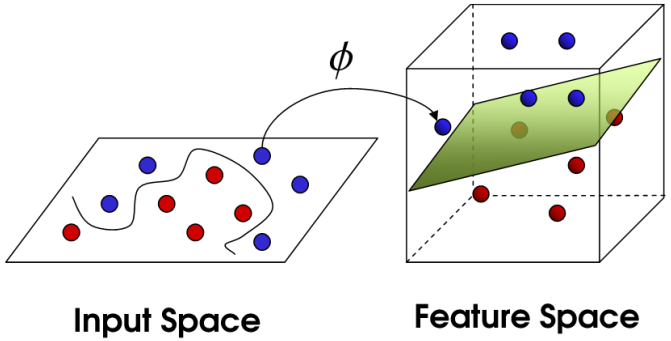
MATH 7339 - Machine Learning and Statistical Learning Theory 2

Section Kernel Methods

1. Dual representation
2. Kernel functions
3. Kernel Linear Regressions
4. Kernel Logistic Regression
5. Radial Basis Functions
6. Gaussian Processes

Feature map

For any linear method (e.g., linear regression, logistics regression, LDA), we can easily generalize it to non-linear method by introducing new variables (features).



For example,

$$z_1 = x_1, z_2 = x_2,$$

$$z_3 = x_1^2, z_4 = x_2^2, z_5 = x_1 x_2,$$

$$z_6 = x_1^3,\ z_7 = x_2^3, z_8 = x_1^2 x_2,\ z_9 = x_1 x_2^2\ , \ldots$$

Formally, we can consider this procedure as defining a **feature map:**

$$\phi\colon \mathbb{R}^d \rightarrow \mathbb{R}^D$$

$$\vec{x} \rightarrow \phi(\vec{x}) = \begin{bmatrix} \phi_1(\vec{x}) \\ \vdots \\ \phi_D(\vec{x}) \end{bmatrix}$$

$\phi_i(\vec{x})$ are the basis functions.

The **difficulty** is that dimension D is very large or even infinite.

For example, using polynomial of degree m, there are $D \sim O(d^m)$ parameters.

For a relatively easy question, if $d = 100$ and $m = 4$, there are about $d^4 \approx 4$ million parameters!

Question: How to solve the difficulty?

Answer: The kernel method (trick) to avoid the explicit computation in $\phi(\vec{x})$, but only compute the inner product by a very easy computation.

➢ Dual Representation of Linear Regressions:

Data: $D = \{(\vec{x}^{(i)}, y^{(i)}) \mid i = 1, \dots n\}$

Model: $h(\vec{x}) = \vec{\theta}^T \vec{x}$

If the **mean** of the data matrix X is **zero**, **Ridge regression** cost function:

$$J^{Ridge}(\vec{\theta}) := (X\vec{\theta} - \vec{y})^T (X\vec{\theta} - \vec{y}) + \lambda \vec{\theta}^T \vec{\theta}$$

The optimal solution is

$$\vec{\theta} = (X^T X + \lambda I)^{-1} X^T \vec{y}$$

Define $\vec{\theta} = X^T \vec{\beta}$ for some new parameter vector $\vec{\beta} \in \mathbb{R}^n$, called dual parameters

$$\vec{\theta} = X^T \vec{\beta} = \begin{bmatrix} \vec{x}^{(1)} & \dots & \vec{x}^{(n)} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_n \end{bmatrix} = \sum_{i=1}^{n} \beta_i \vec{x}^{(i)}$$

The **dual model for linear regression** is

$$h(\vec{x}) = \vec{\theta}^T \vec{x} = \langle \vec{x}, \vec{\theta} \rangle = \sum_{i=1}^{n} \beta_i \langle \vec{x}, \vec{x}^{(i)} \rangle$$

The **cost function**

$$J^{Ridge}(\vec{\beta}) := \left(XX^T\vec{\beta} - \vec{y}\right)^T \left(XX^T\vec{\beta} - \vec{y}\right) + \lambda \vec{\beta}^T XX^T \vec{\beta}$$

Solutions of $\vec{\beta}$ for optimizing the cost function:

$$\vec{\beta} = (XX^T + \lambda I)^{-1} \vec{y}$$

Here, $XX^T = \begin{bmatrix} \vdots \\ \cdots \langle \vec{x}^{(i)}, \vec{x}^{(j)} \rangle \cdots \\ \vdots \end{bmatrix}$

All computation is about $\vec{x}_*^T \vec{x}$

Bayesian Linear Regressions:

- **Data** : $\mathcal{D} = \{(\vec{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$
- **Model Assumption**: $y^{(i)} = f(\vec{x}^{(i)}) + \epsilon_i = \sum_{i=1}^{p} \theta_j h_j(\vec{x}^{(i)}) + \epsilon_i = \vec{h}^T(\vec{x}^{(i)})\vec{\theta} + \epsilon_i$

  ϵ_i are iid $N(0, \sigma^2)$

  Likelihood: $(y^{(i)}|\vec{\theta}, \vec{x}^{(i)}) \sim N(\vec{h}^T(\vec{x}^{(i)})\vec{\theta}, \sigma^2)$

- **Prior Assumption**: $\vec{\theta} \sim N(0, \Sigma)$ (or more generally $\vec{\theta} \sim N(\vec{\mu}, \Sigma)$)
- **Conclusion: Posterior** $\vec{\theta}|\mathcal{D}$ is also *a **normal distribution** with **mean***

$$E(\vec{\theta}|\mathcal{D}) = (H^T H + \Sigma^{-1}\sigma^2)^{-1} H^T \vec{y}$$

$$H_{ij} := h_j(\vec{x}^{(i)})$$

The **covariance matrix** is

$$Cov(\vec{\theta}|\mathcal{D}) = (H^T H + \sigma^2 \Sigma^{-1})^{-1} \sigma^2$$

Use the matrix identity: $(AB + cI)^{-1}A = A(BA + cI)^{-1}$

We can check:

$$E(\vec{\theta}|\mathcal{D}) = (H^T H + \Sigma^{-1}\sigma^2)^{-1}H^T\vec{y} = \Sigma H^T(H\Sigma H^T + \sigma^2 I)^{-1}\vec{y}$$

$$Cov(\vec{\theta}|\mathcal{D}) = (H^T H + \sigma^2\Sigma^{-1})^{-1}\sigma^2 = \Sigma - \Sigma H^T(H\Sigma H^T + \sigma^2 I)^{-1}H\Sigma$$

If we wish to use our model to predict the outputs y_* given $\vec{x}_*$, we will use the normal distribution with mean:

$$\mathrm{H}_*\Sigma H^T(H\Sigma H^T + \sigma^2 I)^{-1}\vec{y}$$

and variance

$$\mathrm{H}_*^{\mathrm{T}}\Sigma\mathrm{H}_* - \mathrm{H}_*^{\mathrm{T}}\Sigma H^T(H\Sigma H^T + \sigma^2 I)^{-1}H\Sigma\mathrm{H}_* + \sigma^2 I$$

$H_* \coloneqq \vec{h}(\vec{x}_*)$ So, all computations are about $\vec{h}(\vec{x}_*)^{\mathrm{T}}\Sigma\vec{h}(\vec{x})$

➢ The kernel method

Suppose there is a machine learning model, in the optimization of the cost and the prediction formula, only **inner products** of the data points are involved: $\langle \vec{x}^{(i)}, \vec{x}^{(j)} \rangle$, or $\langle \vec{x}^{(i)}, \vec{x} \rangle$ for prediction for $\vec{x}$.

After we applied the feature map,

$$\phi\colon\ \mathbb{R}^d \to \mathbb{R}^D$$

all calculations will be replaced by $\phi(\vec{x}) \in \mathbb{R}^D$. (Very large dimension)

We assume that all calculations **only involve inner products**

$$\langle \phi(\vec{x}^{(i)}), \phi(\vec{x}^{(j)}) \rangle \text{ or } \langle \phi(\vec{x}^{(i)}), \phi(\vec{x}) \rangle$$

Define it as the **Kernel function**:

$$K(\vec{x}^{(i)}, \vec{x}^{(j)}) := \langle \phi(\vec{x}^{(i)}), \phi(\vec{x}^{(j)}) \rangle$$

Example: (quadratic)

For $\vec{x}$ and $\vec{z} \in \mathbb{R}^3$, consider the quadratic feature map:

$$\phi(\vec{x}) := \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \end{bmatrix} \in \mathbb{R}^{3^2}$$

The **kernel function:**

$$K(\vec{x}, \vec{z}) := \langle \phi(\vec{x}), \phi(\vec{z}) \rangle = \sum_{i=1}^{d} \sum_{j=1}^{d} x_i x_j z_i z_j$$

$$= \left(\sum_{i=1}^{d} x_i z_i \right) \left(\sum_{j=1}^{d} x_j z_j \right) = \left(\sum_{i=1}^{d} x_i z_i \right)^2 = (\vec{x}^T \vec{z})^2$$

➢ Kernel Functions

1. Quadratic Kernel

For $\vec{x}$ and $\vec{z} \in \mathbb{R}^d$, define kernel function:

$$K(\vec{x}, \vec{z}) := (\vec{x}^T \vec{z} + c)^2$$

What is the feature map ϕ: $\mathbb{R}^d \to \mathbb{R}^D$?

$$\phi(\vec{x}) := \begin{bmatrix} x_1 x_1 \\ \vdots \\ x_1 x_d \\ \vdots \\ x_d x_d \\ \sqrt{2c}\, x_1 \\ \vdots \\ \sqrt{2c}\, x_3 \\ c \end{bmatrix} \in \mathbb{R}^{d^2+d+1}$$

Do we need the feature map ϕ?

