

Example with Numbers

Assume we have a second sensor:

$$p(z_2 \mid \text{open}) = 0.5 \quad p(z_2 \mid \neg \text{open}) = 0.6$$

$$p(\text{open} \mid z_1) = \frac{2}{3} \quad (\text{from above})$$

Then: $p(\text{open} \mid z_1, z_2) =$

$$\frac{p(z_2 \mid \text{open})p(\text{open} \mid z_1)}{p(z_2 \mid \text{open})p(\text{open} \mid z_1) + p(z_2 \mid \neg \text{open})p(\neg \text{open} \mid z_1)}$$
$$= \frac{\frac{1}{2} \cdot \frac{2}{3}}{\frac{1}{2} \cdot \frac{2}{3} + \frac{3}{5} \cdot \frac{1}{3}} = \frac{5}{8} = 0.625$$

“ z_2 lowers the probability that the door is open”



General Form

Measurements: $z_1, \dots, z_n$

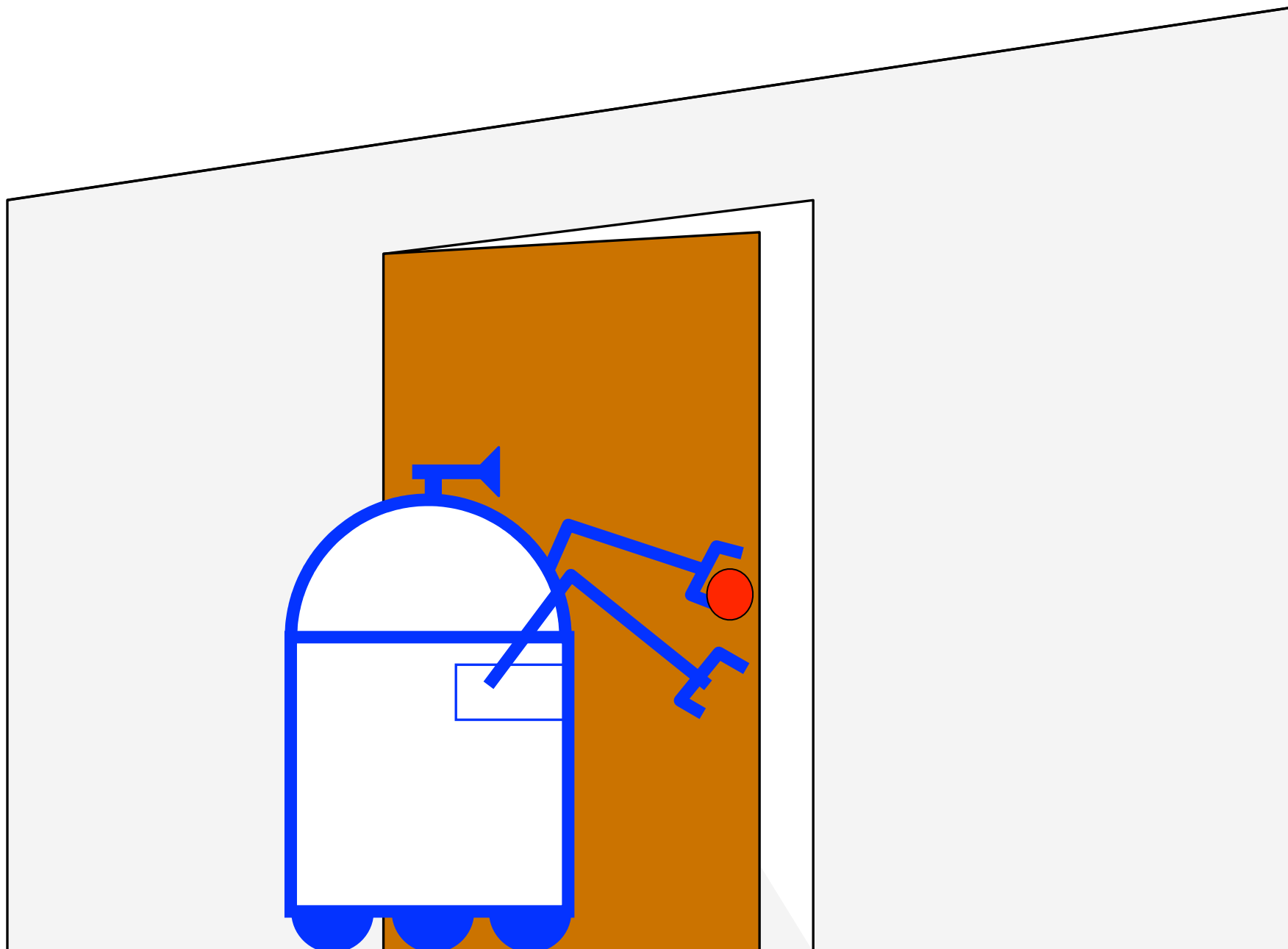
Markov assumption: z_n and $z_1, \dots, z_{n-1}$ are conditionally independent given the state x .

