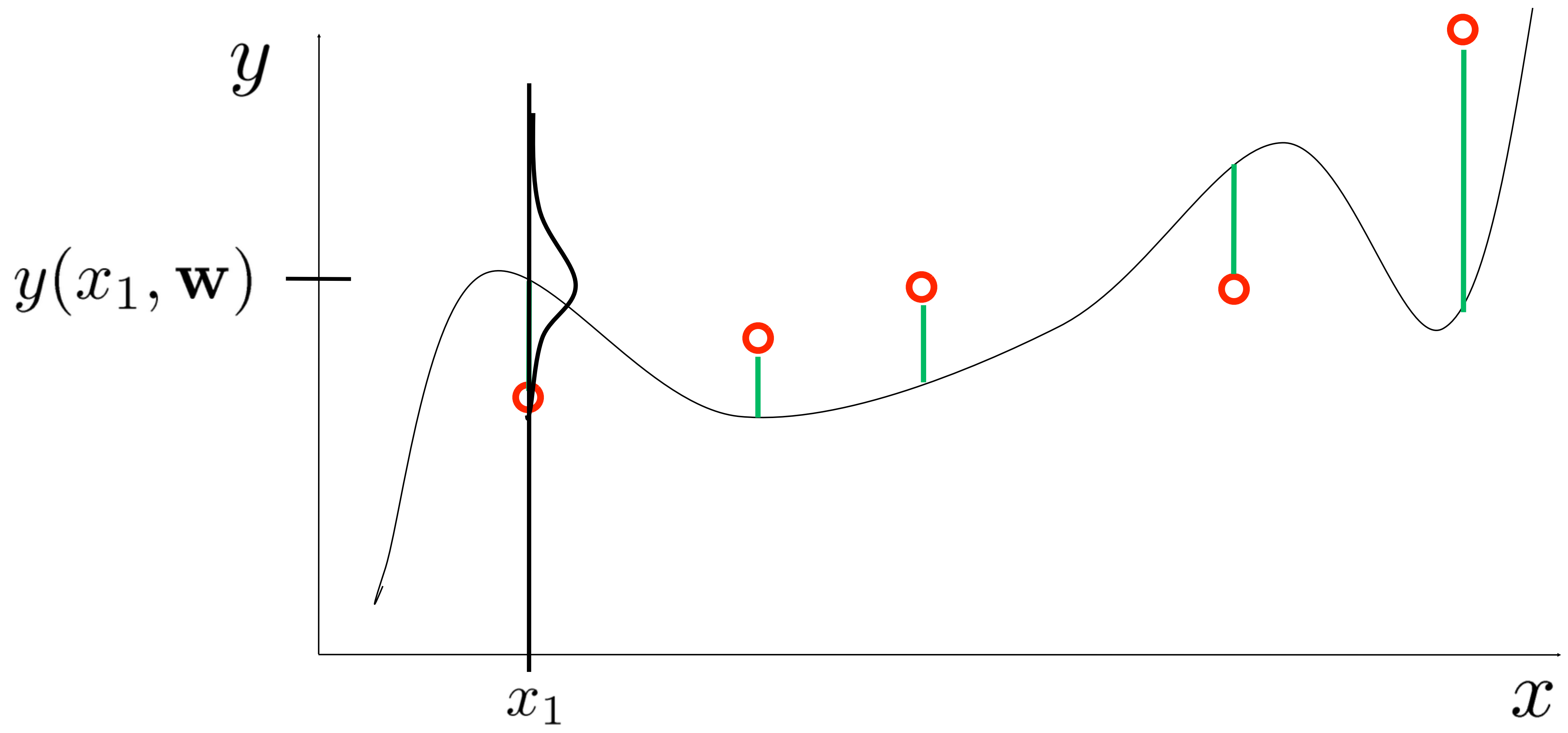
2. Regression (cont.)

Regression with MLE (Rep.)

Assume that y is affected by Gaussian noise :

$$t = f(x, \mathbf{w}) + \epsilon \qquad \text{where} \qquad \epsilon \rightsquigarrow \mathcal{N}(.; 0, \sigma^2)$$

Thus, we have $p(t \mid x, \mathbf{w}, \sigma) = \mathcal{N}(t; f(x, \mathbf{w}), \sigma^2)$



Maximum A-Posteriori Estimation

So far, we searched for parameters $\mathbf{w}$, that maximize the data likelihood. Now, we assume a Gaussian ***prior***:

$$p(\mathbf{w} \mid \sigma_2) = \mathcal{N}(\mathbf{w}; \mathbf{0}, \sigma_2 I)$$

Using this, we can compute the ***posterior*** (Bayes):

$$\underbrace{p(\mathbf{w} \mid \mathbf{x}, \mathbf{t})}_{\text{Posterior}} \propto \underbrace{p(\mathbf{t} \mid \mathbf{w}, \mathbf{x})}_{\text{Likelihood}} \underbrace{p(\mathbf{w})}_{\text{Prior}}$$

"Maximum A-Posteriori Estimation (MAP)"

Maximum A-Posteriori Estimation

So far, we searched for parameters $\mathbf{w}$, that maximize the data likelihood. Now, we assume a Gaussian ***prior***:

$$p(\mathbf{w} \mid \sigma_2) = \mathcal{N}(\mathbf{w}; \mathbf{0}, \sigma_2 I)$$

Using this, we can compute the ***posterior*** (Bayes):

$$p(\mathbf{w} \mid x, \mathbf{t}, \sigma_1, \sigma_2) \propto p(t \mid x, \mathbf{w}, \sigma_1) p(\mathbf{w} \mid \sigma_2)$$

strictly: $$p(\mathbf{w} \mid x, \mathbf{t}, \sigma_1, \sigma_2) = \frac{p(t \mid x, \mathbf{w}, \sigma_1) p(\mathbf{w} \mid \sigma_2)}{\int p(t \mid x, \mathbf{w}, \sigma_1) p(\mathbf{w} \mid \sigma_2) d\mathbf{w}}$$

but the denominator is independent of $\mathbf{w}$ and we want to maximize p.

Maximum A-Posteriori Estimation

$$\ln p(\mathbf{w} \mid x, \mathbf{t}, \sigma_1, \sigma_2) \propto \underbrace{\ln p(t \mid x, \mathbf{w}, \sigma_1)}_{\text{const.} - \frac{1}{2\sigma_1^2}\sum_{i=1}^{N}(\mathbf{w}^T\phi(x) - t_i)^2} + \underbrace{\ln p(\mathbf{w} \mid \sigma_2)}_{\text{const.} - \frac{1}{2\sigma_2^2}\mathbf{w}^T\mathbf{w}}$$

$$\propto -\frac{1}{2\sigma_1^2}\left(\sum_{i=1}^{N}(\mathbf{w}^T\phi(x) - t_i)^2 + \frac{\sigma_1^2}{\sigma_2^2}\mathbf{w}^T\mathbf{w}\right)$$

This is equal to the regularized error minimization.

The MAP Estimate corresponds to a regularized error minimization where $\lambda = (\sigma_1 / \sigma_2)^2$

Summary: MAP Estimation

To summarize, we have the following optimization problem:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\mathbf{w}^T\phi(\mathbf{x}_n) - t_n)^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \qquad \phi(\mathbf{x}_n) \in \mathbb{R}^M$$

The same in vector notation:

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\Phi^T\Phi\mathbf{w} - \mathbf{w}\Phi^T\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \qquad \mathbf{t} \in \mathbb{R}^N$$

$$\Phi = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \dots & \phi_{M-1}(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \dots & \phi_{M-1}(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_N) & \phi_1(x_N) & \dots & \phi_{M-1}(x_N) \end{pmatrix} \in \mathbb{R}^{N\times M}$$

“Feature Matrix”

Summary: MAP Estimation

To summarize, we have the following optimization problem:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\mathbf{w}^T\phi(\mathbf{x}_n) - t_n)^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \qquad \phi(\mathbf{x}_n) \in \mathbb{R}^M$$

The same in vector notation:

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\Phi^T\Phi\mathbf{w} - \mathbf{w}\Phi^T\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \quad \mathbf{t} \in \mathbb{R}^N$$

And the solution is

$$\mathbf{w}^* = (\lambda I_M + \Phi^T\Phi)^{-1}\Phi^T\mathbf{t}$$

Identity matrix of size M by M

MLE And MAP

- The benefit of MAP over MLE is that prediction is less sensitive to **overfitting,** i.e. even if there is only little data the model predicts well.
- This is achieved by using **prior information,** i.e. model assumptions that are not based on any observations (= data)
- But: both methods only give the **most likely** model, there is no notion of **uncertainty** yet

Idea 1: Find a **distribution** over model parameters (“parameter posterior”)

MLE And MAP

- The benefit of MAP over MLE is that prediction is less sensitive to **overfitting,** i.e. even if there is only little data the model predicts well.
- This is achieved by using **prior information,** i.e. model assumptions that are not based on any observations (= data)
- But: both methods only give the **most likely** model, there is no notion of **uncertainty** yet

Idea 1: Find a **distribution** over model parameters

Idea 2: Use that distribution to estimate **prediction uncertainty** ("predictive distribution")

When Bayes Meets Gauß

Theorem: If we are given this:

I. $$p(\mathbf{x}) = \mathcal{N}(\mathbf{x} \mid \mu, \Sigma_1)$$

II. $$p(\mathbf{y} \mid \mathbf{x}) = \mathcal{N}(\mathbf{y} \mid A\mathbf{x} + \mathbf{b}, \Sigma_2)$$

linear dependency on $\mathbf{x}$

Then it follows (properties of Gaussians):

III. $$p(\mathbf{y}) = \mathcal{N}(\mathbf{y} \mid A\mu + \mathbf{b}, \Sigma_2 + A\Sigma_1 A^T)$$

IV. $$p(\mathbf{x} \mid \mathbf{y}) = \mathcal{N}(\mathbf{x} \mid \Sigma(A^T\Sigma_2^{-1}(\mathbf{y} - \mathbf{b}) + \Sigma_1^{-1}\mu), \Sigma)$$

where

$$\Sigma = (\Sigma_1^{-1} + A^T\Sigma_2^{-1}A)^{-1}$$

See Bishop's book for the proof!