2. Polynomial Kernel

For $\vec{x}$ and $\vec{z} \in \mathbb{R}^d$, define degree n polynomial kernel function:

$$K(\vec{x}, \vec{z}) := (\vec{x}^T \vec{z} + c)^n$$

3. Sigmoid Kernel

For $\vec{x}$ and $\vec{z} \in \mathbb{R}^d$, define Sigmoid kernel function:

$$K(\vec{x}, \vec{z}) := \tanh(\eta \vec{x}^T \vec{z} + c)$$

where $\tanh(t) = \frac{e^t - e^{-t}}{e^t + e^{-t}}$

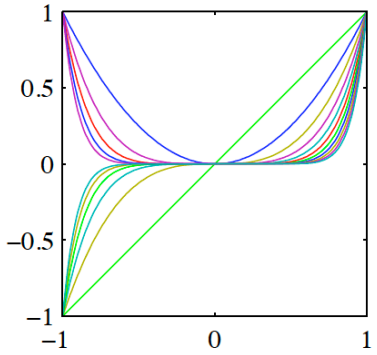
Illustrations of the kernel functions and basis functions.

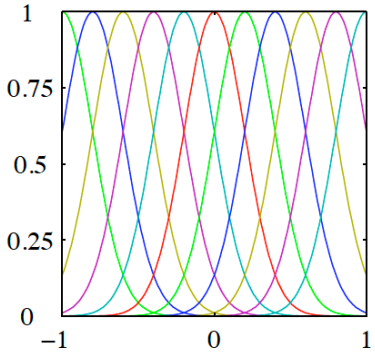
Polynomial

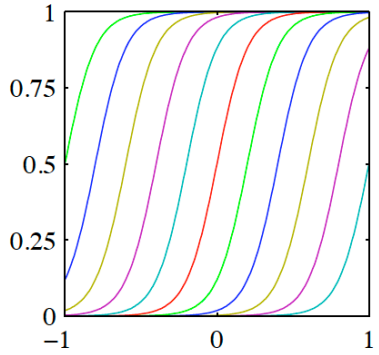
Gaussians

Sigmoid

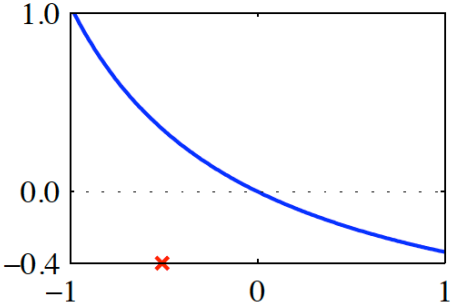
Basis Functions $\phi_i(x)$

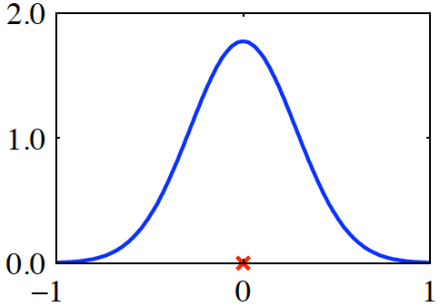


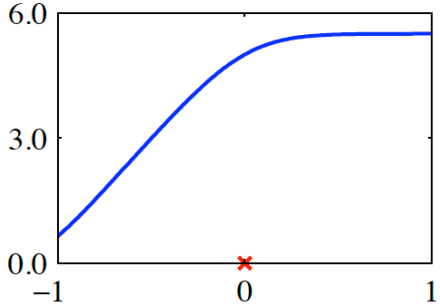




$K(x, x')$ where x' is the red cross ×







4. Gaussian Kernel

For $\vec{x}$ and $\vec{z} \in \mathbb{R}^d$, define **Gaussian kernel function (also called Squared exponential kernel, or RBF kernel.):**

$$K(\vec{x}, \vec{z}) := \exp\left(-\frac{\|\vec{x} - \vec{z}\|^2}{2\sigma^2}\right)$$

Remark:

- If σ is very small, then overfitting. If σ is very large, then underfitting
- What is the feature map ϕ: $\mathbb{R}^d \to \mathbb{R}^D$?

5. More popular kernels:

Laplacian kernel: $K(\vec{x}, \vec{z}) := \exp(-\alpha\|\vec{x} - \vec{z}\|)$

Abel kernel: $K(x, z) := \exp(-\alpha|x - z|)$ for $x, z \in \mathbb{R}$

6. More kernel See: The Kernel Cookbook:

https://www.cs.toronto.edu/~duvenaud/cookbook/

How to show a map is a feature maps?

Theorem: (Mercer 1909)

Let K: $\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a binary map.

The map K is a **kernel** function *if and only if* for any finite sequence $\{\vec{x}^{(1)}, \ldots, \vec{x}^{(m)}\}$, the matrix

$$M = \begin{bmatrix} & \vdots & \\ \cdots & K\left(\vec{x}^{(i)}, \vec{x}^{(j)}\right) & \cdots \\ & \vdots & \end{bmatrix}$$

is **symmetric** and **positive semi-definite.**

Proof: "$\Longrightarrow$"

If K is a kernel function, then there exists a map ϕ: $\mathbb{R}^d \to \mathbb{R}^D$ such that

$$K\left(\vec{x}^{(i)}, \vec{x}^{(j)}\right) = \left\langle \phi(\vec{x}^{(i)}), \phi(\vec{x}^{(j)}) \right\rangle$$

First, $K\left(\vec{x}^{(i)}, \vec{x}^{(j)}\right) = K\left(\vec{x}^{(j)}, \vec{x}^{(i)}\right)$ by the property of inner product.

Second, the quadratic form

$$\vec{z}^T M \vec{z} = \sum_{i,j}^{d} z_i \left\langle \phi(\vec{x}^{(i)}), \phi(\vec{x}^{(j)}) \right\rangle z_j = \sum_{i,j}^{d} \left\langle z_i \phi(\vec{x}^{(i)}), \phi(\vec{x}^{(j)}) z_j \right\rangle$$

$$= \left\langle \sum_{i=1}^{d} z_i \phi(\vec{x}^{(i)}), \sum_{j=1}^{d} z_j \phi(\vec{x}^{(j)}) \right\rangle = \left\| \sum_{i=1}^{d} z_i \phi(\vec{x}^{(i)}) \right\|^2 \geq 0$$

M defined by inner product this way is called the **Gram matrix**.

"$\Leftarrow$"

Suppose K is a binary map such that $M = \left[K\left(\vec{x}^{(i)}, \vec{x}^{(j)}\right)\right]$ satisfies the properties.

Consider $\phi_{(\vec{x})}(-) \coloneqq K(-, \vec{x})$, which is map from $\mathbb{R}^n$ to $\mathbb{R}$.

Let $\mathcal{F} \coloneqq \mathrm{Span}\left\{\phi_{(\vec{x})} \;\middle|\; \vec{x} \in \mathbb{R}^n\right\}$ be a subspace of the function space $C(\mathbb{R}^n, \mathbb{R})$

Claim 1. $\phi_{(\vec{x})}$ defines a map from $\mathbb{R}^n$ to $\mathcal{F}$.

Claim 2. $\mathcal{F}$ is an **inner product space** with

$$\left\langle \phi_{(\vec{x})}, \phi_{(\vec{z})} \right\rangle_{\mathcal{F}} \coloneqq K(\vec{x}, \vec{z})$$

How to construct new kernel functions from old kernels?

Theorem:

If K_1 and K_2 are kernel functions, then the following are also kernel functions.

- $K(\vec{x}, \vec{z}) := aK_1(\vec{x}, \vec{z}) + bK_2(\vec{x}, \vec{z})$, where $a, b \geq 0$
- $K(\vec{x}, \vec{z}) := K_1(\vec{x}, \vec{z})K_2(\vec{x}, \vec{z})$
- $K(\vec{x}, \vec{z}) \coloneqq K_1\big(f(\vec{x}), f(\vec{z})\big)$, where f is a function from $\mathbb{R}^d \to \mathbb{R}^M$
- $K(\vec{x}, \vec{z}) \coloneqq P\big(K_1(\vec{x}, \vec{z})\big)$, where $P(t)$ is a polynomial with non-negative coeffects.
- $K(\vec{x}, \vec{z}) \coloneqq \exp\big(K_1(\vec{x}, \vec{z})\big)$
- $K(\vec{x}, \vec{z}) \coloneqq \vec{x}^T S\vec{z}$, where S is a symmetric positive semidefinite matrix.
- $K(\vec{x}, \vec{z}) \coloneqq f(\vec{x})K_1(\vec{x}, \vec{z})f(\vec{z})$, where $f: \mathbb{R}^d \to \mathbb{R}$ is any function.

❑ Kernel linear regression

The **Kernel linear regression** is

$$h(\vec{x}) = \sum_{i=1}^{n} \beta_i \, K(\vec{x}\,, \vec{x}^{(i)})$$

The **cost function**

$$J^{Ridge}(\vec{\beta}) := \left(K\vec{\beta} - \vec{y}\right)^T \left(K\vec{\beta} - \vec{y}\right) + \lambda \vec{\beta}^T K \vec{\beta}$$

Solutions of $\vec{\beta}$ for optimizing the cost function:

$$\vec{\beta} = (K + \lambda I)^{-1} \, \vec{y}$$

Here, $K = \begin{bmatrix} \vdots \\ \cdots K(\vec{x}\,, \vec{x}^{(i)}) \cdots \\ \vdots \end{bmatrix}$