$$\begin{aligned} p(x \mid z_1, \dots, z_n) &= \frac{p(z_n \mid x)p(x \mid z_1, \dots, z_{n-1})}{p(z_n \mid z_1, \dots, z_{n-1})} \\ &\stackrel{\text{Recursion}}{=} \prod_{i=1}^n \eta_i p(z_i \mid x)p(x) \end{aligned}$$



Example: Sensing and Acting

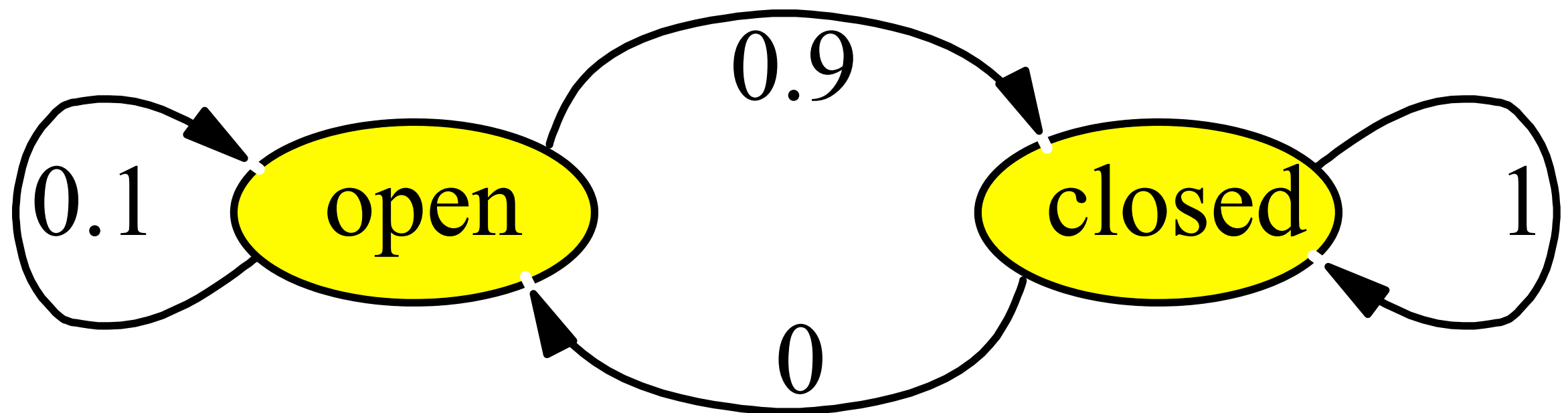
Now the robot *senses* the door state and *acts* (it opens or closes the door).



State Transitions

The *outcome* of an action is modeled as a random variable U where $U = u$ in our case means “state after closing the door”.

State transition example:



If the door is open, the action “close door” succeeds in 90% of all cases.



The Outcome of Actions

For a given action u we want to know the probability $p(x \mid u)$. We do this by integrating over all possible **previous** states x' .

If the state space is discrete:

$$p(x \mid u) = \sum_{x'} p(x \mid u, x') p(x')$$

If the state space is continuous:

$$p(x \mid u) = \int p(x \mid u, x') p(x') dx'$$



Back to the Example

$$\begin{aligned} p(\text{open} \mid u) &= \sum_{x'} p(\text{open} \mid u, x') p(x') \\ &= p(\text{open} \mid u, \text{open}') p(\text{open}') + \\ &\quad p(\text{open} \mid u, \neg \text{open}') p(\neg \text{open}') \\ &= \frac{1}{10} \cdot \frac{5}{8} + 0 \cdot \frac{3}{8} \\ &= \frac{1}{16} = 0.0625 \end{aligned}$$

$$p(\neg \text{open} \mid u) = 1 - p(\text{open} \mid u) = \frac{15}{16} = 0.9375$$



Sensor Update and Action Update

So far, we learned two different ways to update the system state:

- Sensor update: $p(x \mid z)$
- Action update: $p(x \mid u)$
- Now we want to combine both:

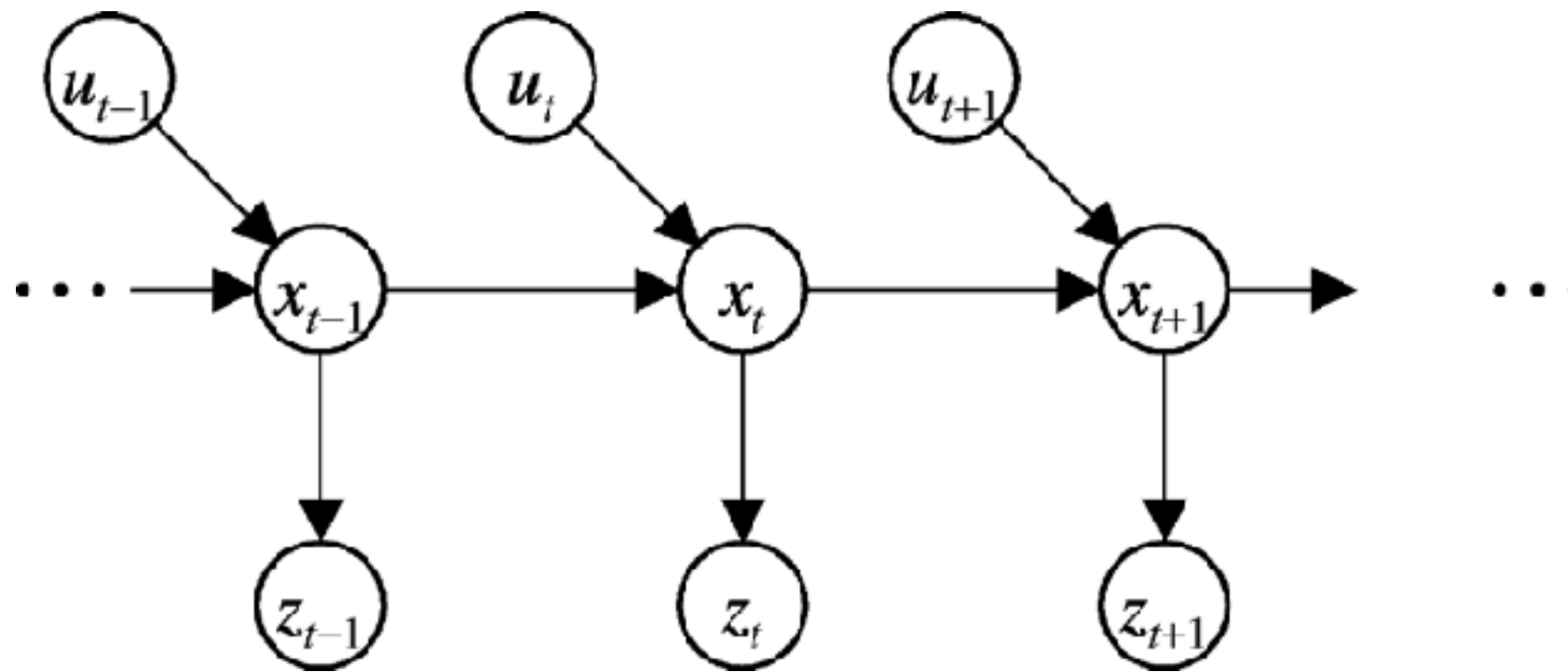
Definition 2.1: Let $D_t = u_1, z_1, \dots, u_t, z_t$ be a sequence of sensor measurements and actions until time t . Then the **belief** of the current state x_t is defined as

$$\text{Bel}(x_t) = p(x_t \mid u_1, z_1, \dots, u_t, z_t)$$



Graphical Representation

We can describe the overall process using a *Dynamic Bayes Network*:



This incorporates the following Markov assumptions:

$$p(z_t \mid x_{0:t}, u_{1:t}, z_{1:t}) = p(z_t \mid x_t) \quad (\text{measurement})$$

$$p(x_t \mid x_{0:t-1}, u_{1:t}, z_{1:t-1}) = p(x_t \mid x_{t-1}, u_t) \quad (\text{state})$$



The Overall Bayes Filter

$$\text{Bel}(x_t) = p(x_t \mid u_1, z_1, \dots, u_t, z_t)$$

(Bayes)

$$= \eta \, p(z_t \mid x_t, u_1, z_1, \dots, u_t) p(x_t \mid u_1, z_1, \dots, u_t)$$

(Markov)

$$= \eta \, p(z_t \mid x_t) p(x_t \mid u_1, z_1, \dots, u_t)$$

(Tot. prob.)

$$= \eta \, p(z_t \mid x_t) \int p(x_t \mid u_1, z_1, \dots, u_t, x_{t-1}) \\ p(x_{t-1} \mid u_1, z_1, \dots, u_t) dx_{t-1}$$

(Markov)

$$= \eta \, p(z_t \mid x_t) \int p(x_t \mid u_t, x_{t-1}) p(x_{t-1} \mid u_1, z_1, \dots, u_t) dx_{t-1}$$

(Markov)

$$= \eta \, p(z_t \mid x_t) \int p(x_t \mid u_t, x_{t-1}) p(x_{t-1} \mid u_1, z_1, \dots, z_{t-1}) dx_{t-1}$$
$$= \eta \, p(z_t \mid x_t) \int p(x_t \mid u_t, x_{t-1}) \text{Bel}(x_{t-1}) dx_{t-1}$$



The Bayes Filter Algorithm

$$\text{Bel}(x_t) = \eta p(z_t | x_t) \int p(x_t | u_t, x_{t-1}) \text{Bel}(x_{t-1}) dx_{t-1}$$

Algorithm Bayes_filter ($\text{Bel}(x)$, d)

1. if d is a sensor measurement z then
2. $\eta = 0$
3. for all x do
4. $\text{Bel}'(x) \leftarrow p(z | x) \text{Bel}(x)$
5. $\eta \leftarrow \eta + \text{Bel}'(x)$
6. for all x do $\text{Bel}'(x) \leftarrow \eta^{-1} \text{Bel}'(x)$
7. else if d is an action u then
8. for all x do $\text{Bel}'(x) \leftarrow \int p(x | u, x') \text{Bel}(x') dx'$
9. return $\text{Bel}'(x)$



Bayes Filter Variants

$$\text{Bel}(x_t) = \eta p(z_t | x_t) \int p(x_t | u_t, x_{t-1}) \text{Bel}(x_{t-1}) dx_{t-1}$$