"Linear Gaussian Model"

When Bayes Meets Gauß

Thus: When using the Bayesian approach, we can do even more than MLE and MAP by using these formulae.

This means:

If the prior and the likelihood are Gaussian then the **posterior** and the **normalizer** are also Gaussian and we can compute them in closed form.

This gives us a natural way to compute uncertainty!

The Posterior Distribution

Remember Bayes Rule:

$$\underbrace{p(\mathbf{w} \mid \mathbf{x}, \mathbf{t})}_{\text{Posterior}} \propto \underbrace{p(\mathbf{t} \mid \mathbf{w}, \mathbf{x})}_{\text{Likelihood}} \underbrace{p(\mathbf{w})}_{\text{Prior}}$$

With our theorem, we can compute the posterior in **closed form** (and not just its maximum)!

The posterior is also a Gaussian and its **mean** is the MAP solution.

The Posterior Distribution

We have $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}; \mathbf{0}, \sigma_2^2 I_M)$

and $p(\mathbf{t} \mid \mathbf{w}, \mathbf{x}) = \mathcal{N}(\mathbf{t}; \Phi\mathbf{w}, \sigma_1^2 I_N)$

From this and IV. we get the **posterior covariance:**

$$\Sigma = (\sigma_2^{-2} I_M + \sigma_1^{-2} \Phi^T \Phi)^{-1}$$

$$= \sigma_1^2 \left(\frac{\sigma_1^2}{\sigma_2^2} I_M + \Phi^T \Phi \right)^{-1}$$

and the **mean:** $\boldsymbol{\mu} = \sigma_1^{-2} \Sigma \Phi^T \mathbf{t}$

So the entire posterior distribution is

$$p(\mathbf{w} \mid \mathbf{t}, \mathbf{x}) = \mathcal{N}(\mathbf{w}; \boldsymbol{\mu}, \Sigma)$$

The Predictive Distribution

We obtain the **predictive distribution** by integrating over all possible model parameters:

$$p(t \mid x, \mathbf{t}, \mathbf{x}) = \int \underbrace{p(t \mid x, \mathbf{w})}_{\text{New data likelihood}} \underbrace{p(\mathbf{w} \mid \mathbf{x}, \mathbf{t})}_{\text{Parameter posterior}} d\mathbf{w}$$

This distribution can be computed in closed form, because both terms on the RHS are Gaussian.

From above we have $\boxed{p(\mathbf{w} \mid \mathbf{t}, \mathbf{x})} = \mathcal{N}(\mathbf{w}; \boldsymbol{\mu}, \Sigma)$

where $\boldsymbol{\mu} = \sigma_1^{-2} \Sigma \Phi^T \mathbf{t}$

and $\Sigma = \sigma_1^2 (\frac{\sigma_1^2}{\sigma_2^2} I_M + \Phi^T \Phi)^{-1}$

The Predictive Distribution

We obtain the **predictive distribution** by integrating over all possible model parameters:

$$p(t \mid x, \mathbf{t}, \mathbf{x}) = \int \underbrace{p(t \mid x, \mathbf{w})}_{\text{New data likelihood}} \underbrace{p(\mathbf{w} \mid \mathbf{x}, \mathbf{t})}_{\text{Parameter posterior}} d\mathbf{w}$$

This distribution can be computed in closed form, because both terms on the RHS are Gaussian.

From above we have $\boxed{p(\mathbf{w} \mid \mathbf{t}, \mathbf{x})} = \mathcal{N}(\mathbf{w}; \boldsymbol{\mu}, \Sigma)$

where $\boldsymbol{\mu} = \sigma_1^{-2} \Sigma \Phi^T \mathbf{t}$

and $\Sigma = \sigma_1^2 (\frac{\sigma_1^2}{\sigma_2^2} I_M + \Phi^T \Phi)^{-1}$ $\Rightarrow \boldsymbol{\mu} = (\lambda I_M + \Phi^T \Phi)^{-1} \Phi^T \mathbf{t}$

MAP solution

The Predictive Distribution

Using formula III. from above (linear Gaussian),

$$p(t \mid x, \mathbf{t}, \mathbf{x}) = \int p(t \mid x, \mathbf{w}) p(\mathbf{w} \mid \mathbf{x}, \mathbf{t}) d\mathbf{w}$$

$$= \int \mathcal{N}(t; \phi(x)^T \mathbf{w}, \sigma) \mathcal{N}(\mathbf{w}; \boldsymbol{\mu}, \Sigma) d\mathbf{w}$$

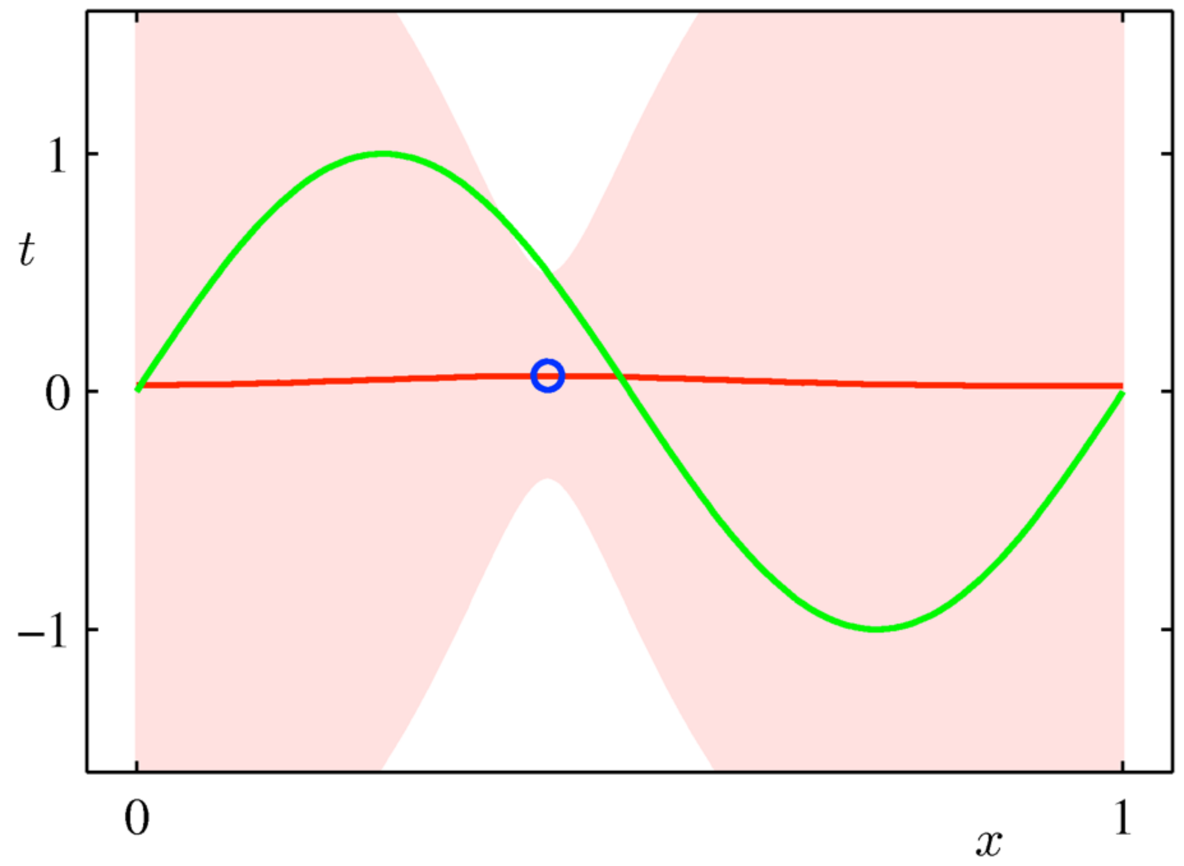
$$= \mathcal{N}(t; \phi(x)^T \boldsymbol{\mu}, \sigma_N^2(x))$$

where

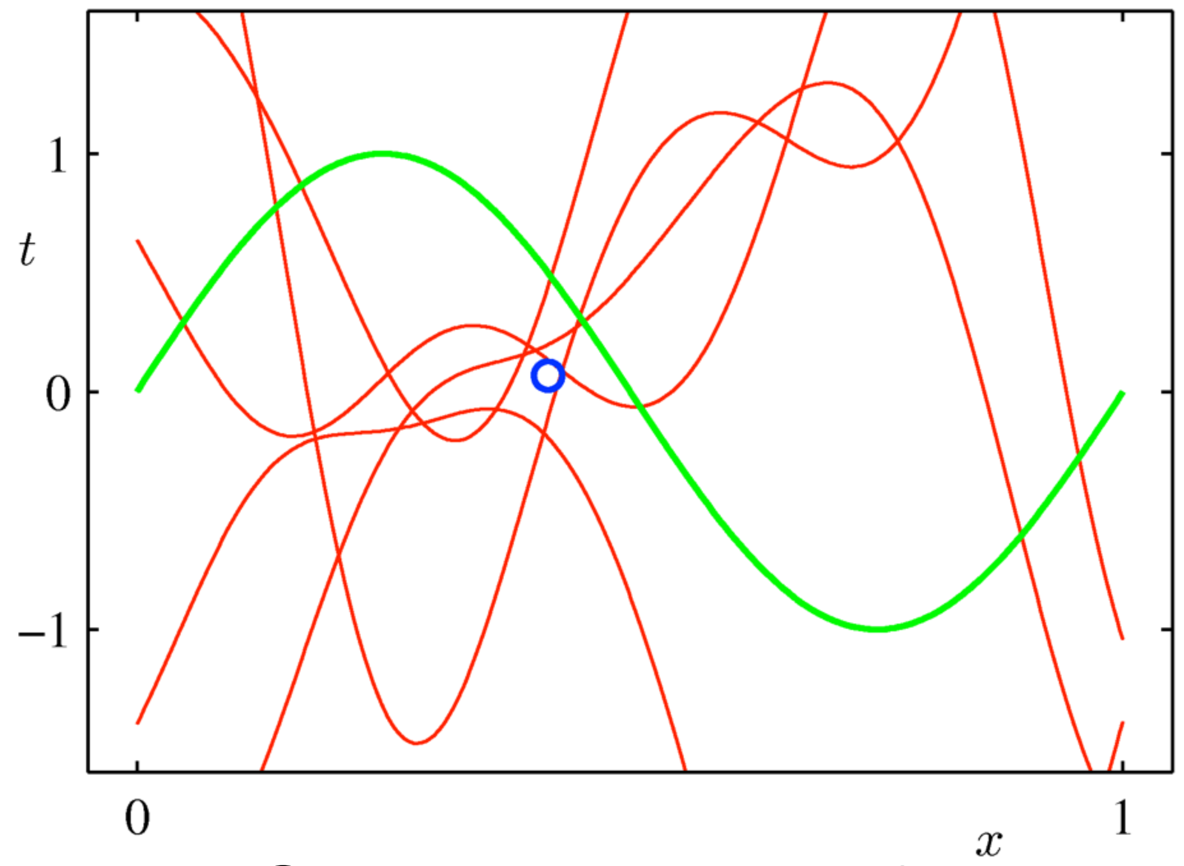
$$\sigma_N^2(x) = \sigma^2 + \phi(x)^T \Sigma \phi(x)$$