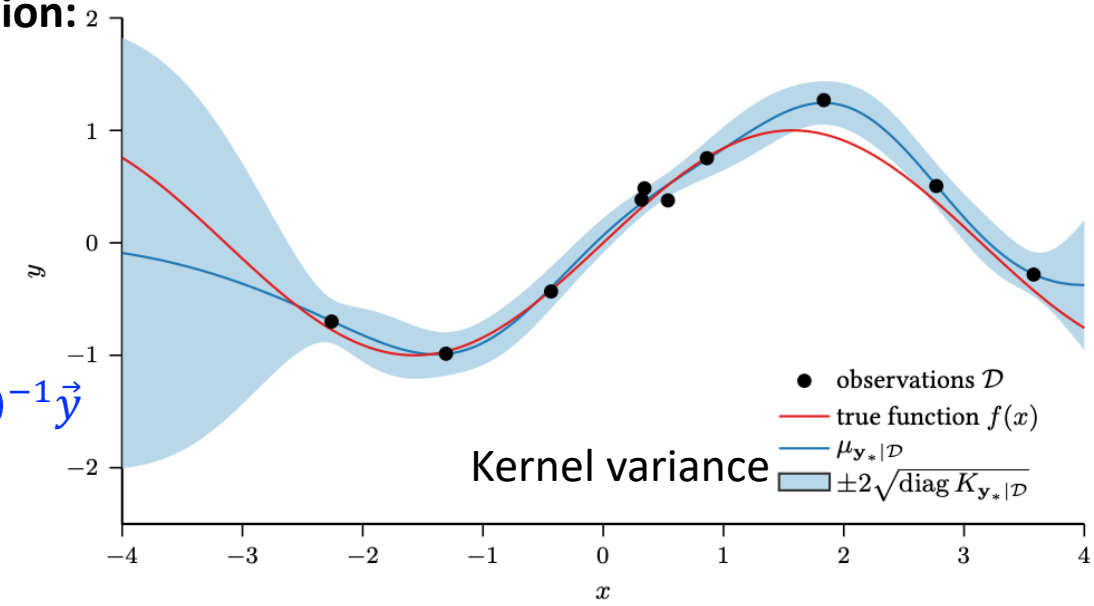
❑ **Kernel Bayesian Linear Regression:**

Predict the outputs y_* given $\vec{x}_*$, we will use the normal distribution with **mean**:

$$\mu_{y_*|\mathcal{D}} = K(\vec{x}_*, X)(K(X,X) + \sigma^2 I)^{-1}\vec{y}$$

and **variance**

$$K_{y_*|\mathcal{D}} = K(\vec{x}_*, \vec{x}_*) - K(\vec{x}_*, X)(K(X,X) + \sigma^2 I)^{-1}K(X, \vec{x}_*)$$



Example of Bayesian linear regression using the squared exponential covariance function.

$$K(\vec{x}, \vec{x}'; \lambda, l) \coloneqq \lambda^2 \exp\left(-\frac{\|\vec{x} - \vec{x}'\|^2}{2l^2}\right)$$

The true function is $f = \sin(x)$. The kernel parameters are $\lambda = l = 1$, and the noise variance was set to $\sigma^2 = 0.1^2$.

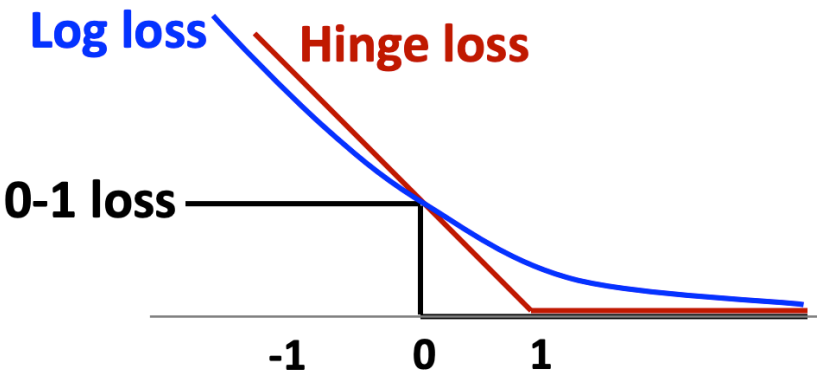
❑ Kernel Logististics regression

Logistic Regression with labels $\{-1, 1\}$

Model: $$P\left(Y = 1 \middle| \vec{x}, \vec{\theta}\right) = h_{\vec{\theta}}(\vec{x}) := \frac{1}{1 + e^{-\vec{\theta}^T \vec{x}}} = \frac{1}{1 + e^{-(\vec{w}^T \vec{x} + b)}}$$

The **Log loss** for each data point is

$$\text{loss}\left(h\left(\vec{x}^{(j)}\right), y^{(j)}\right) = -\log P\left(y^{(j)} \middle| \vec{x}^{(j)}, \vec{\theta}\right) = \log\left(1 + e^{-\left(\vec{\theta}^T \vec{x}^{(j)}\right) y^{(j)}}\right)$$



Suppose there is a **feature** map $\phi: \mathbb{R}^d \to \mathbb{R}^D$

$$h_{\vec{\theta}}(\vec{x}) := \frac{1}{1 + e^{-\vec{\theta}^T \phi(\vec{x})}}$$

Define weights in terms of features:

$$\vec{\theta} = \left[\phi(\vec{x}^{(1)}) \quad \ldots \phi\left(\vec{x}^{(N)}\right)\right] \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_N \end{bmatrix} = \sum_{i=1}^{N} \beta_i \, \phi(\vec{x}^{(i)})$$

The kernel logistics model:

$$P\left(Y = 1 \middle| \vec{x}, \vec{\theta}\right) = h_{\vec{\theta}}(\vec{x}) = \frac{1}{1 + e^{-\sum_{i=1}^{N} \beta_i \phi(\vec{x}^{(i)})^T \phi(\vec{x})}}$$

$$= \frac{1}{1 + e^{-\sum_{i=1}^{N} \beta_i \, K\left(\vec{x}^{(i)}, \vec{x}\right)}}$$

$$\text{Loss}(\vec{\beta}) = \frac{1}{N}\sum_{j=1}^{N} \text{loss}\left(h\left(\vec{x}^{(j)}\right), y^{(j)}\right) = \frac{1}{N}\sum_{j=1}^{N} \log\left(1 + e^{-\left(\sum_{i=1}^{N} \beta_i K\left(\vec{x}^{(i)}, \vec{x}^{(j)}\right)\right) y^{(j)}}\right)$$

Let $K_{ij} = K\left(\vec{x}^{(i)}, \vec{x}^{(j)}\right)$

$$= \frac{1}{N}\sum_{j=1}^{N} \log\left(1 + e^{-\left(\vec{\beta}^T K\right) y^{(j)}}\right)$$

Then we need to solve the optimization question

$$\underset{\vec{\beta}}{\text{argmin}}\, \text{Loss}(\vec{\beta})$$

by gradient descent or Newton's method.

Remark: we can also generalize the loss with penalty $\lambda \beta^T K \beta$

❑ **Kernel SVM (using hinge loss):** $y = 1\ or\ -1$

We already see the Kernel SVM, through margin maximization..

Equivalently, the soft margin SVM optimization problem is the same as **minimize** the **Hinge loss:**

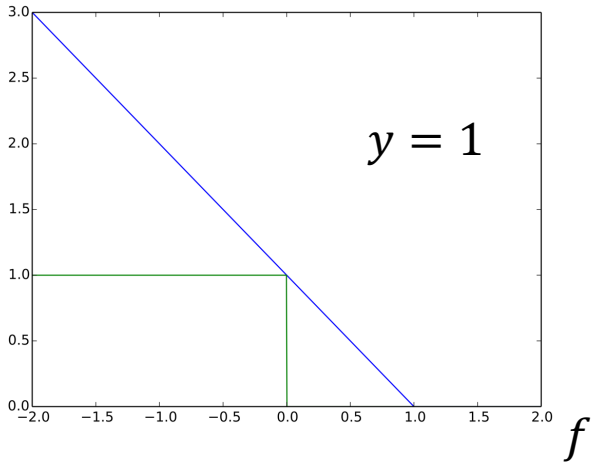
$$\min_{b,\vec{w}} \sum_{i=1}^{n} \left[1 - y^{(i)} f\left(x^{(i)}\right)\right]_+ + \frac{\lambda}{2} \|\vec{w}\|^2$$

Here: $f(\vec{x}) = \vec{w}^T \vec{x} + b$

Compare soft margin SVM, we set $\lambda = \frac{1}{C}$.

$l(y, f) \coloneqq [1 - yf]_+ = \max(0, 1 - yf)$

$l(1, f)$



Similar calculation as in kernel logistics, we can achieve the kernel SVM with hinge loss.

➢ Hilbert spaces and Kernels

Definition: Given a vector space V , a map(function) $f: V \to \mathbb{R}$ from V to the real numbers is **linear** if

$$f(a\vec{x} + b\vec{y}) = af(\vec{x}) + bf(\vec{y})$$

for any $a, b \in \mathbb{R}$, any $\vec{x}, \vec{y} \in V$

Definition: If V is an inner product space, we say that f is **bounded** if

$$f(\vec{x}) \leq C\|\vec{x}\|$$

for some fixed number $C > 0$ and all $\vec{x} \in V$

Reproducing Kernel Hilbert space

Definition. Let $X \subset \mathbb{R}^d$ be compact (i.e., a closed bounded subset). A (real) reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ on X is **a** Hilbert space of functions on X. (i.e., a complete collection of functions which is closed under addition and scalar multiplication, and for which an inner product is defined)

The space $\mathcal{H}$ also needs the property: for any fixed $\vec{x} \in X$ the evaluation function $\vec{x}^*$: $\mathcal{H} \to \mathbb{R}$ defined by

$$\vec{x}^*(f) := f(\vec{x})$$

is **bounded, linear** function on $\mathcal{H}$

Theorem: Given a reproducing kernel Hilbert space $\mathcal{H}$ of functions on $X \subset \mathbb{R}^d$, there exists a **unique** *symmetric positive kernel function* $K(\vec{x}, \vec{y})$ such that for all $f \in \mathcal{H}$,

$$f(\vec{x}) \coloneqq \langle f(\vec{z}), K(\vec{z}, \vec{x}) \rangle_{\mathcal{H}}$$

inner product above is in the variable $\vec{z}$. ($\vec{x}$ is fixed.)

This theorem means that evaluation of f at fixed $\vec{x}$ is equivalent to taking inner product of $f(\vec{z})$ with the fixed function $K(\vec{z}, \vec{x})$ (in variable $\vec{z}$ with $\vec{x}$ fixed)

Proof: Recall **Riesz Representation Theorem** from functional analysis: If $\phi: \mathcal{H} \to \mathbb{R}$ is a bounded linear functional on $\mathcal{H}$, there exists a unique $y \in \mathcal{H}$ such that $\phi(\vec{x}) = \langle y, x \rangle$ for any $\vec{x} \in \mathcal{H}$.