The Bayes filter principle is used in

- Kalman filters
- Particle filters
- Hidden Markov models
- Dynamic Bayesian networks
- Partially Observable Markov Decision Processes (POMDPs)



Summary

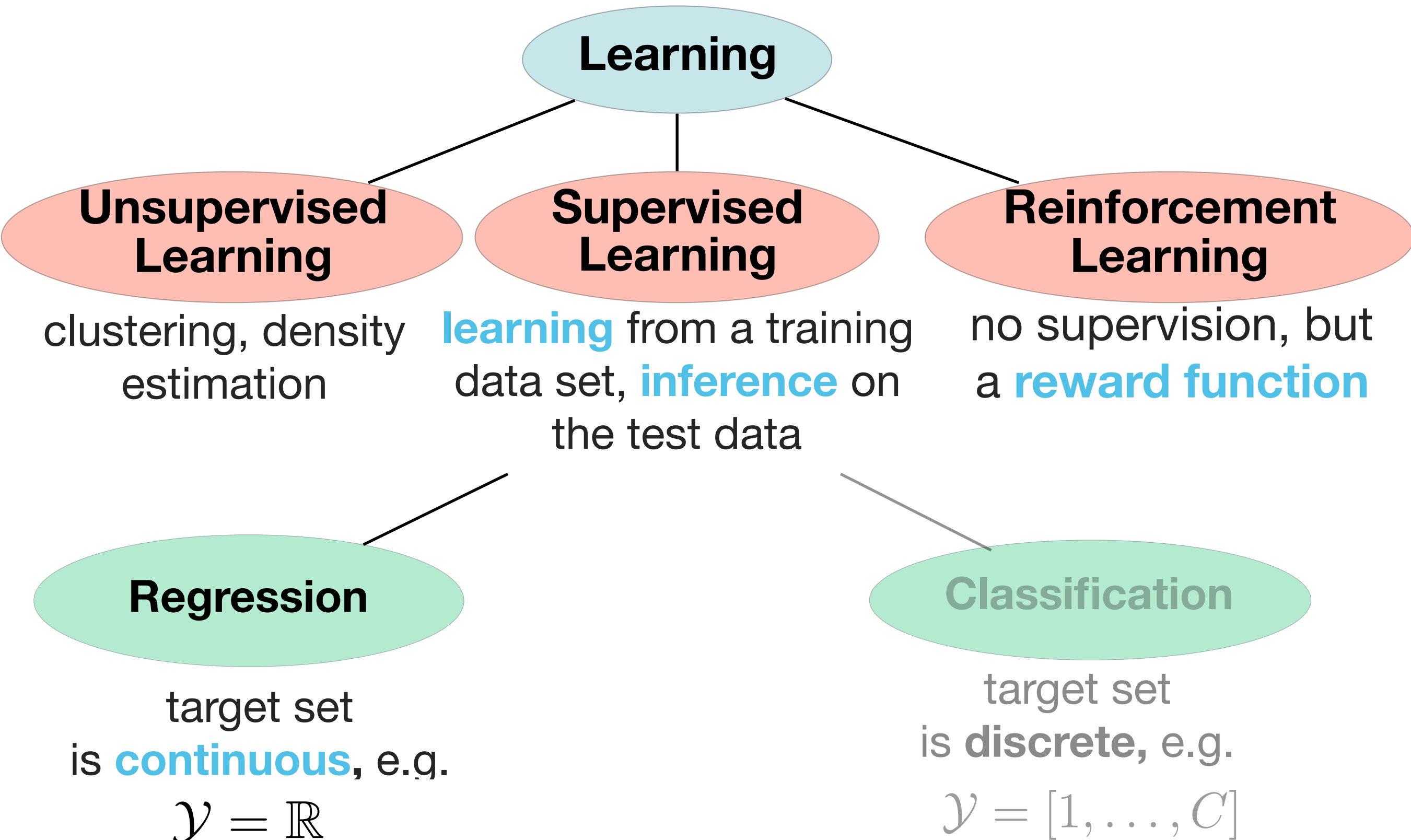
- *Probabilistic reasoning* is necessary to deal with uncertain information, e.g. sensor measurements
- Using *Bayes rule*, we can do diagnostic reasoning based on causal knowledge
- The outcome of a robot's action can be described by a *state transition diagram*
- Probabilistic state estimation can be done recursively using the *Bayes filter* using a sensor and a motion update
- A graphical representation for the state estimation problem is the *Dynamic Bayes Network*





2. Regression

Categories of Learning (Rep.)



Mathematical Formulation (Rep.)

Suppose we are given a set $\mathcal{X}$ of objects and a set $\mathcal{Y}$ of object categories (classes). In the learning task we search for a mapping $\varphi : \mathcal{X} \rightarrow \mathcal{Y}$ such that *similar* elements in $\mathcal{X}$ are mapped to *similar* elements in $\mathcal{Y}$.

Difference between regression and classification:

- In regression, $\mathcal{Y}$ is **continuous**, in classification it is discrete
- Regression learns a **function**, classification usually learns **class labels**

For now we will treat regression



Basis Functions

In principal, the elements of $\mathcal{X}$ can be anything (e.g. real numbers, graphs, 3D objects). To be able to treat these objects mathematically we need functions ϕ that map from $\mathcal{X}$ to $\mathbb{R}^M$. We call these the **basis functions**.