The Predictive Distribution (2)

- Example: Sinusoidal data, 9 Gaussian basis functions, 1 data point



The predictive distribution

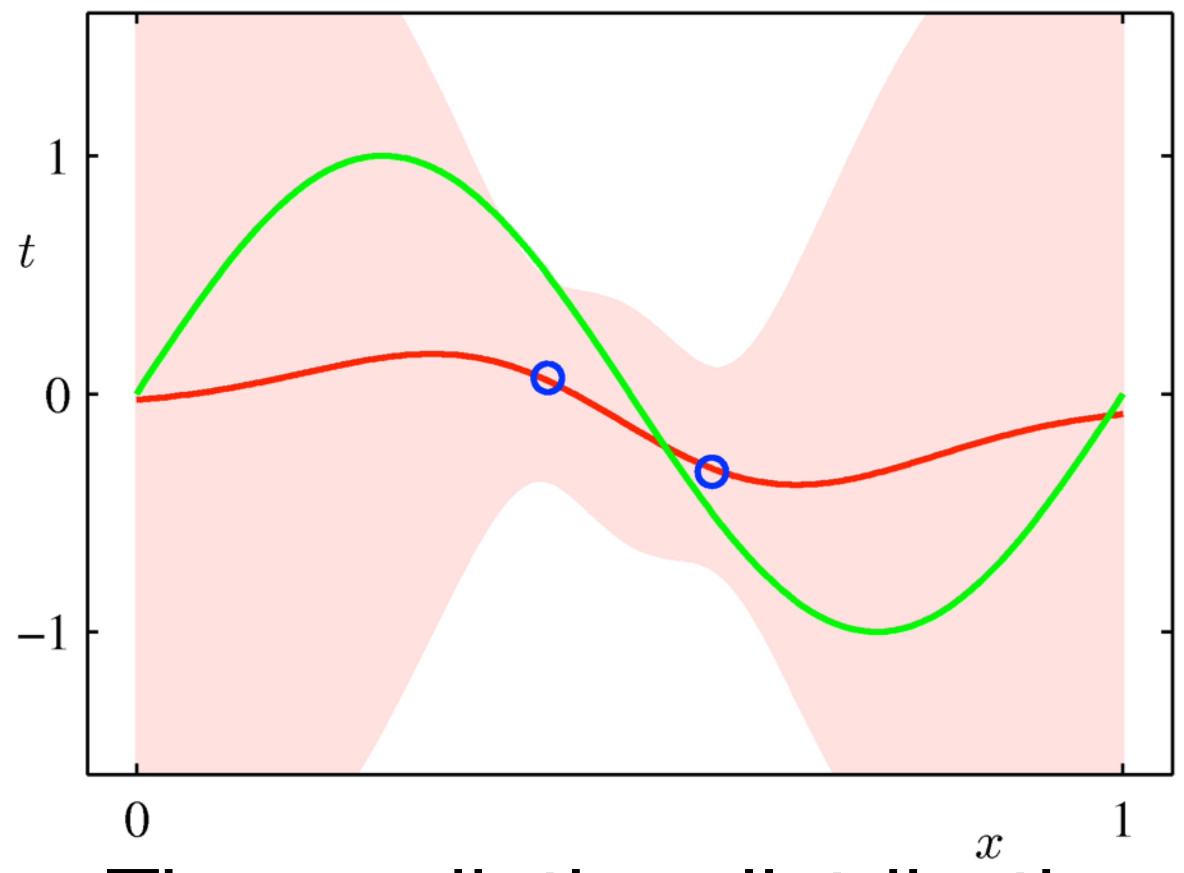


Some samples from the posterior

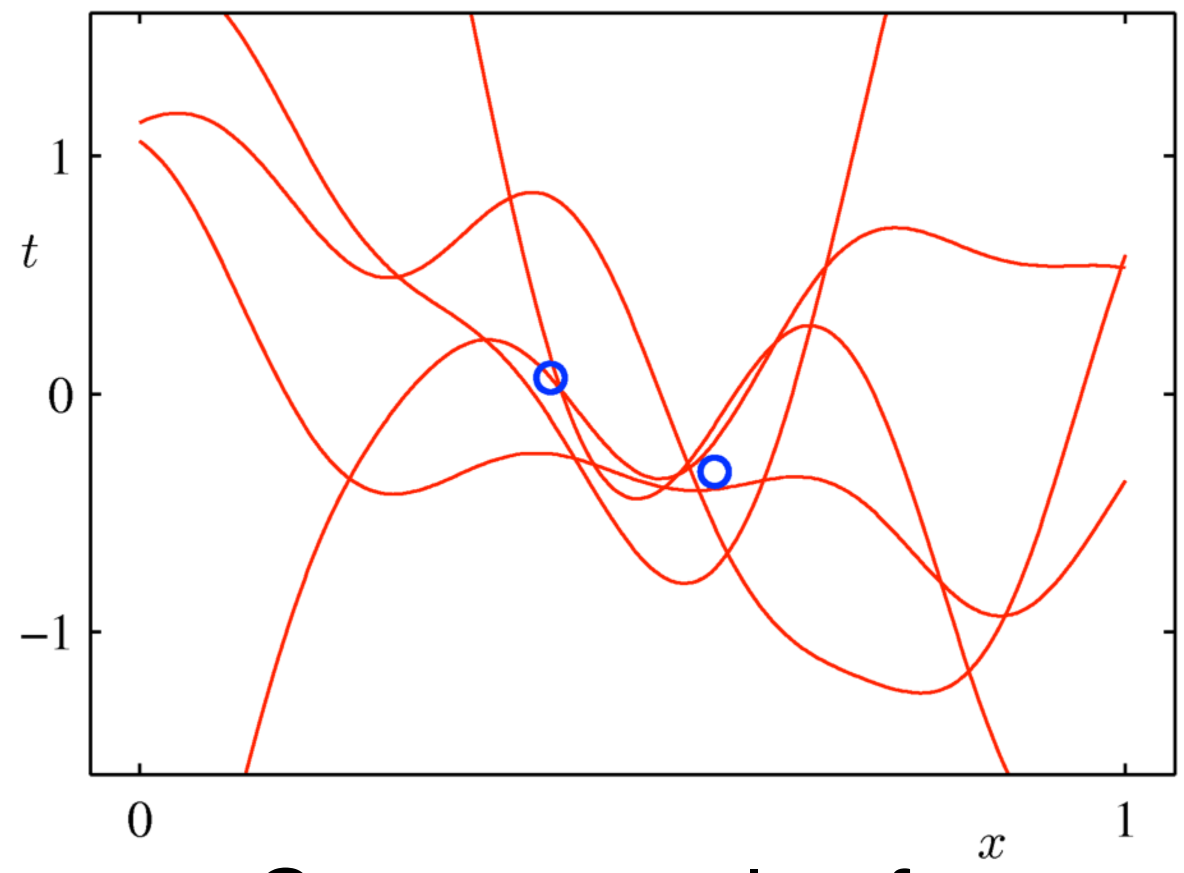
From: C.M. Bishop

Predictive Distribution (3)