For any fixed $\vec{\mathrm{x}} \in \mathrm{X}$, recall $\vec{\mathrm{x}}^*$ is a bounded linear functional on $\mathcal{H}$. By Riesz Representation Theorem, there exists a fixed function, $K_{\vec{x}}(z)$ such that for all $f \in \mathcal{H}$

$$f(\vec{x}) = \vec{x}^*(f) = \langle f(z), K_{\vec{x}}(z) \rangle_{\mathcal{H}}$$

That is, evaluation of f at $\vec{x}$ is equivalent to an inner product with the function $K_{\vec{x}}(z)$.

Define $K(\vec{x}, \vec{y}) = K_{\vec{x}}(\vec{y})$.

1. $K(\vec{x}, \vec{y})$ is symmetric, that is $K(\vec{x}, \vec{y}) = K(\vec{y}, \vec{x})$

2. $K(\vec{x}, \vec{y})$ is positive definite (That is $\vec{c}^T K \vec{c} \geq 0$).

Definition: We call the above kernel $K(\vec{x}, \vec{y})$ the **reproducing kernel** of $\mathcal{H}$.

Definition: A **Mercer kernel** is a positive definite kernel $K(\vec{x}, \vec{y})$ which is also continuous as a function of x and y and bounded.

Definition: For a continuous function f on a compact set $X \subset \mathbb{R}^d$ we define

$$\|f\|_\infty := \max_{\vec{x} \in X} |f(\vec{x})|$$

Theorem

(i) For every Mercer kernel $K: X \times X \to \mathbb{R}$, there exists a **unique** Hilbert space $\mathcal{H}$ (an RKHS) of functions on X such that K is its reproducing kernel.

(ii) Moreover, this $\mathcal{H}$ consists of continuous functions, and for any $f \in \mathcal{H}$

$$\|f\|_\infty \leq M_K \, \|f\|_{\mathcal{H}}$$

where $M_K \coloneqq \max_{\vec{x},\vec{y} \in X} |K(\vec{x},\vec{y})|$

Every reproducing kernel K induces a unique RKHS,

Every RKHS has a unique reproducing kernel.

Every reproducing kernel is positive-definite,

Every positive definite kernel defines a unique RKHS, of which it is the unique reproducing kernel.

Radial Basis Functions(RBF)

Radial Basis Function (RBF) is a real-valued function whose value depends only on the **distance** from two points $\vec{x}$ and $\vec{c_i}$ in multi-dimensional space $\mathbb{R}^d$. ($\{\vec{c}_i\}_{i=1}^N$ is a set of fixed centers.)

$$\phi_i(\vec{x}) = h(\|\vec{x} - \vec{c}_i\|) \text{ for } i = 1, \dots, N$$

Here, $h: [0, \infty) \to \mathbb{R}$ is a **radial** function.

The RBFs are typically used to construct function approximations defined on scattered multidimensional data $\mathcal{D} = \left\{(\vec{x}^{(i)}, y^{(i)})\right\}_{i=1}^N$ of the form

$$f(\vec{x}) = \sum_{i=1}^{N} w_i h\left(\left\|\vec{x} - \vec{x}^{(i)}\right\|\right)$$

The coefficients can be calculated by least squares methods $\vec{w} = (H^T H)^{-1} H^T \vec{y}$.

RBFs were initially used (Powell, late 1970s) to perform interpolation (exact fit) rather than regression.

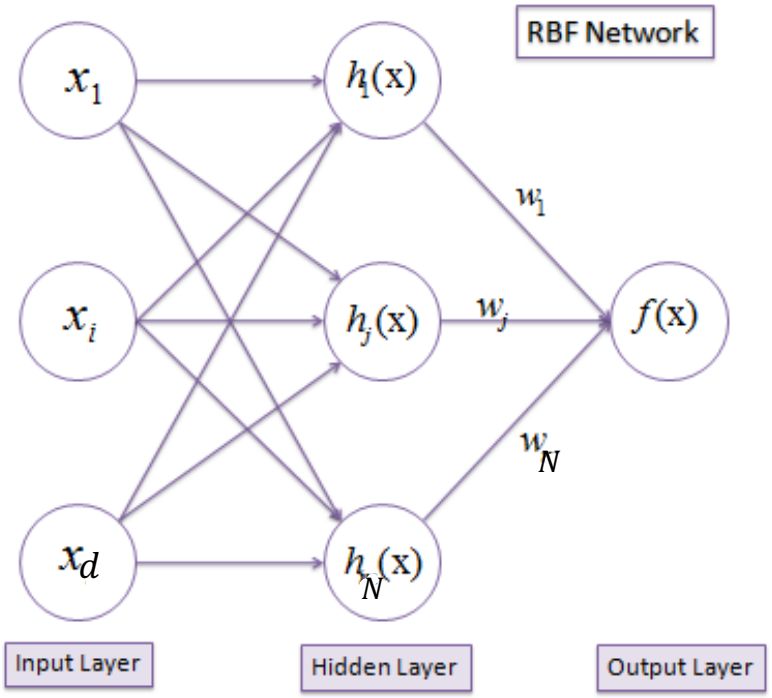
Euclidean norm is usually used in the distance between $\vec{x}\ and\ \vec{c}_i$.
The **Mahalanobis** distance $\|\vec{x} - \vec{c}_i\|^2 := (\vec{x} - \vec{c}_i)^T S(\vec{x} - \vec{c}_i)$ performs better with pattern recognition.

Commonly used of radial functions $h: [0, \infty) \to \mathbb{R}$ include

- **Gaussian**: $h(r, \sigma) = \exp\left(-\frac{r^2}{\sigma^2}\right)$, where σ is a hyperparameter (shape parameter).
- **Multiquadric:** $h(r) = \sqrt{r^2 + b}$
- **Inverse Multiquadric:** $h(r) = \frac{1}{\sqrt{r^2+b}}$
- **Thin plate spline:** $h(r) = r^2 \ln r$
- **Polyharmonic spline:** $h(r) = r^k$ for k=1,3,5,…

  $h(r) = r^k \ln r$ for k=2,4,6,…

➢ RBF Network



$$f(\vec{x}) = \sum_{i=1}^{N} w_i h\left(\left\|\vec{x} - \vec{x}^{(i)}\right\|\right)$$

$$h_i(\vec{x}) = h(\|\vec{x} - \vec{c}_i\|) \text{ for } i = 1, \ldots, N$$

- Dave Broomhead and David Lowe, “Multivariable Functional Interpolation and Adaptive Networks” (1988) connects the RBF to the neural net.

Normalized RBF network

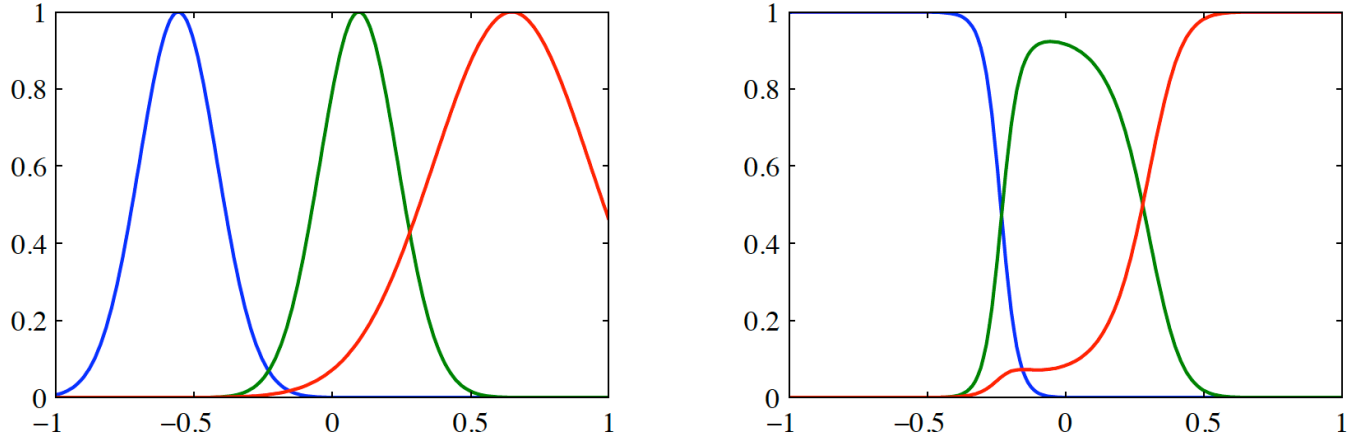
We can normalize the above RBF function

$$f(\vec{x}) = \sum_{i=1}^{N} w_i h\left(\left\|\vec{x} - \vec{x}^{(i)}\right\|\right)$$

As the **normalized** RBF network:

$$g(\vec{x}) := \frac{\sum_{i=1}^{N} w_i h\left(\left\|\vec{x} - \vec{x}^{(i)}\right\|\right)}{\sum_{i=1}^{N} h\left(\left\|\vec{x} - \vec{x}^{(i)}\right\|\right)} = \sum_{i=1}^{N} w_i u\left(\left\|\vec{x} - \vec{x}^{(i)}\right\|\right)$$

where $$u\left(\left\|\vec{x} - \vec{x}^{(i)}\right\|\right) := \frac{h\left(\left\|\vec{x} - \vec{x}^{(i)}\right\|\right)}{\sum_{i=1}^{N} h\left(\left\|\vec{x} - \vec{x}^{(i)}\right\|\right)}$$



Normalized Basis Functions.