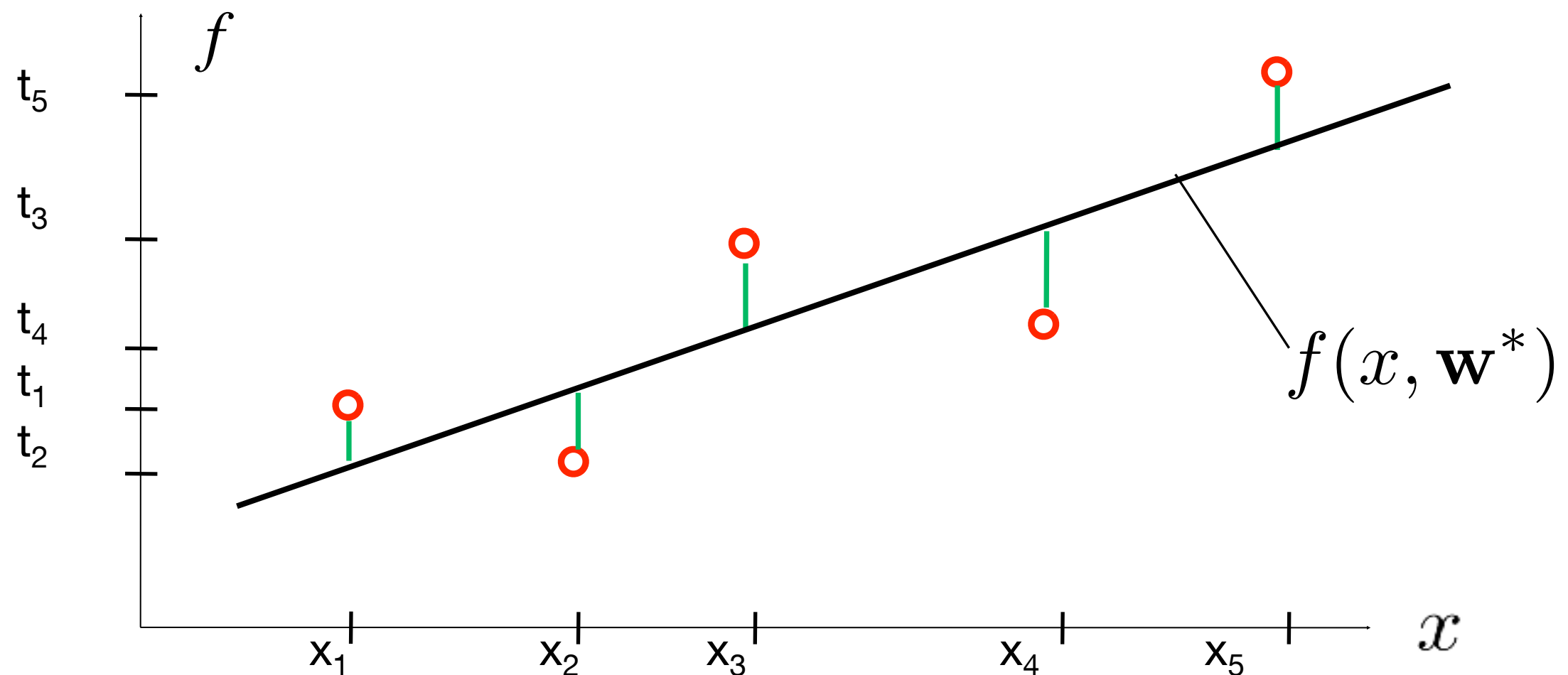
We can also interpret the basis functions as functions that extract **features** from the input data.

Features reflect the **properties** of the objects (width, height, etc.).



Simple Example: Linear Regression

- Assume: $\mathcal{X} = \mathbb{R}$, $\mathcal{Y} = \mathbb{R}$, $\phi = I$ (identity)
- **Given:** data points $(x_1, t_1), (x_2, t_2), \dots$
- **Goal:** predict the value t of a new example x
- Parametric formulation: $f(x, \mathbf{w}) = w_0 + w_1 x$



Linear Regression

To determine the function f , we need an error function:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (f(x_i, \mathbf{w}) - t_i)^2$$

“Sum of Squared Errors”

We search for parameters $\mathbf{w}^*$ s.th. $E(\mathbf{w}^*)$ is minimal:

$$\nabla E(\mathbf{w}) = \sum_{i=1}^N (f(x_i, \mathbf{w}) - t_i) \nabla f(x_i, \mathbf{w}) \stackrel{!}{=} \begin{pmatrix} 0 & 0 \end{pmatrix}$$

$$f(x, \mathbf{w}) = w_0 + w_1 x \quad \Rightarrow \quad \nabla f(x_i, \mathbf{w}) = \begin{pmatrix} 1 & x_i \end{pmatrix}$$



Linear Regression

To determine the function f , we need an error function:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (f(x_i, \mathbf{w}) - t_i)^2$$

“Sum of Squared Errors”