- Example: Sinusoidal data, 9 Gaussian basis functions, 2 data points



The predictive distribution

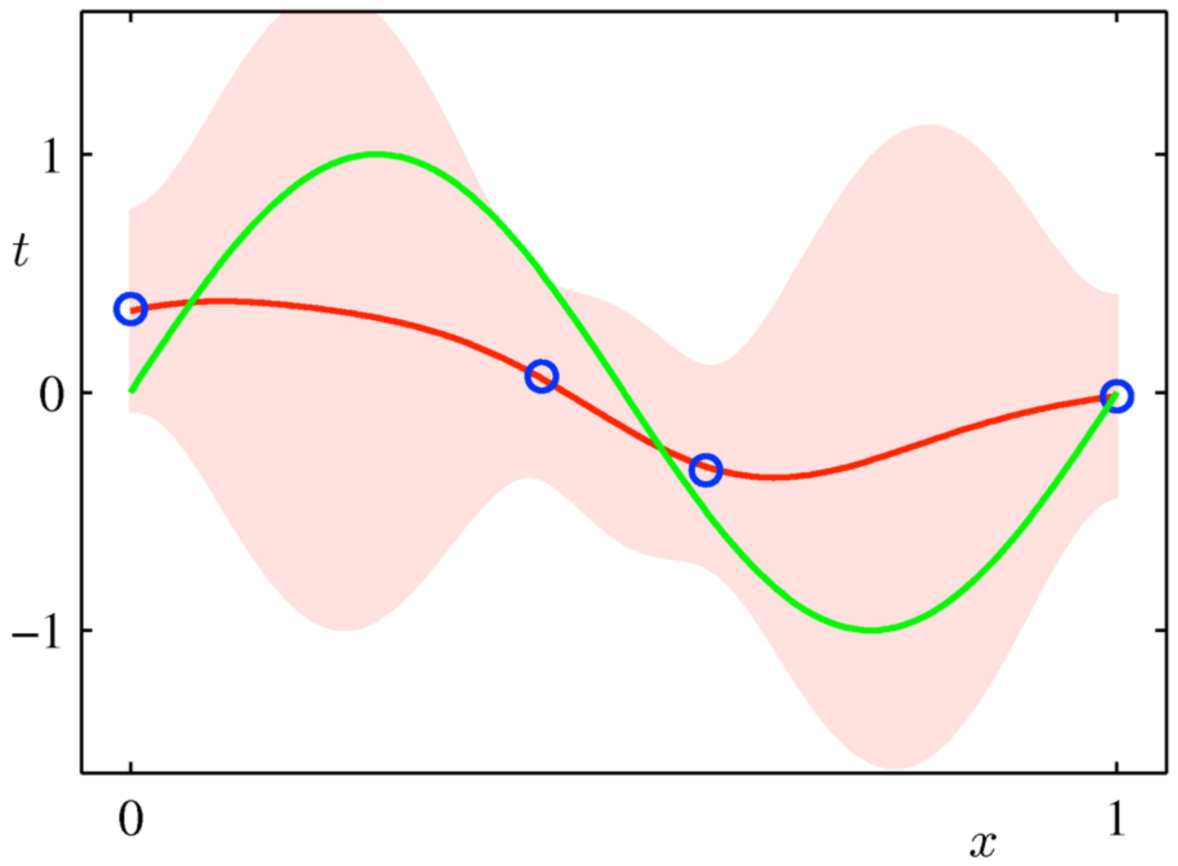


Some samples from the posterior

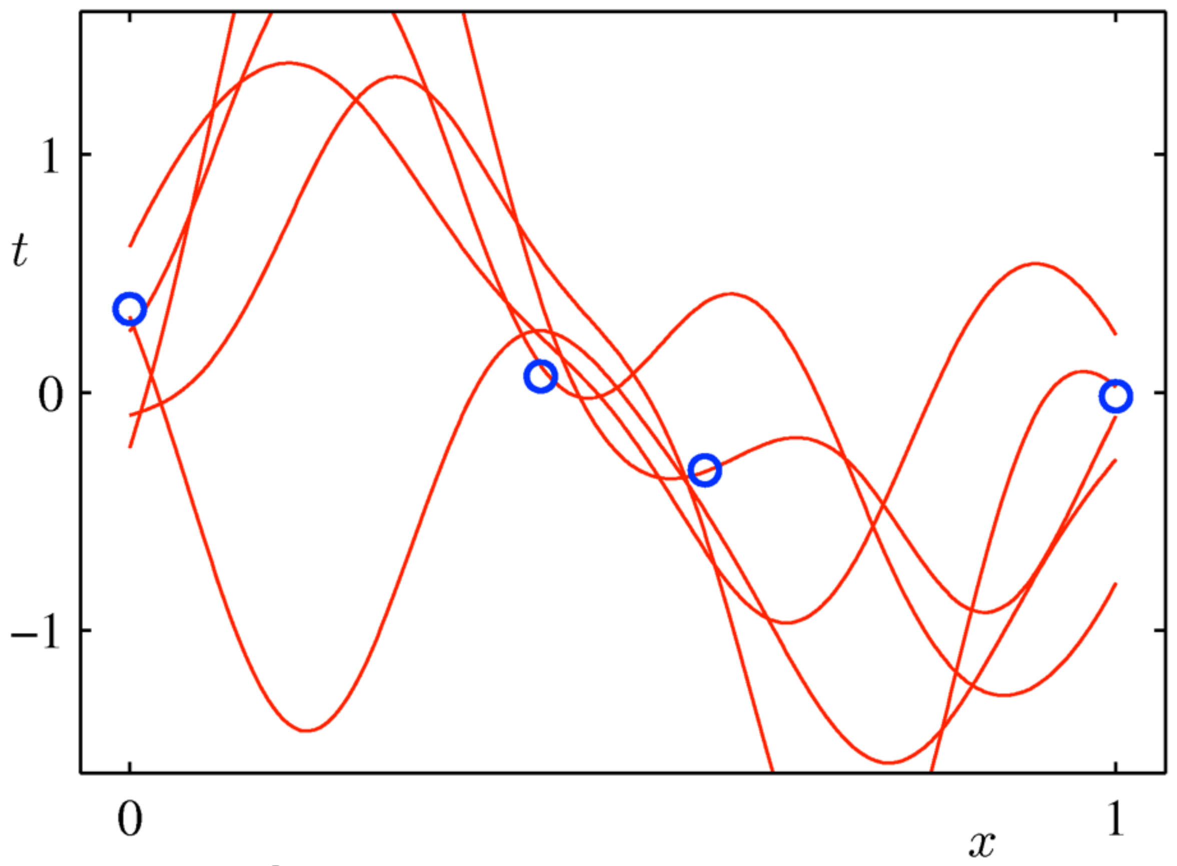
From: C.M. Bishop

Predictive Distribution (4)