Nadaraya-Watson Models

Data $\mathcal{D} = \left\{(\vec{x}^{(i)}, y^{(i)})\right\}_{i=1}^{N}$

We assume the noise on input variable $\vec{x}$ is $\vec{\xi}$ with distribution $v(\vec{\xi})$

The sum of square error is

$$E = \frac{1}{2}\sum_{i=1}^{N}\int \left(f\left(\vec{x}^{(i)} + \vec{\xi}\right) - y^{(n)}\right)^2 v(\vec{\xi}) d\,\vec{\xi}$$

Optimize E with respect to $f(\vec{x})$, we have a popular interpolation strategy is:

$$f(\vec{x}) = \sum_{i=1}^{N} y^{(i)} h\left(\vec{x} - \vec{x}^{(i)}\right)$$

where $h\left(\vec{x} - \vec{x}^{(i)}\right) = \dfrac{v(\vec{x} - \vec{x}^{(i)})}{\sum_{j=1}^{N} v(\vec{x} - \vec{x}^{(j)})}$ is the normalized basis.

Logistic map in time series:

The logistic map was derived from a differential equation describing population growth, popularized by Robert May. It has become the prototype for chaotic time series.

$$x(t+1) \coloneqq rx(t)\big(1-x(t)\big)$$

where r can be considered as a growth rate

Time Series Plots
http://s3.amazonaws.com/complexityexplorer/DynamicsAndChaos/Programs/time_series.html

Gaussian Process for Regression:

Consider the general regression problem:

Data $\mathcal{D} = \left\{(\vec{x}^{(i)}, y^{(i)})\right\}_{i=1}^{N}$ where $\vec{x}^{(i)} \in \mathcal{X} \subset \mathbb{R}^d$ and $y^{(i)} \in \mathcal{C} \subset \mathbb{R}$

Suppose $y^{(i)} = f\left(\vec{x}^{(i)}\right) + \epsilon_i$, where iid $\epsilon_i \sim N(0, \sigma^2)$

Goal: predict the value of the function $f(\vec{x}_*)$ for a test location $\vec{x}_*$.

Gaussian processes take a **non-parameteric** approach to regression. We select a **prior** distribution over the function f and condition this distribution on our observations, using the posterior distribution to make predictions. (Bayesian)

Problem: the latent function $f: \mathcal{X} \to \mathbb{R}$ is usually infinite dimensional; however, the multivariate Gaussian distribution is only useful in finite dimensions.
The Gaussian process is a natural generalization of the multivariate Gaussian distribution to potentially infinite settings.

Definition: A **Gaussian process** is a (potentially infinite) collection of **random variables** such that the **joint distribution** of any finite number of them is multivariate Gaussian.

A Gaussian process distribution on f is written

$$p(f) = \mathcal{GP}(f; \mu, K)$$

and just like the multivariate Gaussian distribution, is parameterized by its first two moments (now functions):

- $E[f] = \mu: \mathcal{X} \to \mathbb{R}$ the mean function.
- $E\left[\left(f(x) - \mu(x)\right)\left(f(x') - \mu(x')\right)\right] = K: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a positive semidefinite covariance function (or kernel.)

Mean and covariance functions

The **mean function** encodes the *central tendency* of the function, and is often assumed to be a constant (usually zero).

The **covariance function** encodes information about the shape and structure we expect the function to have. A simple and very common example is the squared exponential covariance:

$$K(\vec{x}, \vec{x}') = \exp\left(-\frac{1}{2}\|\vec{x} - \vec{x}'\|\right)$$

which encodes the notation that “nearby points should have similar function values.”

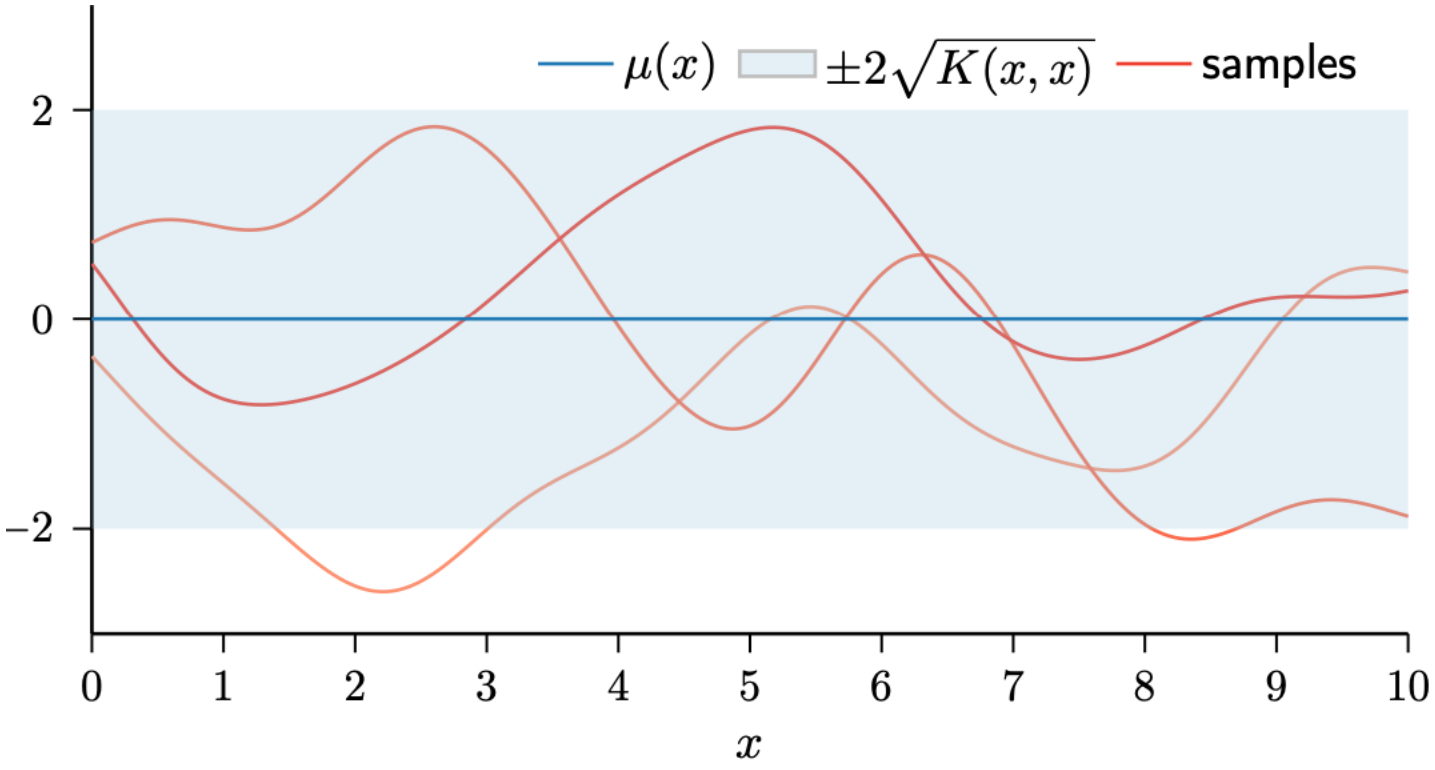
Prior on finite sets

Suppose we have selected a GP prior $\mathcal{GP}(f; \mu, K)$ for the function f.
Consider a **finite** set of points $\boldsymbol{X} \subseteq \mathcal{X}$. The GP prior on f, by definition, implies the following joint distribution on the associated function values $\boldsymbol{f} = f(\boldsymbol{X})$:

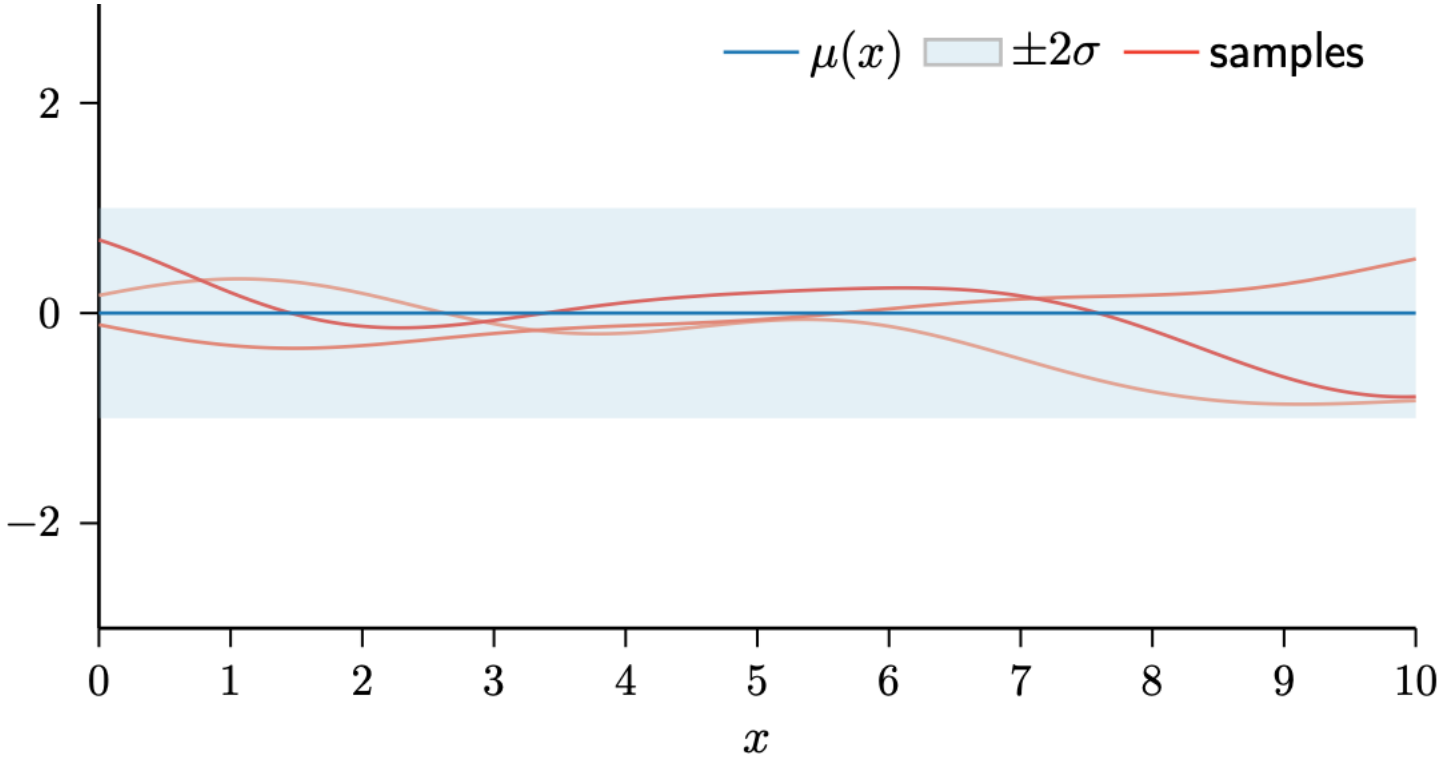
$$p(\boldsymbol{f}|\boldsymbol{X}) = N(\boldsymbol{f}; \mu(\boldsymbol{X}), K(\boldsymbol{X}, \boldsymbol{X}))$$

That is, we simply evaluate the mean and covariance functions at $\boldsymbol{X}$
and take the associated multivariate Gaussian distribution.