We search for parameters $\mathbf{w}^*$ s.th. $E(\mathbf{w}^*)$ is minimal:

$$\nabla E(\mathbf{w}) = \sum_{i=1}^N (f(x_i, \mathbf{w}) - t_i) \nabla f(x_i, \mathbf{w}) \stackrel{!}{=} \begin{pmatrix} 0 & 0 \end{pmatrix}$$

$$f(x, \mathbf{w}) = w_0 + w_1 x \quad \Rightarrow \quad \nabla f(x_i, \mathbf{w}) = \begin{pmatrix} 1 & x_i \end{pmatrix}$$

Using vector notation: $\mathbf{x}_i = \begin{pmatrix} 1 & x_i \end{pmatrix}^T \Rightarrow f(x_i, \mathbf{w}) = \mathbf{w}^T \mathbf{x}_i$



Linear Regression

To determine the function f , we need an error function:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (f(x_i, \mathbf{w}) - t_i)^2$$

“Sum of Squared Errors”

We search for parameters $\mathbf{w}^*$ s.th. $E(\mathbf{w}^*)$ is minimal:

$$\nabla E(\mathbf{w}) = \sum_{i=1}^N (f(x_i, \mathbf{w}) - t_i) \nabla f(x_i, \mathbf{w}) \stackrel{!}{=} \begin{pmatrix} 0 & 0 \end{pmatrix}$$

$$f(x, \mathbf{w}) = w_0 + w_1 x \Rightarrow \nabla f(x_i, \mathbf{w}) = \begin{pmatrix} 1 & x_i \end{pmatrix}$$

Using vector notation: $\mathbf{x}_i = \begin{pmatrix} 1 & x_i \end{pmatrix}^T \Rightarrow f(x_i, \mathbf{w}) = \mathbf{w}^T \mathbf{x}_i$

$$\nabla E(\mathbf{w}) = \sum_{i=1}^N \mathbf{w}^T \mathbf{x}_i \mathbf{x}_i^T - \sum_{i=1}^N t_i \mathbf{x}_i^T = \begin{pmatrix} 0 & 0 \end{pmatrix} \Rightarrow \mathbf{w}^T \underbrace{\sum_{i=1}^N \mathbf{x}_i \mathbf{x}_i^T}_{=: A^T} = \underbrace{\sum_{i=1}^N t_i \mathbf{x}_i^T}_{=: b^T}$$



Polynomial Regression

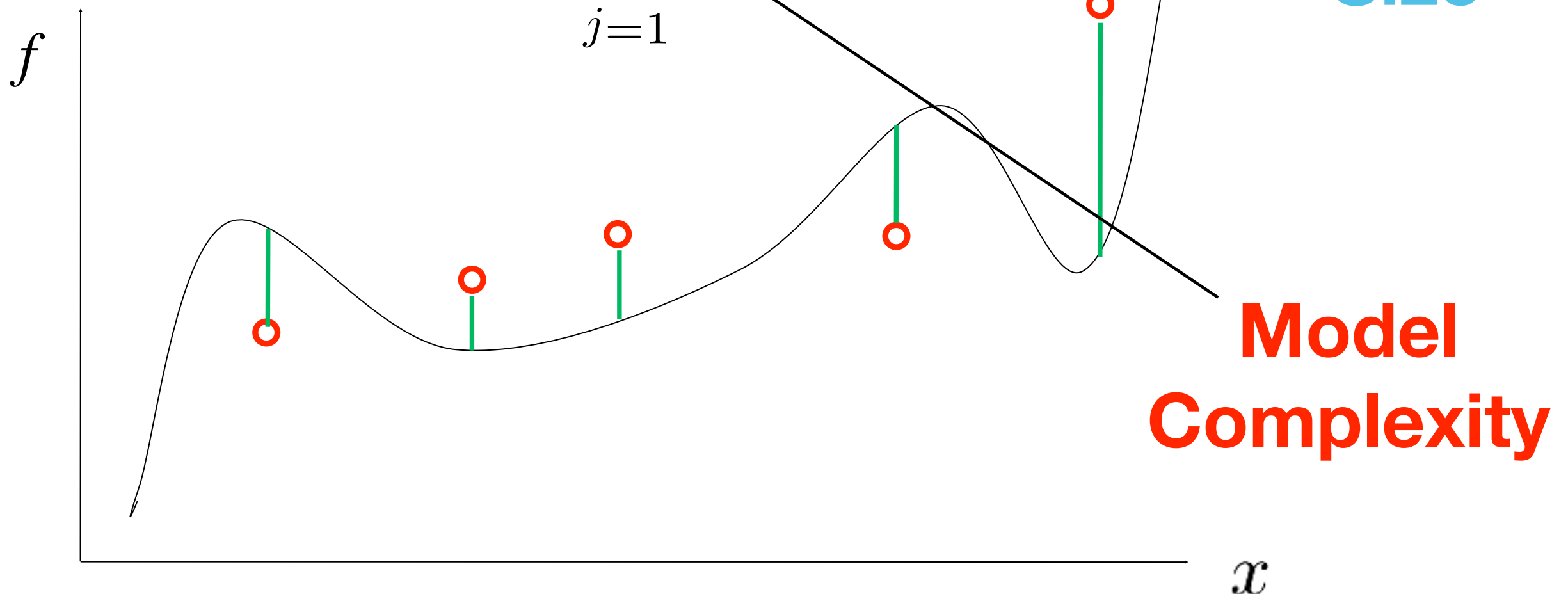
Now we have: $\mathcal{X} = \mathbb{R}$, $\mathcal{Y} = \mathbb{R}$, $\phi_j(x) = x^j$