- Example: Sinusoidal data, 9 Gaussian basis functions, 4 data points



The predictive distribution

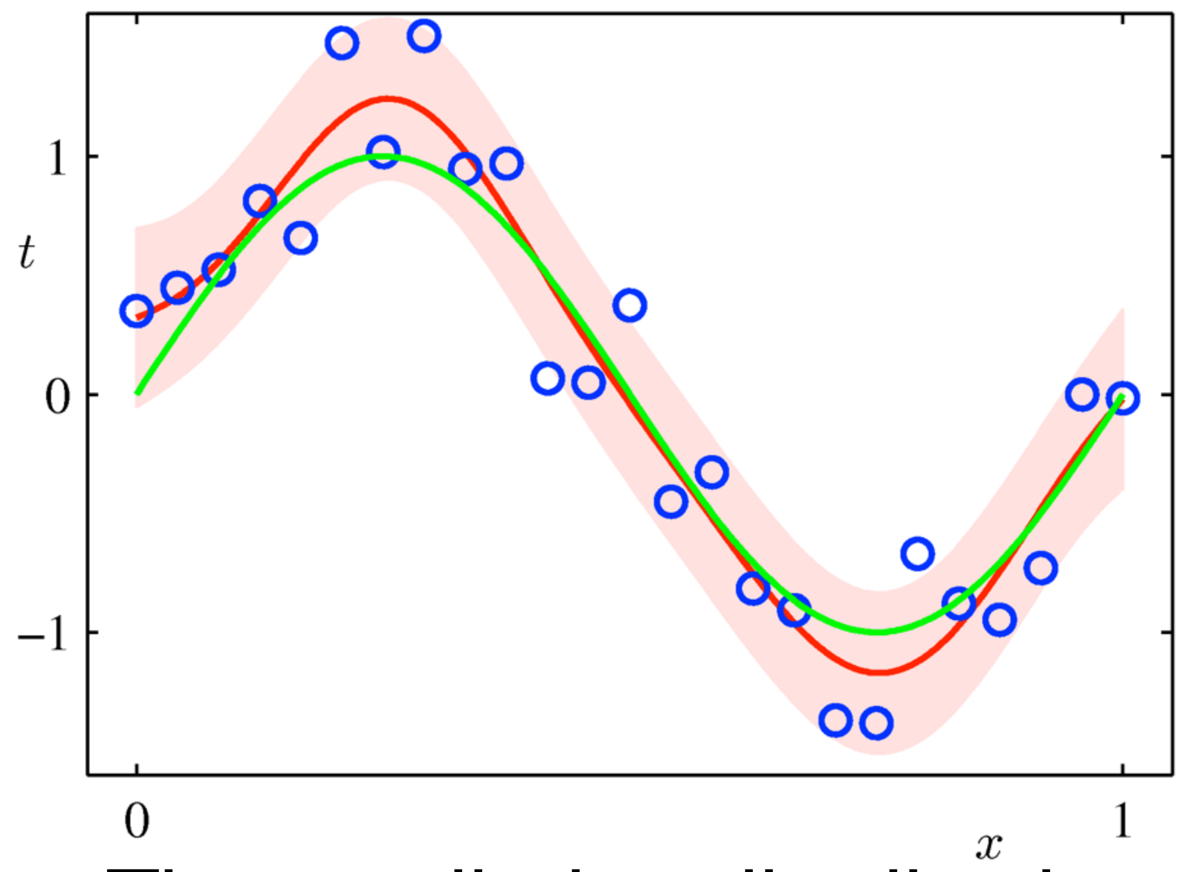


Some samples from the posterior

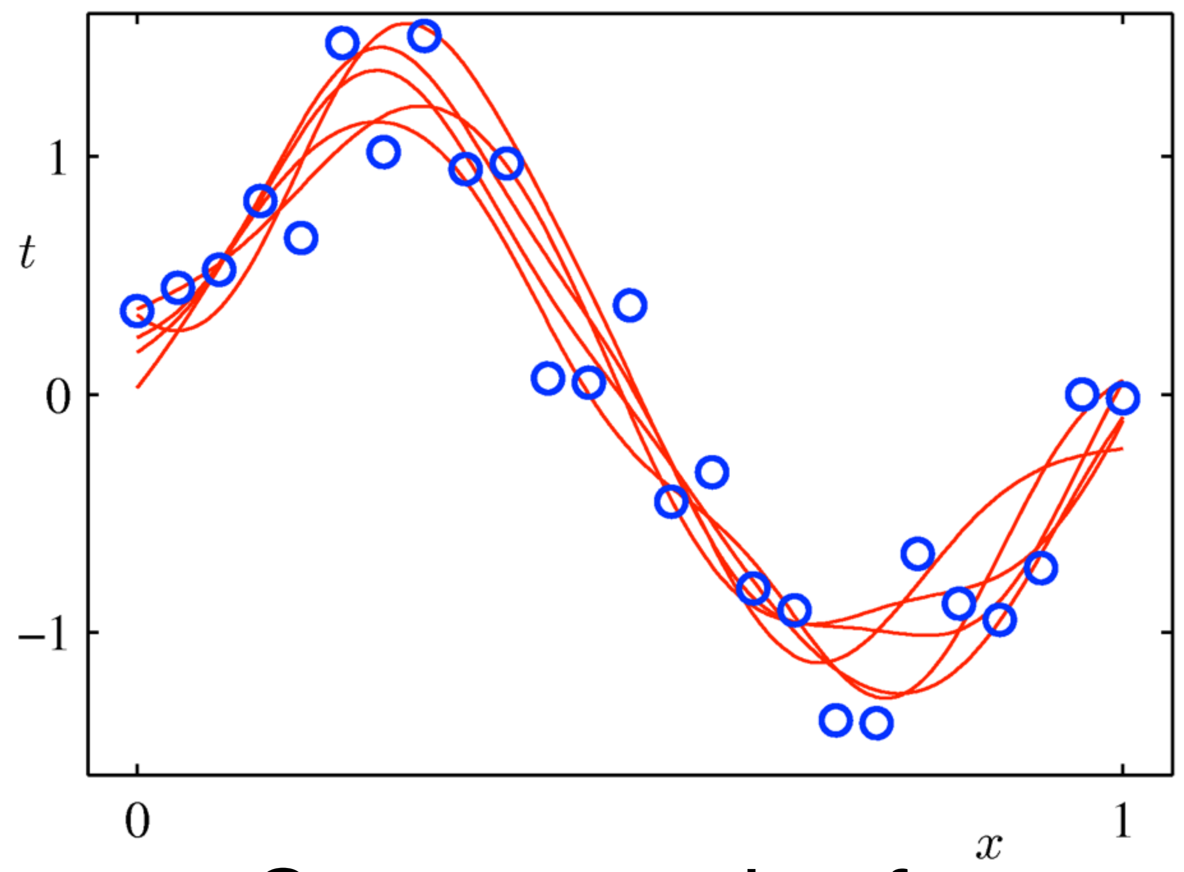
From: C.M. Bishop

Predictive Distribution (5)

- Example: Sinusoidal data, 9 Gaussian basis functions, 25 data points



The predictive distribution



Some samples from the posterior

From: C.M. Bishop

Summary

- Regression can be expressed as a **least-squares** problem
- To avoid overfitting, we need to introduce a **regularisation term** with an additional parameter λ
- Regression **without** regularisation is equivalent to Maximum Likelihood Estimation
- Regression **with** regularisation is Maximum A-Posteriori
- When using Gaussian priors (and Gaussian noise), all computations can be done **analytically**
- This gives a closed form of the **parameter posterior** and the **predictive distribution**

3. Kernel Methods

Motivation

- Usually learning algorithms assume that some kind of feature function is given
- Reasoning is then done on a feature vector of a given (finite) length
- But: some objects are hard to represent with a fixed-size feature vector, e.g. text documents, molecular structures, evolutionary trees
- Idea: use a way of measuring similarity **without** the need of features, e.g. the edit distance for strings
- This we will call a **kernel function**

Dual Representation

Many problems can be expressed using a **dual** formulation. Example (linear regression):

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\mathbf{w}^T\phi(\mathbf{x}_n) - t_n)^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \qquad \phi(\mathbf{x}_n) \in \mathbb{R}^M$$

Dual Representation

Many problems can be expressed using a **dual** formulation. Example (linear regression):

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\mathbf{w}^T\phi(\mathbf{x}_n) - t_n)^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \qquad \phi(\mathbf{x}_n) \in \mathbb{R}^M$$

if we write this in vector form, we get

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\Phi^T\Phi\mathbf{w} - \mathbf{w}\Phi^T\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \qquad \mathbf{t} \in \mathbb{R}^N$$

$$\Phi \in \mathbb{R}^{N\times M}$$

Dual Representation

Many problems can be expressed using a **dual** formulation. Example (linear regression):

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\mathbf{w}^T\phi(\mathbf{x}_n) - t_n)^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \qquad \phi(\mathbf{x}_n) \in \mathbb{R}^M$$

if we write this in vector form, we get

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\Phi^T\Phi\mathbf{w} - \mathbf{w}\Phi^T\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \qquad \mathbf{t} \in \mathbb{R}^N$$

$$\Phi \in \mathbb{R}^{N\times M}$$

and the solution is

$$\mathbf{w} = (\Phi^T\Phi + \lambda I_M)^{-1}\Phi^T\mathbf{t}$$

Dual Representation

Many problems can be expressed using a **dual** formulation, including linear regression.

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\Phi^T\Phi\mathbf{w} - \mathbf{w}\Phi^T\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

$$\mathbf{w} = (\Phi^T\Phi + \lambda I_M)^{-1}\Phi^T\mathbf{t}$$

However, we can express this result in a different way using the **matrix inversion lemma:**

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}$$

Dual Representation

Many problems can be expressed using a **dual** formulation. Example (linear regression):

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\Phi^T\Phi\mathbf{w} - \mathbf{w}\Phi^T\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

$$\mathbf{w} = (\Phi^T\Phi + \lambda I_M)^{-1}\Phi^T\mathbf{t}$$

However, we can express this result in a different way using the **matrix inversion lemma:**

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}$$

$$\mathbf{w} = \Phi^T(\Phi\Phi^T + \lambda I_N)^{-1}\mathbf{t}$$

Dual Representation

Many problems can be expressed using a **dual** formulation. Example (linear regression):

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\Phi^T\Phi\mathbf{w} - \mathbf{w}\Phi^T\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

$$\mathbf{w} = (\Phi^T\Phi + \lambda I_M)^{-1}\Phi^T\mathbf{t}$$

$$\mathbf{w} = \Phi^T\underbrace{(\Phi\Phi^T + \lambda I_N)^{-1}\mathbf{t}}_{=:\,\mathbf{a}}$$

"Dual Variables"

Dual Representation

Many problems can be expressed using a **dual** formulation. Example (linear regression):

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\Phi^T\Phi\mathbf{w} - \mathbf{w}\Phi^T\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

$$\mathbf{w} = (\Phi^T\Phi + \lambda I_M)^{-1}\Phi^T\mathbf{t}$$

$$\mathbf{w} = \Phi^T\underbrace{(\Phi\Phi^T + \lambda I_N)^{-1}\mathbf{t}}_{=:\,\mathbf{a}}$$

“Dual Variables”

Plugging $\mathbf{w} = \Phi^T\mathbf{a}$ into $J(\mathbf{w})$ gives:

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^T\underbrace{\Phi\Phi^T}_{=:\,K}\Phi\Phi^T\mathbf{a} - \mathbf{a}^T\Phi\Phi^T\mathbf{t} + \mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^T\Phi\Phi^T\mathbf{a}$$

Dual Representation

Many problems can be expressed using a **dual** formulation. Example (linear regression):

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\Phi^T\Phi\mathbf{w} - \mathbf{w}\Phi^T\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^T KK\mathbf{a} - \mathbf{a}^T K\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^T K\mathbf{a} \qquad K = \Phi\Phi^T$$

This is called the **dual formulation.**

Note: $\mathbf{a} \in \mathbb{R}^N \quad \mathbf{w} \in \mathbb{R}^M$

Dual Representation

Many problems can be expressed using a **dual** formulation. Example (linear regression):

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\Phi^T\Phi\mathbf{w} - \mathbf{w}\Phi^T\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^T K K\mathbf{a} - \mathbf{a}^T K\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^T K\mathbf{a}$$

This is called the **dual formulation**.

The solution to the dual problem is:

$$\mathbf{a} = (K + \lambda I_N)^{-1}\mathbf{t}$$

Dual Representation

Many problems can be expressed using a **dual** formulation. Example (linear regression):

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\Phi^T\Phi\mathbf{w} - \mathbf{w}\Phi^T\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^T K K\mathbf{a} - \mathbf{a}^T K\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^T K\mathbf{a}$$

$$\mathbf{a} = (K + \lambda I_N)^{-1}\mathbf{t}$$

This we can use to make **predictions**:

$$f(\mathbf{x}^*) = \mathbf{w}^T\phi(\mathbf{x}^*) = \mathbf{a}^T\Phi\phi(\mathbf{x}^*) = \mathbf{k}(\mathbf{x}^*)^T(K + \lambda I_N)^{-1}\mathbf{t}$$

(now $\mathbf{x^*}$ is unknown and $\mathbf{a}$ is given from training)

Dual Representation

$$f(\mathbf{x}^*) = \mathbf{k}(\mathbf{x}^*)^T (K + \lambda I_N)^{-1} \mathbf{t}$$

where:

$$\mathbf{k}(\mathbf{x}^*) = \begin{pmatrix} \phi(\mathbf{x}_1)^T \phi(\mathbf{x}^*) \\ \vdots \\ \phi(\mathbf{x}_N)^T \phi(\mathbf{x}^*) \end{pmatrix} \quad K = \begin{pmatrix} \phi(\mathbf{x}_1)^T \phi(\mathbf{x}_1) & \dots & \phi(\mathbf{x}_1)^T \phi(\mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ \phi(\mathbf{x}_N)^T \phi(\mathbf{x}_1) & \dots & \phi(\mathbf{x}_N)^T \phi(\mathbf{x}_N) \end{pmatrix}$$

Thus, f is expressed only in terms of **dot products** between different pairs of $\phi(\mathbf{x})$, or in terms of the **kernel function**

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

Representation using the Kernel

$$f(\mathbf{x}^*) = \mathbf{k}(\mathbf{x}^*)^T (K + \lambda I_N)^{-1} \mathbf{t}$$

Now we have to invert a matrix of size $N \times N$, before it was $M \times M$ where $M < N$, but:

By expressing everything with the kernel function, we can deal with very high-dimensional or even **infinite**-dimensional feature spaces!

Idea: Don’t use features at all but simply define a **similarity** function expressed as the kernel!

Constructing Kernels

The straightforward way to define a kernel function is to first find a basis function $\phi(\mathbf{x})$ and to define:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

This means, k is an inner product in some space $\mathcal{H}$, i.e:

1. Symmetry: $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_j), \phi(\mathbf{x}_i) \rangle = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$
2. Linearity: $\langle a(\phi(\mathbf{x}_i) + \mathbf{z}), \phi(\mathbf{x}_j) \rangle = a \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle + a \langle \mathbf{z}, \phi(\mathbf{x}_j) \rangle$
3. Positive definite: $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_i) \rangle \geq 0$, equal if $\phi(\mathbf{x}_i) = \mathbf{0}$

Can we find conditions for k under which there is a (possibly infinite dimensional) basis function into $\mathcal{H}$, where k is an inner product?