Prior: Sampling examples

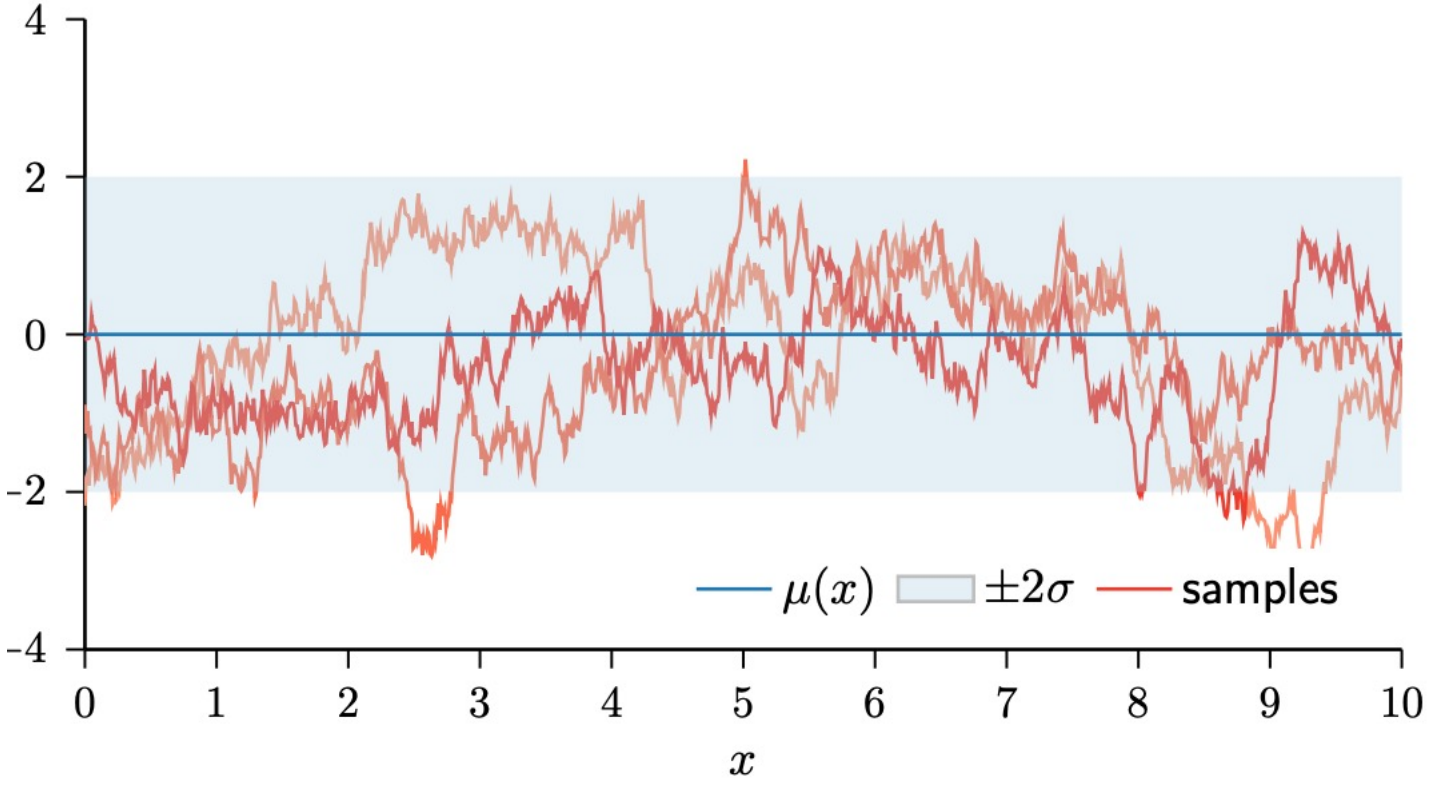


$$K(\vec{x}, \vec{x}') = \exp\left(-\frac{1}{2}\|\vec{x} - \vec{x}'\|^2\right)$$



$$K(\vec{x}, \vec{x}') = \lambda^2 \exp\left(-\frac{1}{2l^2}\|\vec{x} - \vec{x}'\|^2\right)$$

$$\lambda = \frac{1}{2}, \qquad l = 2.$$



$$K(\vec{x}, \vec{x}') = \exp(-\|\vec{x} - \vec{x}'\|)$$

From the prior to the posterior

We have constructed **prior** distributions over the function f.

How do we **condition** our prior on some observations $\boldsymbol{D} = (\boldsymbol{X}, \boldsymbol{f})$ to make predictions about the value of f at some points $\vec{x}_*$

Write the joint distribution between the training function values $f(\boldsymbol{X}) = \boldsymbol{f}$ and the test function values $f(\vec{x}_*) = f_*$

$$p(\boldsymbol{f}, f_*) = N\left(\begin{bmatrix}\boldsymbol{f}\\ f_*\end{bmatrix}; \begin{bmatrix}\boldsymbol{\mu}(\boldsymbol{X})\\ \mu(f_*)\end{bmatrix}, \begin{bmatrix}\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) & \boldsymbol{K}(\boldsymbol{X},\vec{\boldsymbol{x}}_*)\\ \boldsymbol{K}(\vec{\boldsymbol{x}}_*,\boldsymbol{X}) & \boldsymbol{K}(\vec{\boldsymbol{x}}_*,\vec{\boldsymbol{x}}_*)\end{bmatrix}\right)$$

Condition this multivariate Gaussian on the known training values $\boldsymbol{f}$.

$$p(f_*|\vec{x}_*, \boldsymbol{D}) = N(f_*\,; \mu_{f|D}\,, K_{f|D}\,(\vec{x}_*, \vec{x}_*)\,)$$

where

$$\mu_{f|D}\,(\vec{x}) \coloneqq \mu(\vec{x}) + K(\vec{x}, X)\boldsymbol{K}^{-1}(f - \mu(X))$$

$$K_{f|D}\,(\vec{x}\,, \vec{x}\,') \coloneqq K(\vec{x}\,, \vec{x}') - K(\vec{x}, X)\boldsymbol{K}^{-1}K(X, \vec{x})$$

$$\boldsymbol{K} = \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$$

The posterior mean

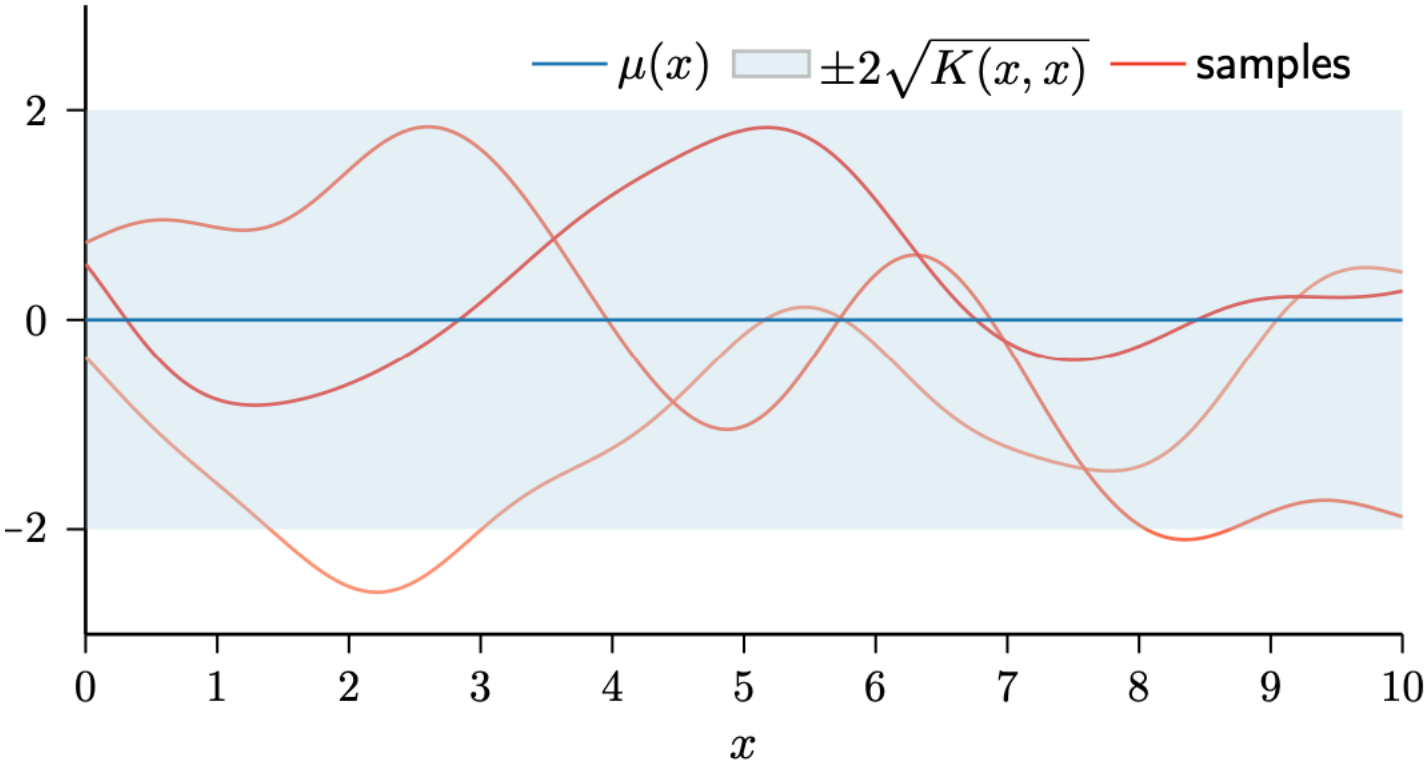
One way to understand the posterior mean function $\mu_{f|D}(\vec{x})$ is as a correction to the prior mean consisting of a weighted combination of kernel functions, one for each training data point:

$$\mu_{f|D}(\vec{x}) \coloneqq \mu(\vec{x}) + K(\vec{x}, X)K^{-1}(f - \mu(X))$$

$$= \mu(\vec{x}) + \sum_{i=1}^{N} \alpha_i K\left(\vec{x}^{(i)}, \vec{x}\right)$$

