K_{nm} = \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m)$
- Note that $p(\mathbf{f}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$ can be seen as the finite-dimensional version of the GP prior over f
- If the mean function is zero, we will have $p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \mathbf{K})$

Gaussian Process Regression

GP Regression

- Training data: $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$. $\boldsymbol{x}_n \in \mathbb{R}^D$, $y_n \in \mathbb{R}$

GP 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

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

GP 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 noise: $\epsilon_n \sim \mathcal{N}(\epsilon_n|0, \sigma^2)$

GP 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 noise: $\epsilon_n \sim \mathcal{N}(\epsilon_n|0, \sigma^2)$
- This implies the following likelihood model: $p(y_n|f_n) = \mathcal{N}(y_n|f_n, \sigma^2)$

GP 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 noise: $\epsilon_n \sim \mathcal{N}(\epsilon_n|0, \sigma^2)$
- This implies the following likelihood model: $p(y_n|f_n) = \mathcal{N}(y_n|f_n, \sigma^2)$
- Denote $\boldsymbol{f} = [f_1, \ldots, f_N]$ and $\boldsymbol{y} = [y_1, \ldots, y_N]$.

GP 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 noise: $\epsilon_n \sim \mathcal{N}(\epsilon_n|0, \sigma^2)$
- This implies the following likelihood model: $p(y_n|f_n) = \mathcal{N}(y_n|f_n, \sigma^2)$
- Denote $\boldsymbol{f} = [f_1, \ldots, f_N]$ and $\boldsymbol{y} = [y_1, \ldots, y_N]$. For i.i.d. responses, the joint likelihood will be

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

GP 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 noise: $\epsilon_n \sim \mathcal{N}(\epsilon_n|0, \sigma^2)$
- This implies the following likelihood model: $p(y_n|f_n) = \mathcal{N}(y_n|f_n, \sigma^2)$
- Denote $\boldsymbol{f} = [f_1, \ldots, f_N]$ and $\boldsymbol{y} = [y_1, \ldots, y_N]$. For i.i.d. responses, the joint likelihood will be

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

- We now need a prior on the function f that enables us to model a nonlinear f

GP 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 noise: $\epsilon_n \sim \mathcal{N}(\epsilon_n|0, \sigma^2)$
- This implies the following likelihood model: $p(y_n|f_n) = \mathcal{N}(y_n|f_n, \sigma^2)$
- Denote $\boldsymbol{f} = [f_1, \ldots, f_N]$ and $\boldsymbol{y} = [y_1, \ldots, y_N]$. For i.i.d. responses, the joint likelihood will be

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

- We now need a prior on the function f that enables us to model a nonlinear f
- Let's choose zero mean Gaussian Process prior $\mathcal{GP}(0, \kappa)$ on f, which is equivalent to

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

where $K_{nm} = \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m)$.

GP 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 noise: $\epsilon_n \sim \mathcal{N}(\epsilon_n|0, \sigma^2)$
- This implies the following likelihood model: $p(y_n|f_n) = \mathcal{N}(y_n|f_n, \sigma^2)$
- Denote $\boldsymbol{f} = [f_1, \ldots, f_N]$ and $\boldsymbol{y} = [y_1, \ldots, y_N]$. For i.i.d. responses, the joint likelihood will be

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

- We now need a prior on the function f that enables us to model a nonlinear f
- Let's choose zero mean Gaussian Process prior $\mathcal{GP}(0, \kappa)$ on f, which is equivalent to

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

where $K_{nm} = \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m)$. For now, assume κ is a known function with fixed hyperparameters.

GP Regression

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

GP Regression

- The likelihood model: $p(\boldsymbol{y}|\mathbf{f}) = \mathcal{N}(\boldsymbol{y}|\mathbf{f}, \sigma^2\mathbf{I}_N)$
- The prior distribution: $p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$

GP 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: If we only care about making predictions on new test inputs, we don't actually need to compute the posterior $p(\mathbf{f}|\boldsymbol{y})$ here (although, due to Gaussians, we can do so easily)

GP 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: If we only care about making predictions on new test inputs, we don't actually need to compute the posterior $p(\mathbf{f}|\boldsymbol{y})$ here (although, due to Gaussians, we can do so easily)
- The marginal distribution of the training data responses $\boldsymbol{y}$ (after integrating out the "unknown" f)