Constructing Kernels

Theorem (Mercer): If k is

1. symmetric, i.e. $k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i)$ and
2. positive definite, i.e.

$$K = \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \dots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \dots & k(\mathbf{x}_N, \mathbf{x}_N) \end{pmatrix}$$

"Gram Matrix"

is positive definite, then there exists a mapping $\phi(\mathbf{x})$ into a feature space $\mathcal{H}$ so that k can be expressed as an inner product in $\mathcal{H}$.

This means, we don't need to find $\phi(\mathbf{x})$ explicitly!

We can directly work with k **"Kernel Trick"**

Constructing Kernels

Finding valid kernels from scratch is hard, but:

A number of rules exist to create a new valid kernel k from given kernels k_1 and k_2. For example:

$$k(\mathbf{x}_1, \mathbf{x}_2) = c k_1(\mathbf{x}_1, \mathbf{x}_2), \quad c > 0$$

$$k(\mathbf{x}_1, \mathbf{x}_2) = f(\mathbf{x}_1) k_1(\mathbf{x}_1, \mathbf{x}_2) f(\mathbf{x}_2)$$

$$k(\mathbf{x}_1, \mathbf{x}_2) = \exp(k_1(\mathbf{x}_1, \mathbf{x}_2))$$

$$k(\mathbf{x}_1, \mathbf{x}_2) = k_1(\mathbf{x}_1, \mathbf{x}_2) + k_2(\mathbf{x}_1, \mathbf{x}_2)$$

$$k(\mathbf{x}_1, \mathbf{x}_2) = k_1(\mathbf{x}_1, \mathbf{x}_2) k_2(\mathbf{x}_1, \mathbf{x}_2)$$

$$k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T A \mathbf{x}_2$$

where A is positive semidefinite and symmetric

Examples of Valid Kernels

- Polynomial Kernel:

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + c)^d \quad c > 0 \quad d \in \mathbb{N}$$

- Gaussian Kernel:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2)$$

- Kernel for sets:

$$k(A_1, A_2) = 2^{|A_1 \cap A_2|}$$

- Matern kernel:

$$k(r) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu r}}{l} \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu r}}{l} \right) \qquad r = \|\mathbf{x}_i - \mathbf{x}_j\|, \nu > 0, l > 0$$

A Simple Example

Define a kernel function as

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}')^2 \qquad \mathbf{x}, \mathbf{x}' \in \mathbb{R}^2$$

This can be written as:

$$\begin{aligned}(x_1 x_1' + x_2 x_2')^2 &= x_1^2 x_1'^2 + 2 x_1 x_1' x_2 x_2' + x_2^2 x_2'^2 \\ &= (x_1^2, x_2^2, \sqrt{2} x_1 x_2)(x_1'^2, x_2'^2, \sqrt{2} x_1' x_2')^T \\ &= \phi(\mathbf{x})^T \phi(\mathbf{x}')\end{aligned}$$

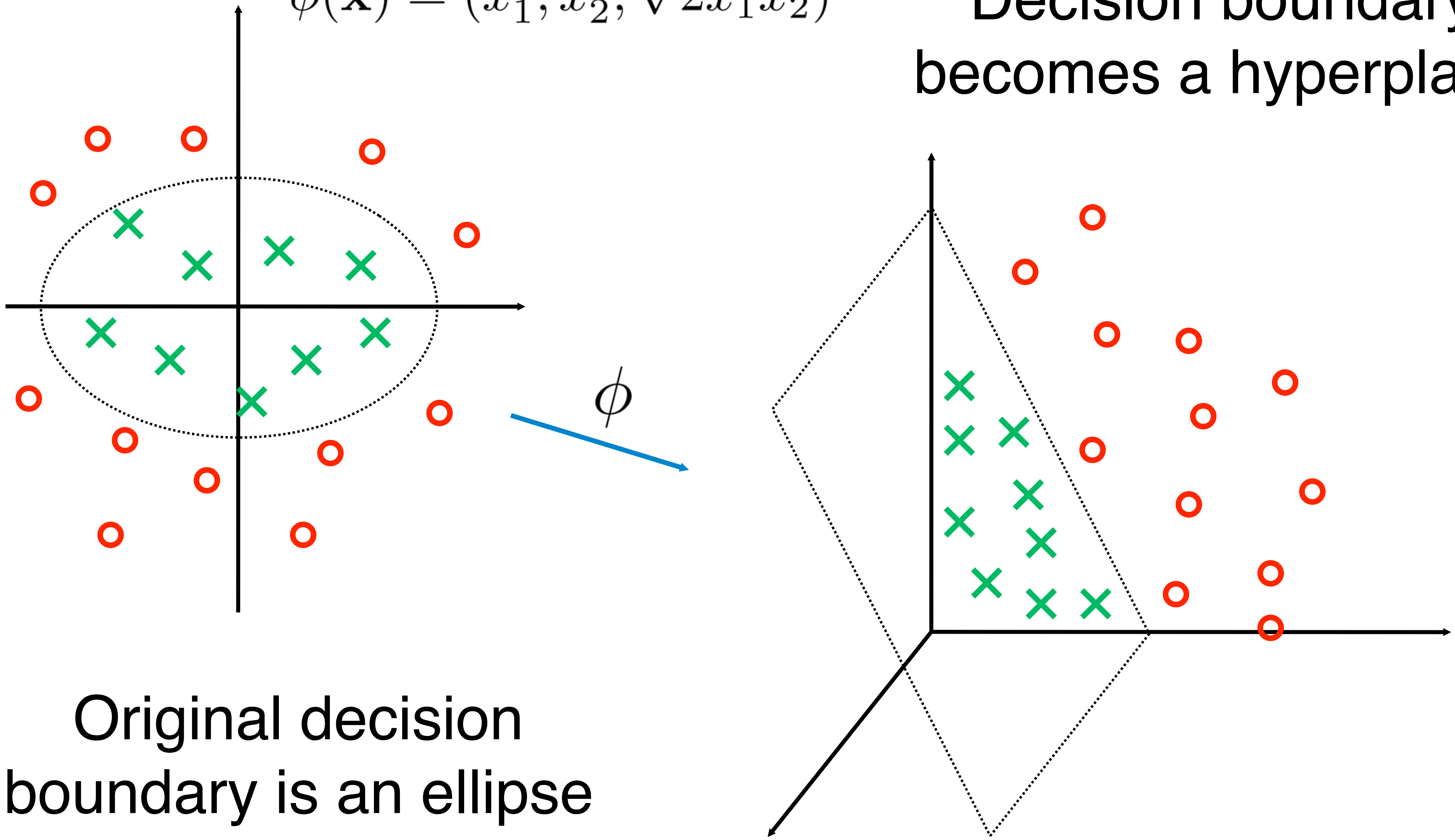
It can be shown that this holds in general for

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^d$$

Visualization of the Example

$$\phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

Decision boundary becomes a hyperplane



Original decision boundary is an ellipse

Application Examples

Kernel Methods can be applied for many different problems, e.g.:

- Density estimation (unsupervised learning)
- Regression
- Principal Component Analysis (PCA)
- Classification

Most important Kernel Methods are

- Support Vector Machines
- Gaussian Processes

Kernelization

- Many existing algorithms can be converted into kernel methods
- This process is called “kernelization”

Idea:

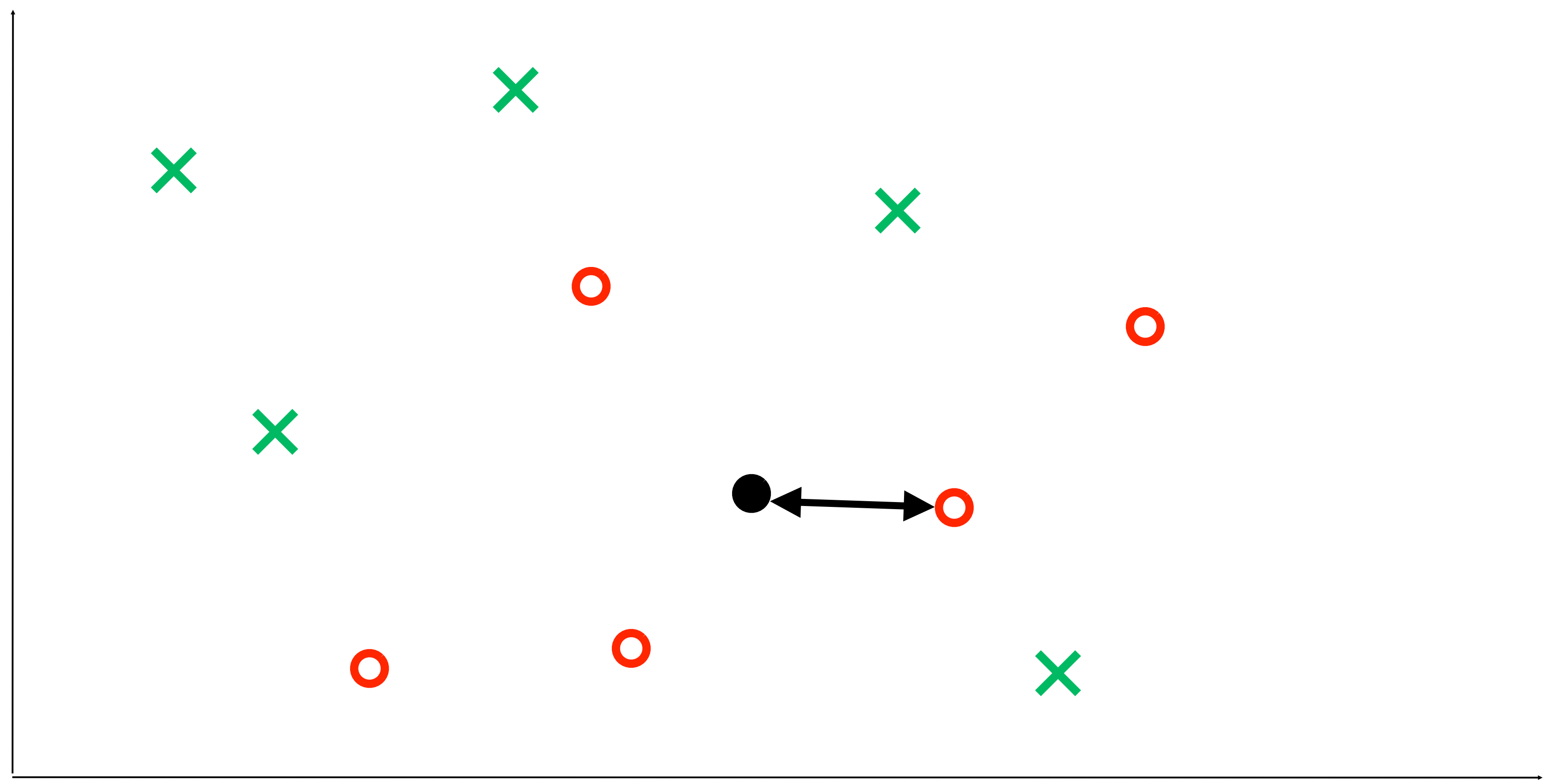
- express similarities of data points in terms of an inner product (dot product)
- replace all occurrences of that inner product by the kernel function