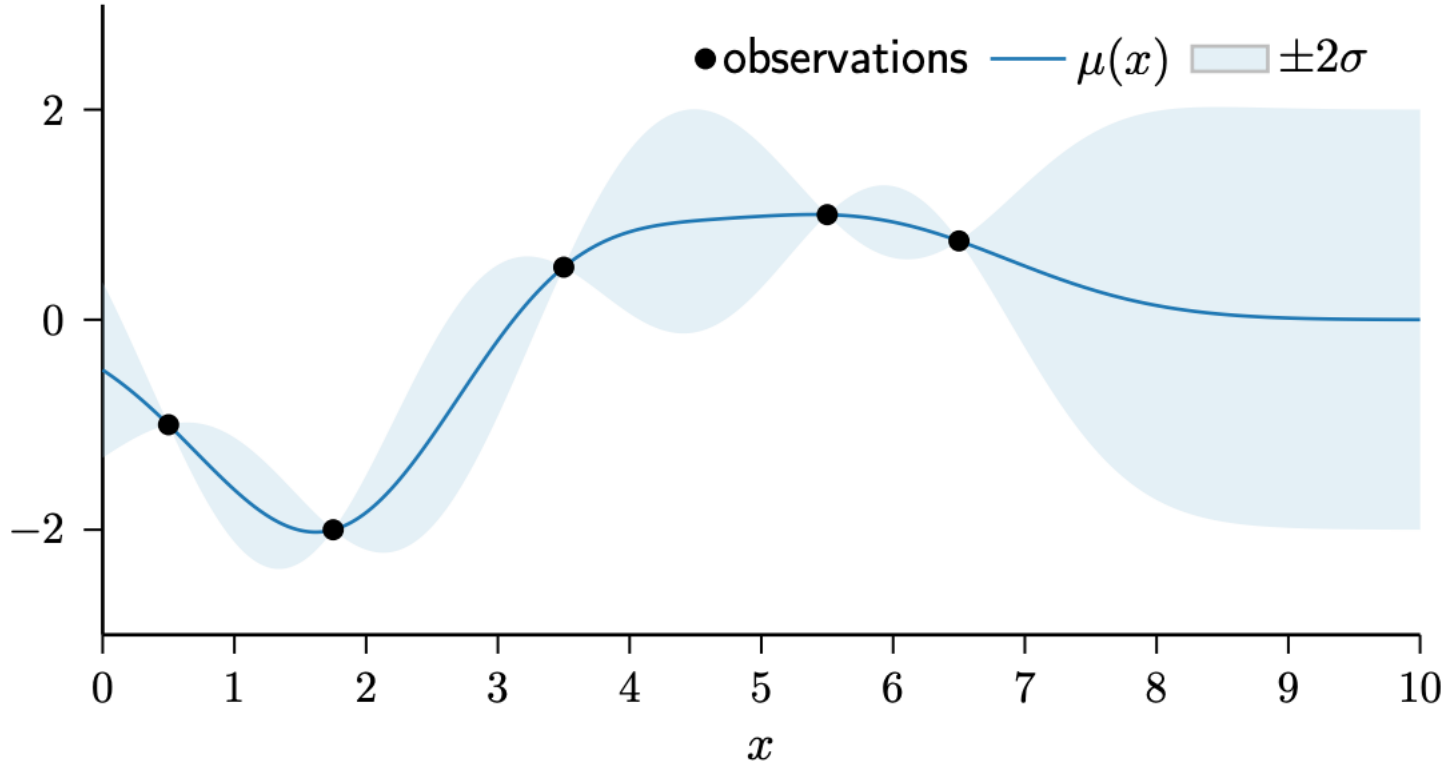
where $\alpha_i = K^{-1}\left(f\left(\vec{x}^{(i)}\right) - \mu\left(\vec{x}^{(i)}\right)\right)$

Prior

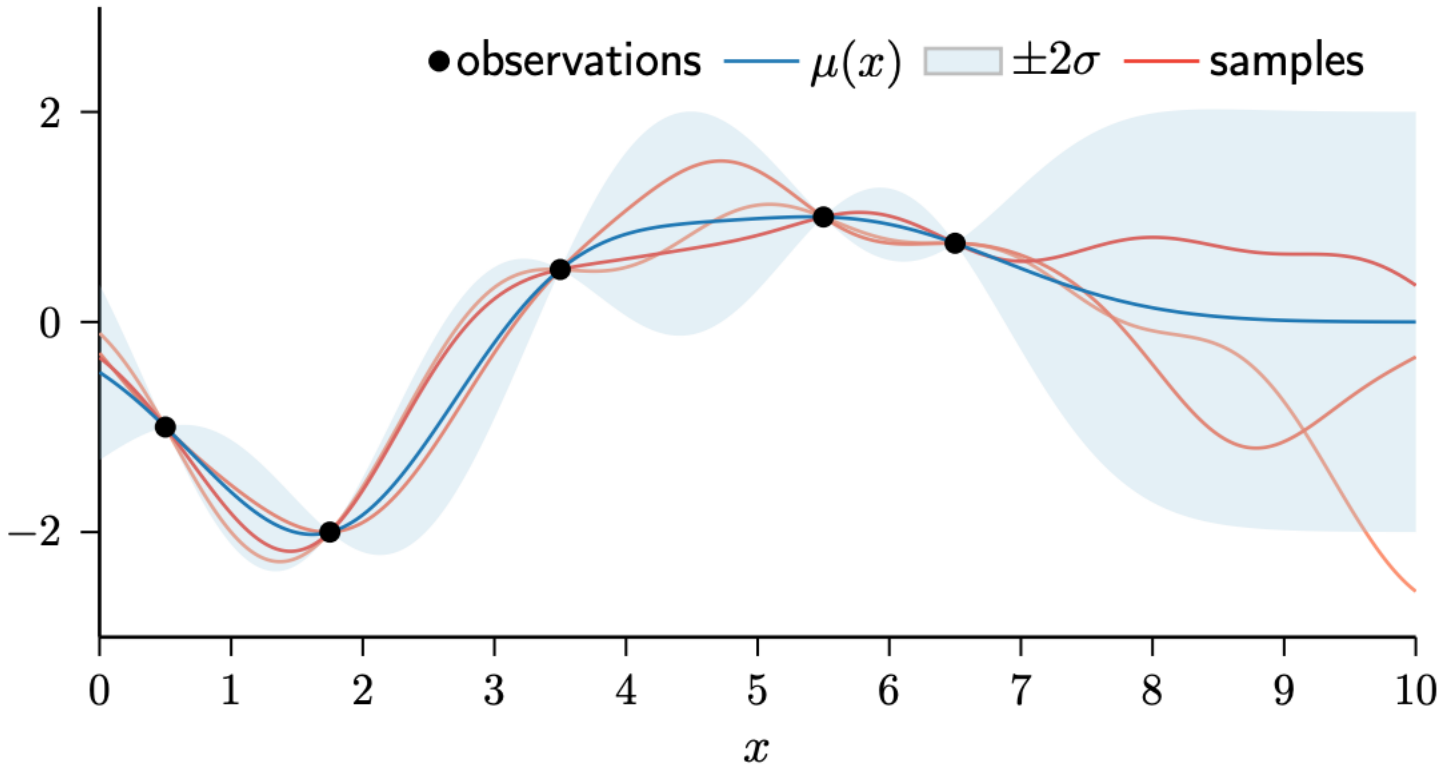


$$K(\vec{x}, \vec{x}') = \exp\left(-\frac{1}{2}\|\vec{x} - \vec{x}'\|^2\right)$$

Posterior example



Posterior: Sampling



Dealing with noise

So far, we have assumed we can **sample** the function f exactly, which is uncommon in regression settings. How do we deal with observation noise?

The same way we did with Bayesian linear regression.