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

GP 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: If we only care about making predictions on new test inputs, we don't actually need to compute the posterior $p(\mathbf{f}|\boldsymbol{y})$ here (although, due to Gaussians, we can do so easily)
- The marginal distribution of the training data responses $\boldsymbol{y}$ (after integrating out the "unknown" f)

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

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

GP 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: If we only care about making predictions on new test inputs, we don't actually need to compute the posterior $p(\mathbf{f}|\boldsymbol{y})$ here (although, due to Gaussians, we can do so easily)
- The marginal distribution of the training data responses $\boldsymbol{y}$ (after integrating out the "unknown" f)

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

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

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

GP 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: If we only care about making predictions on new test inputs, we don't actually need to compute the posterior $p(\mathbf{f}|\boldsymbol{y})$ here (although, due to Gaussians, we can do so easily)
- The marginal distribution of the training data responses $\boldsymbol{y}$ (after integrating out the "unknown" f)

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

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

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

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

GP Regression: 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

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

GP Regression: 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

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

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

GP Regression: Making Predictions

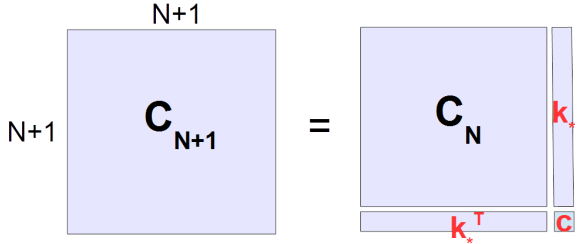
- Let's consider the joint distr. of N training responses $\boldsymbol{y}$ and test response y_*

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

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



GP Regression: Making Predictions

- Given the jointly Gaussian distribution

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

GP Regression: 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 desired predictive posterior will be (using conditional from joint property of Gaussian)

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

GP Regression: 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 desired predictive posterior will be (using conditional from joint property of Gaussian)

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

- Note: The test time cost

GP Regression: 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 desired predictive posterior will be (using conditional from joint property of Gaussian)

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

- Note: The test time cost is $\mathcal{O}(N)$

GP Regression: 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 desired predictive posterior will be (using conditional from joint property of Gaussian)

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

- Note: The test time cost is $\mathcal{O}(N)$: linear in the number of training examples

GP Regression: 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 desired predictive posterior will be (using conditional from joint property of Gaussian)

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

- Note: The test time cost is $\mathcal{O}(N)$: linear in the number of training examples
  - .. just like kernel SVM or nearest neighbor methods

GP Regression: 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 desired predictive posterior will be (using conditional from joint property of Gaussian)

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

- Note: The test time cost is $\mathcal{O}(N)$: linear in the number of training examples
  - .. just like kernel SVM or nearest neighbor methods
  - However, unlike kernel SVM or nearest neighbor methods, GP also gives us the variance σ_*^2

GP Regression: Interpreting GP Predictions

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

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

GP Regression: 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_*

GP Regression: 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_*
  - A kernel 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

GP Regression: 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_*
  - A kernel 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

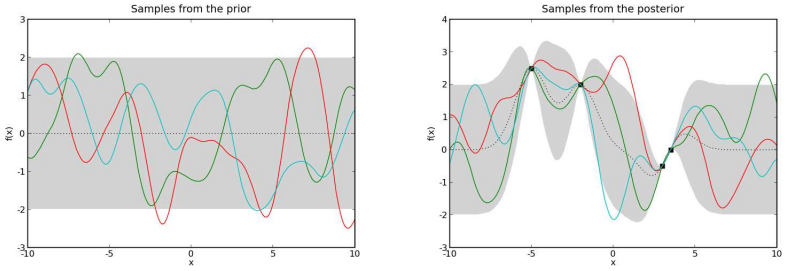
  - A nearest neighbors interpretation

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

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

GP Regression: Pictorially

A GP with a squared-exponential kernel function



Left: Samples of f from the prior $\mathcal{GP}(0, \kappa)$
Right: Samples of f from the posterior of f after 4 observations

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

Summary

- Gaussian Process is a very powerful framework for modeling nonlinear input-output relationships
- Can think of it as Bayesian kernel methods
- Can learn the kernel hyperparameters (big advantage)
- Can use kernels that can be compositions of many kernels

Next Class

- Other examples/applications of GPs
- Inference and other issues
- Gaussian Process Latent Variable Models