This is called the **kernel trick**

Example: Nearest Neighbor

- The NN classifier selects the label of the nearest neighbor in Euclidean distance

$$\|\mathbf{x}_i - \mathbf{x}_j\|^2 = \mathbf{x}_i^T\mathbf{x}_i + \mathbf{x}_j^T\mathbf{x}_j - 2\mathbf{x}_i^T\mathbf{x}_j$$



Example: Nearest Neighbor

- The NN classifier selects the label of the nearest neighbor in Euclidean distance

$$\|\mathbf{x}_i - \mathbf{x}_j\|^2 = \mathbf{x}_i^T \mathbf{x}_i + \mathbf{x}_j^T \mathbf{x}_j - 2\mathbf{x}_i^T \mathbf{x}_j$$

- We can now replace the dot products by a valid Mercer kernel and we obtain:

$$d(\mathbf{x}_i, \mathbf{x}_j)^2 = k(\mathbf{x}_i, \mathbf{x}_i) + k(\mathbf{x}_j, \mathbf{x}_j) - 2k(\mathbf{x}_i, \mathbf{x}_j)$$

- This is a **kernelized** nearest-neighbor classifier
- We do not explicitly compute feature vectors!

Back to Linear Regression (Rep.)

We had the primal and the dual formulation:

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\Phi^T\Phi\mathbf{w} - \mathbf{w}\Phi^T\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^T KK\mathbf{a} - \mathbf{a}^T K\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^T K\mathbf{a}$$

with the dual solution:

$$\mathbf{a} = (K + \lambda I_N)^{-1}\mathbf{t}$$

This we can use to make **predictions (MAP)**:

$$f(\mathbf{x}^*) = \mathbf{w}^T\phi(\mathbf{x}^*) = \mathbf{a}^T\Phi\phi(\mathbf{x}^*) = \mathbf{k}(\mathbf{x}^*)^T(K + \lambda I_N)^{-1}\mathbf{t}$$

Observations

- We have found a way to predict function values of y for new input points $\mathbf{x}^*$
- As we used regularized regression, we can equivalently find the **predictive distribution** by marginalizing out the parameters $\mathbf{w}$

Questions:

- Can we find a closed form for that distribution?
- How can we model the uncertainty of our prediction?
- Can we use that for classification?

Gaussian Marginals and Conditionals

First, we need some formulae:

Assume we have two variables $\mathbf{x}_a$ and $\mathbf{x}_b$ that are **jointly** Gaussian distributed, i.e. $\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \Sigma)$

with

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{pmatrix} \qquad \boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{pmatrix} \qquad \Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}$$

Then the cond. distribution $p(\mathbf{x}_a \mid \mathbf{x}_b) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_{a|b}, \Sigma_{a|b})$

where $\boldsymbol{\mu}_{a|b} = \boldsymbol{\mu}_a + \Sigma_{ab}\Sigma_{bb}^{-1}(\mathbf{x}_b - \boldsymbol{\mu}_b)$

and $\Sigma_{a|b} = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}$ "Schur Complement"

The marginal is $p(\mathbf{x}_a) = \mathcal{N}(\mathbf{x}_a \mid \boldsymbol{\mu}_a, \Sigma_{aa})$

Gaussian Marginals and Conditionals

Main idea of the proof for the conditional (using inverse of block matrices):

$$\begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}^{-1} = \begin{pmatrix} I & 0 \\ -\Sigma_{bb}^{-1}\Sigma_{ba} & I \end{pmatrix} \begin{pmatrix} (\Sigma/\Sigma_{bb})^{-1} & 0 \\ 0 & \Sigma_{bb}^{-1} \end{pmatrix} \begin{pmatrix} I & -\Sigma_{ab}\Sigma_{bb}^{-1} \\ 0 & I \end{pmatrix}$$

The lower line corresponds to a quadratic form that is only dependent on $p(\mathbf{x}_b)$, i.e. the rest can be identified with the conditional Normal distribution $p(\mathbf{x}_a \mid \mathbf{x}_b)$.

(for details see, e.g. Bishop or Murphy)

Definition

Definition: A **Gaussian process** is a collection of random variables, any finite number of which have a joint Gaussian distribution.

The number of random variables can be **infinite**!

This means: a GP is a Gaussian distribution over **functions**!

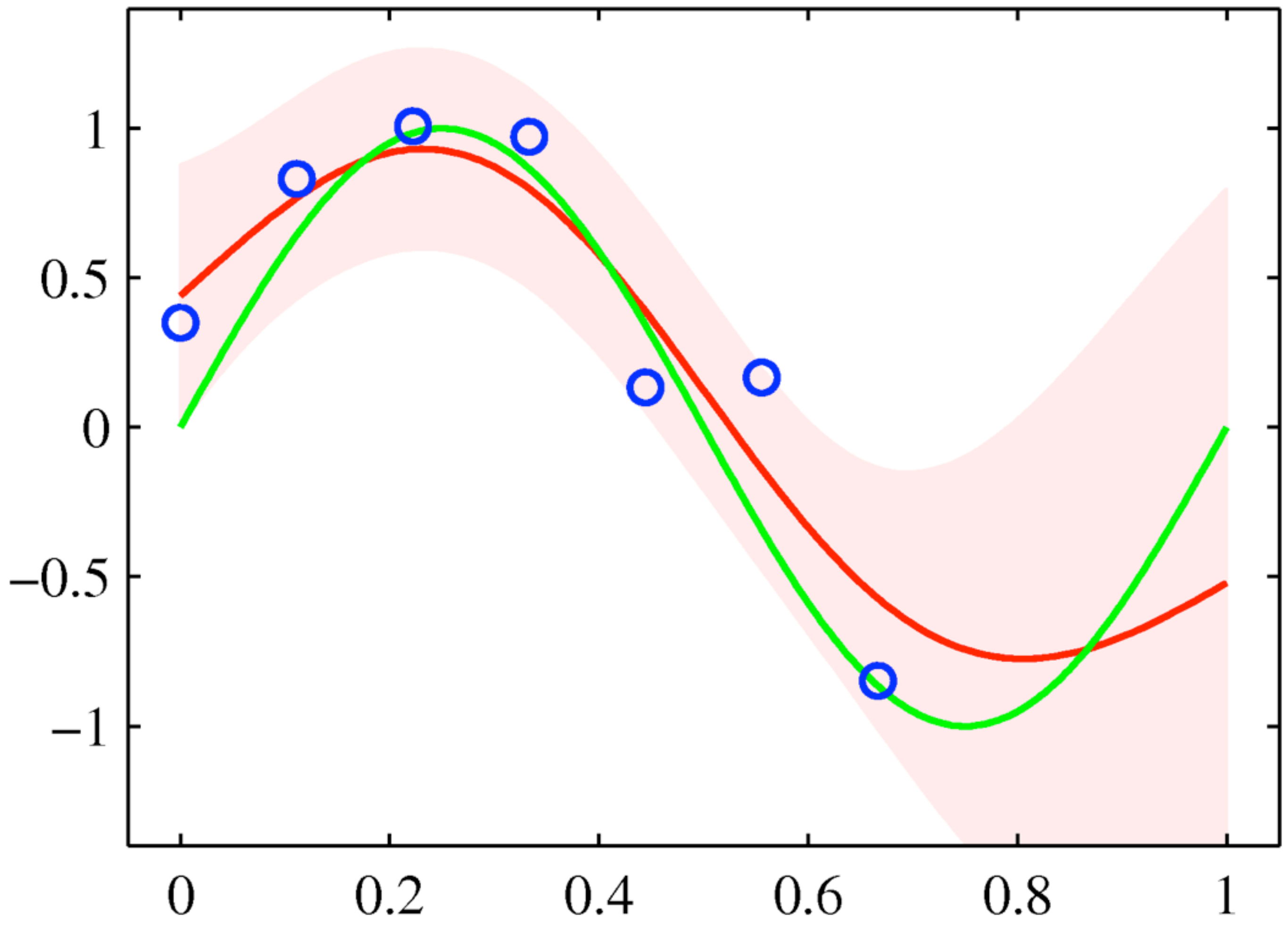
To specify a GP we need:

mean function: $m(\mathbf{x}) = \mathbb{E}[y(\mathbf{x})]$

covariance function:

$$k(\mathbf{x}_1, \mathbf{x}_2) = \mathbb{E}[y(\mathbf{x}_1) - m(\mathbf{x}_1)y(\mathbf{x}_2) - m(\mathbf{x}_2)]$$

Example



- green line: sinusoidal data source
- blue circles: data points with Gaussian noise
- red line: mean function of the Gaussian process

How Can We Handle Infinity?

Idea: split the (infinite) number of random variables into a finite and an infinite subset.

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_f \\ \mathbf{x}_i \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu}_f \\ \boldsymbol{\mu}_i \end{pmatrix}, \begin{pmatrix} \Sigma_f & \Sigma_{fi} \\ \Sigma_{fi}^T & \Sigma_i \end{pmatrix} \right)$$

finite part

infinite part

From the marginalization property we get:

$$p(\mathbf{x}_f) = \int p(\mathbf{x}_f, \mathbf{x}_i) d\mathbf{x}_i = \mathcal{N}(\mathbf{x}_f \mid \boldsymbol{\mu}_f, \Sigma_f)$$

This means we can use finite vectors.