We must create a **model** for our observations given the **latent** function.
To begin, we will choose the simple iid, zero-mean additive Gaussian noise model:

$$y(\vec{x}) = f(\vec{x}) + \epsilon$$

$$\epsilon|\vec{x} \sim N(\epsilon; 0, \sigma^2)$$

So $\vec{y}|\vec{f} \sim N(\vec{y}; \boldsymbol{f}, \sigma^2 I)$

$$\boldsymbol{f} := f(X)$$

Noisy posterior

To derive the posterior given noisy observations **D**, we again write the joint distribution between the training function values $\vec{y}$ and the test function values f_*

$$p(\vec{y}, f_*) = N\left(\begin{bmatrix}\vec{\boldsymbol{y}} \\ f_*\end{bmatrix}; \begin{bmatrix}\boldsymbol{\mu}(\boldsymbol{X}) \\ \mu(f_*)\end{bmatrix}, \begin{bmatrix}\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \boldsymbol{\sigma^2 I} & \boldsymbol{K}(\boldsymbol{X},\vec{\boldsymbol{x}}_*) \\ \boldsymbol{K}(\vec{\boldsymbol{x}}_*,\boldsymbol{X}) & \boldsymbol{K}(\vec{\boldsymbol{x}}_*,\vec{\boldsymbol{x}}_*)\end{bmatrix}\right)$$

condition as before

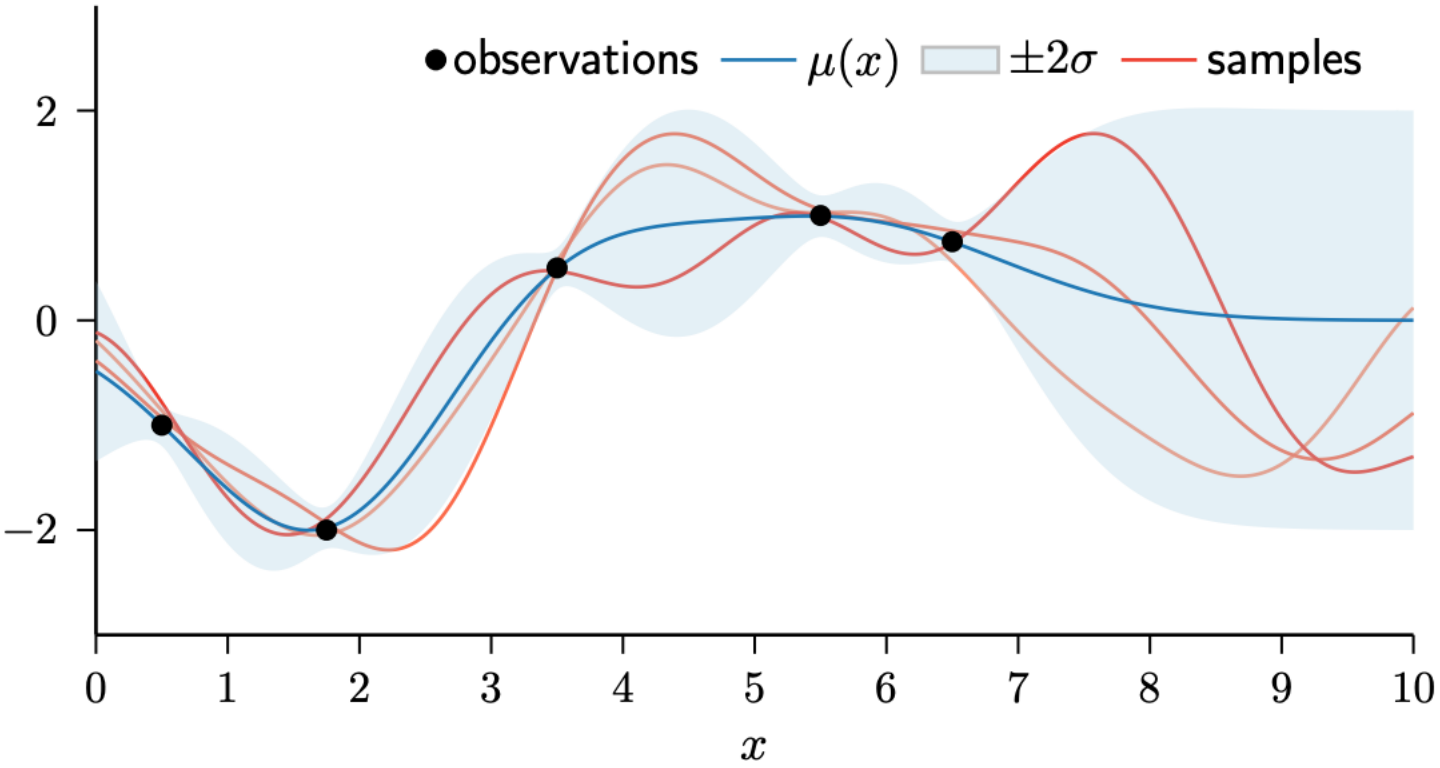
$$p(f_*|\vec{x}_*, \boldsymbol{D}) = N(f_*\,; \mu_{f|D}\,, K_{f|D}\,(\vec{x}_*, \vec{x}_*)\,)$$

where

$$\mu_{f|D}\,(\vec{x}) \coloneqq \mu(\vec{x}) + K(\vec{x}, X)\left(\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \boldsymbol{\sigma^2 I}\right)^{-1}(f - \mu(X))$$

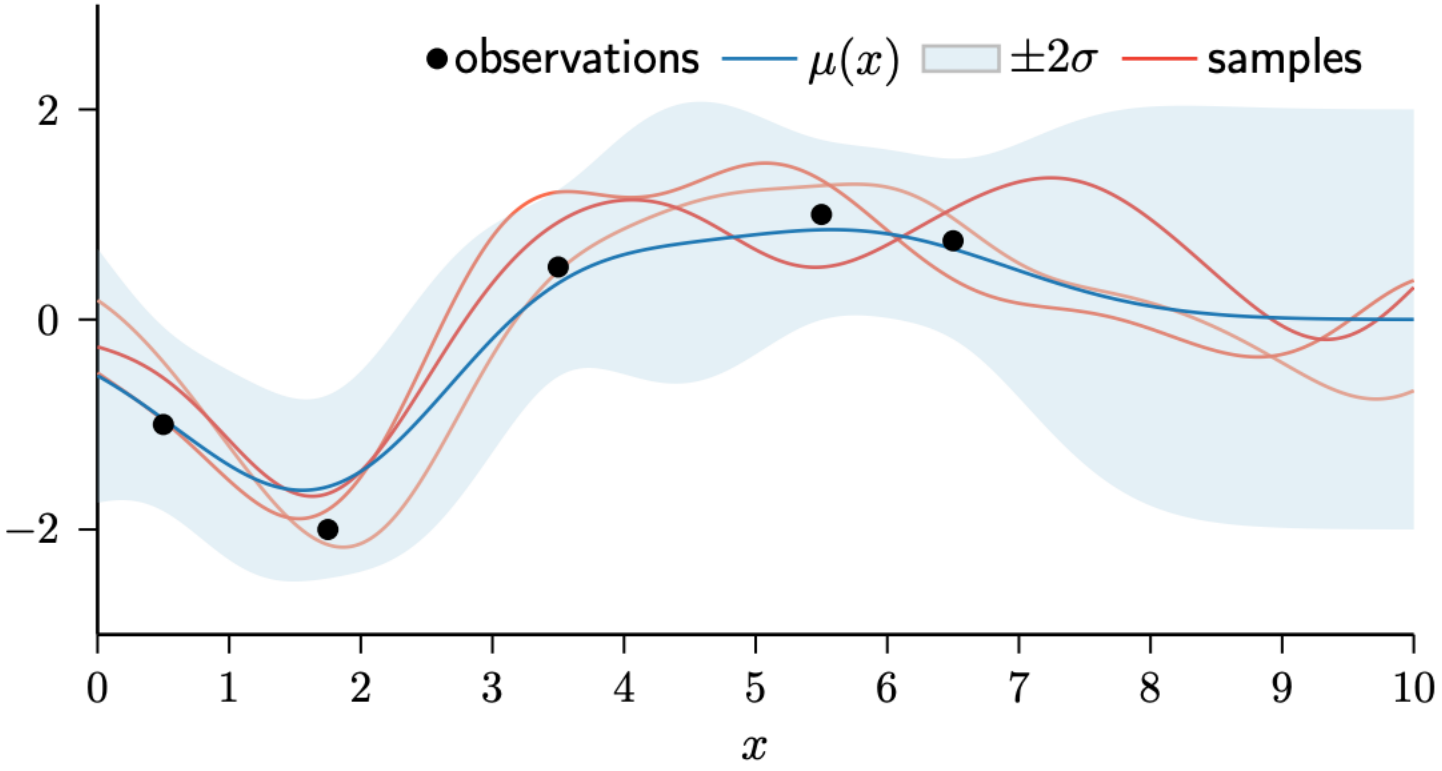
$$K_{f|D}\,(\vec{x}\,, \vec{x}\,') \coloneqq K(\vec{x}\,, \vec{x}') - K(\vec{x}, X)\left(\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \boldsymbol{\sigma^2 I}\right)^{-1} K(X, \vec{x})$$

Noisy posterior: Sampling



$\sigma = 0.1$

Noisy posterior: Sampling



$\sigma = 0.5$

Hyperparameters are important

$$K(\vec{x}, \vec{x}') = \lambda^2 \exp\left(-\frac{1}{2l^2}\|\vec{x} - \vec{x}'\|^2\right)$$

http://gaussianprocess.org/gpml/chapters/RW2.pdf

Textbooks:

[Bishop]: Chapter 6
[Hastie]: 5.8, 6.7,12.3

C. E. Rasmussen & C. K. I. Williams,
Gaussian Processes for Machine Learning", the MIT Press, 2006.
Available from http://www.GaussianProcess.org/gpml Includes a good Matlab tool

References:

- John Shawe-Taylor and Nello Cristianini, Kernel Methods for Pattern Analysis
- Christopher Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery 2, 121–167 (1998).

Other references:

- Aronszajn, Theory of reproducing kernels. Transactions of the American Mathematical Society, 686, 337-404, 1950.
- Felipe Cucker and Steve Smale, On the mathematical foundations of learning. Bulletin of the American Mathematical Society, 2002.
- Teo Evgeniou, Massimo Pontil and Tomaso Poggio, Regularization Networks and Support Vector Machines Advances in Computational Mathematics, 2000.

Book: Radial Basis Functions-Theory and Implementations, Martin D. Buhmann, Cambridge University Press https://doi.org/10.1017/CBO9780511543241

A PhD thesis: **Radial Basis Functions:** Biomedical Applications and Parallelization https://dc.uwm.edu/cgi/viewcontent.cgi?article=2387&context=etd

There are two ways of producing an RBF model in Matlab- one is to do it explicitly yourself, the other is to use Matlab's built-in routines using the "Neural Network Toolbox". If you're using the toolbox, then the OLS is built into the newrb command.