A Simple Example

In Bayesian linear regression, we had $y(\mathbf{x}) = \phi(\mathbf{x})^T\mathbf{w}$ with prior probability $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p)$. This means:

$$\mathbb{E}[y(\mathbf{x})] = \phi(\mathbf{x})^T\mathbb{E}[\mathbf{w}] = \mathbf{0}$$

$$\mathbb{E}[y(\mathbf{x}_1)y(\mathbf{x}_2))] = \phi(\mathbf{x}_1)^T\mathbb{E}[\mathbf{w}\mathbf{w}^T]\phi(\mathbf{x}_2) = \phi(\mathbf{x}_1)^T\Sigma_p\phi(\mathbf{x}_2)$$

Any number of function values $y(\mathbf{x}_1), \dots, y(\mathbf{x}_N)$ is jointly Gaussian with zero mean.

The covariance function of this process is

$$k(\mathbf{x}_1, \mathbf{x}_2) = \phi(\mathbf{x}_1)^T\Sigma_p\phi(\mathbf{x}_2)$$

In general, any valid kernel function can be used.

The Covariance Function

The most used covariance function (kernel) is:

$$k(\mathbf{x}_p, \mathbf{x}_q) = \underbrace{\sigma_f^2}_{\text{signal variance}} \exp(-\frac{1}{\underbrace{2l^2}_{\text{length scale}}}(\mathbf{x}_p - \mathbf{x}_q)^2) + \underbrace{\sigma_n^2}_{\text{noise variance}}\delta_{pq}$$

It is known as “squared exponential”, “radial basis function” or “Gaussian kernel”.

Other possibilities exist, e.g. the exponential kernel:

$$k(\mathbf{x}_p, \mathbf{x}_q) = \exp(-\theta|\mathbf{x}_p - \mathbf{x}_q|)$$

This is used in the “Ornstein-Uhlenbeck” process.

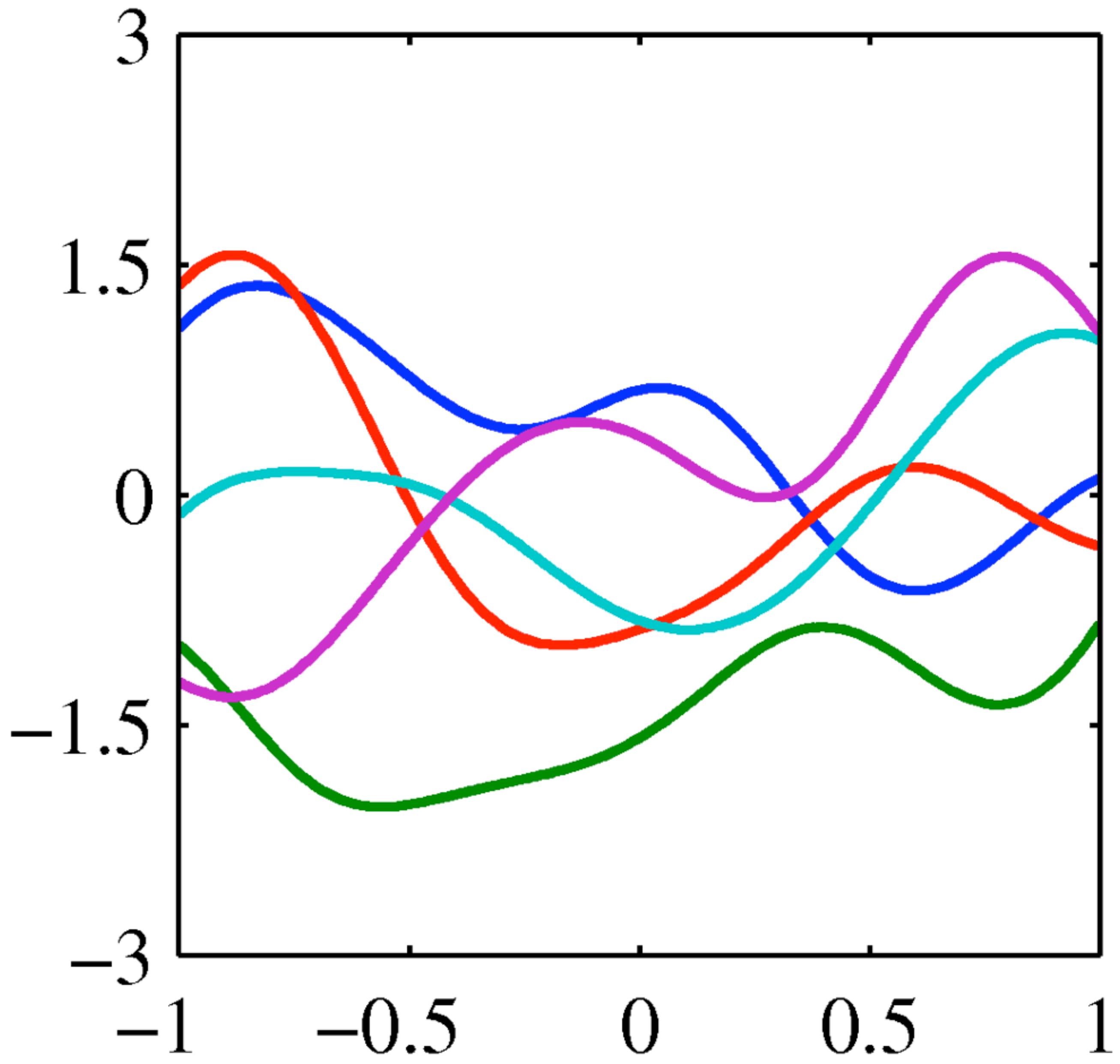
Sampling from a GP

Just as we can sample from a Gaussian distribution, we can also generate samples from a GP. **Every sample will then be a function!**

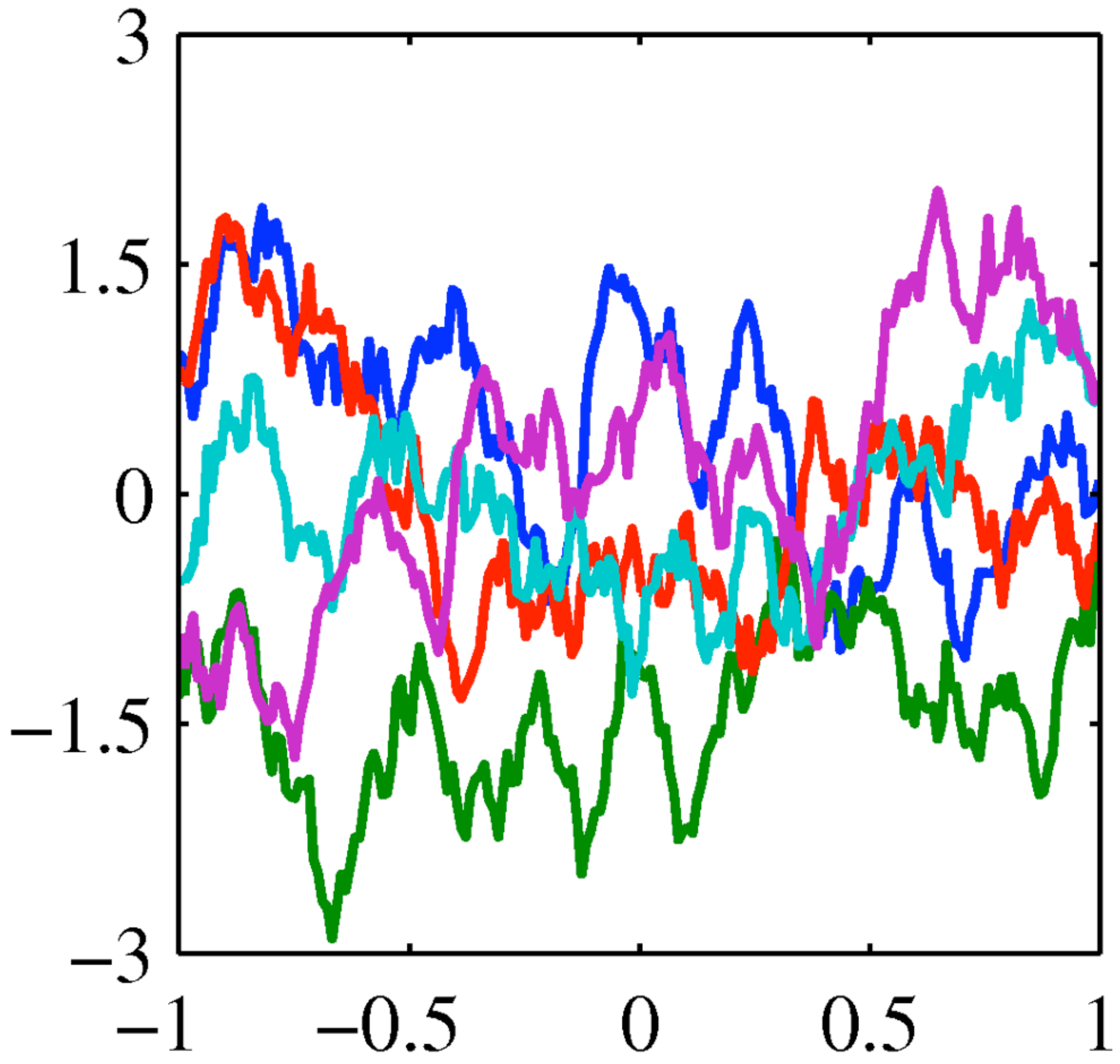
Process:

1. Choose a number of input points $\mathbf{x}_1^*, \dots, \mathbf{x}_M^*$
2. Compute the covariance matrix K where
$$K_{ij} = k(\mathbf{x}_i^*, \mathbf{x}_j^*)$$
3. Generate a random Gaussian vector from
$$\mathbf{y}_* \sim \mathcal{N}(\mathbf{0}, K)$$
4. Plot the values $\mathbf{x}_1^*, \dots, \mathbf{x}_M^*$ versus $y_1^*, \dots, y_M^*$

Sampling from a GP



Squared exponential kernel



Exponential kernel