Given: data points $(x_1, t_1), (x_2, t_2), \dots, (x_N, t_N)$

Assume we are given M **basis** functions

$$f(x, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(x) = \mathbf{w}^T \boldsymbol{\phi}(x)$$

**Data Set
Size**



**Model
Complexity**



Polynomial Regression

We have defined:

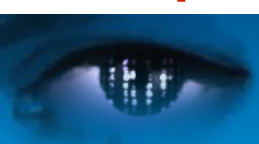
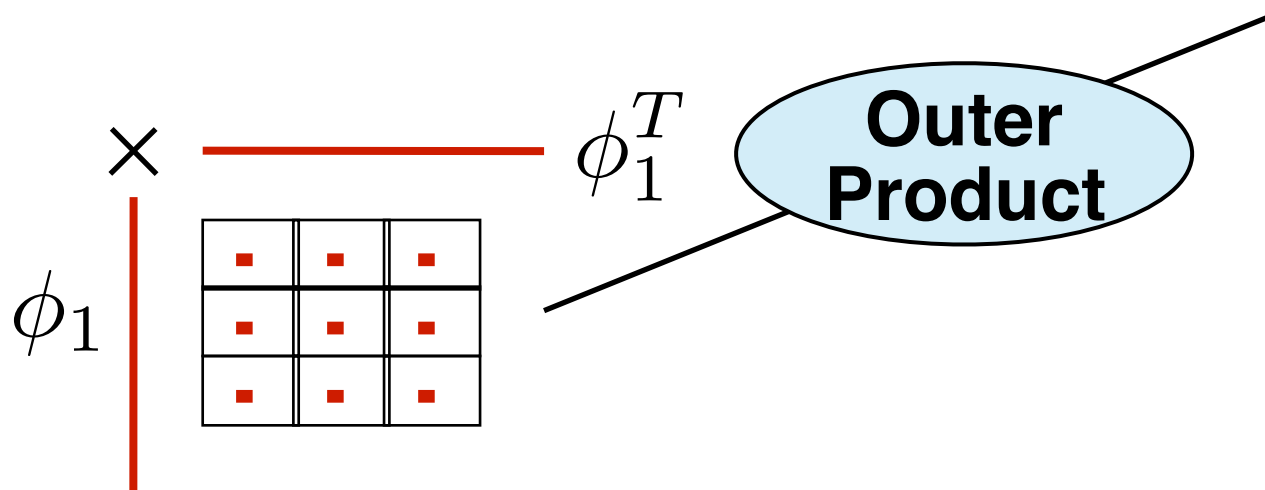
$$\phi(x) := (1, \phi_1(x), \dots, \phi_{M-1}(x))^T$$

“Basis functions”

Therefore: $f(x, \mathbf{w}) = \mathbf{w}^T \phi(x) \quad \mathbf{w} \in \mathbb{R}^M$

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (\mathbf{w}^T \phi(x_i) - t_i)^2$$

$$\nabla E(\mathbf{w}) = \mathbf{w}^T \left(\sum_{i=1}^N \phi(x_i) \phi(x_i)^T \right) - \sum_{i=1}^N t_i \phi(x_i)^T$$



Polynomial Regression

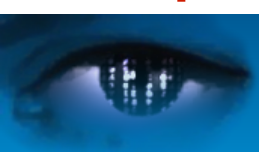
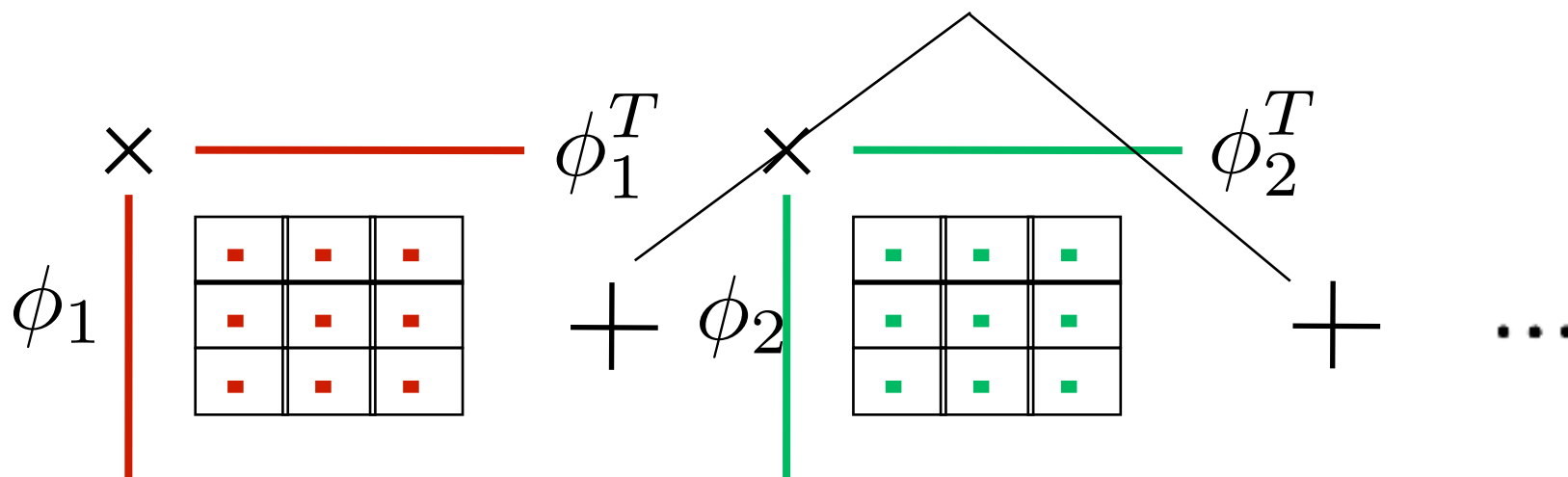
We have defined:

$$\phi(x) := (1, \phi_1(x), \dots, \phi_{M-1}(x))^T$$

Therefore: $f(x, \mathbf{w}) = \mathbf{w}^T \phi(x)$

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (\mathbf{w}^T \phi(x_i) - t_i)^2$$

$$\nabla E(\mathbf{w}) = \mathbf{w}^T \left(\sum_{i=1}^N \phi(x_i) \phi(x_i)^T \right) - \sum_{i=1}^N t_i \phi(x_i)^T$$



Polynomial Regression

We have defined:

$$\phi(x) := (1, \phi_1(x), \dots, \phi_{M-1}(x))^T$$

Therefore: $f(x, \mathbf{w}) = \mathbf{w}^T \phi(x)$

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (\mathbf{w}^T \phi(x_i) - t_i)^2$$

$$\nabla E(\mathbf{w}) = \mathbf{w}^T \left(\sum_{i=1}^N \phi(x_i) \phi(x_i)^T \right) - \sum_{i=1}^N t_i \phi(x_i)^T$$



Polynomial Regression

Thus, we have:
$$\sum_{i=1}^N \phi(x_i) \phi(x_i)^T = \Phi^T \Phi$$

where
$$\Phi = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \dots & \phi_{M-1}(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \dots & \phi_{M-1}(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_N) & \phi_1(x_N) & \dots & \phi_{M-1}(x_N) \end{pmatrix}$$

$$\nabla E(\mathbf{w}) = \mathbf{w}^T \Phi^T \Phi - \mathbf{t}^T \Phi \quad \Rightarrow \quad \Phi^T \Phi \mathbf{w} = \Phi^T \mathbf{t}$$

“Normal Equation”

It follows:

$$\mathbf{w} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{t} \quad \text{--- “Pseudoinverse” } \Phi^+$$



Computing the Pseudoinverse

Mathematically, a pseudoinverse Φ^+ exists for every matrix Φ .

However: If Φ is (close to) singular the direct solution of Φ is numerically unstable.

Therefore: Singular Value Decomposition (SVD) is used: $\Phi = UDV^T$ where

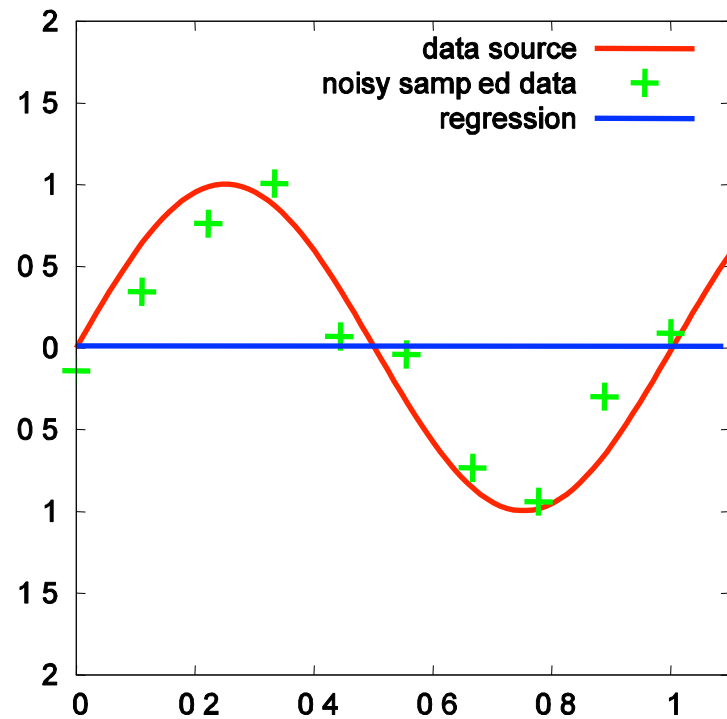
- matrices U and V are orthogonal matrices
- D is a diagonal matrix

Then: $\Phi^+ = VD^+U^T$ where D^+ contains the *reciprocal* of all non-zero elements of D



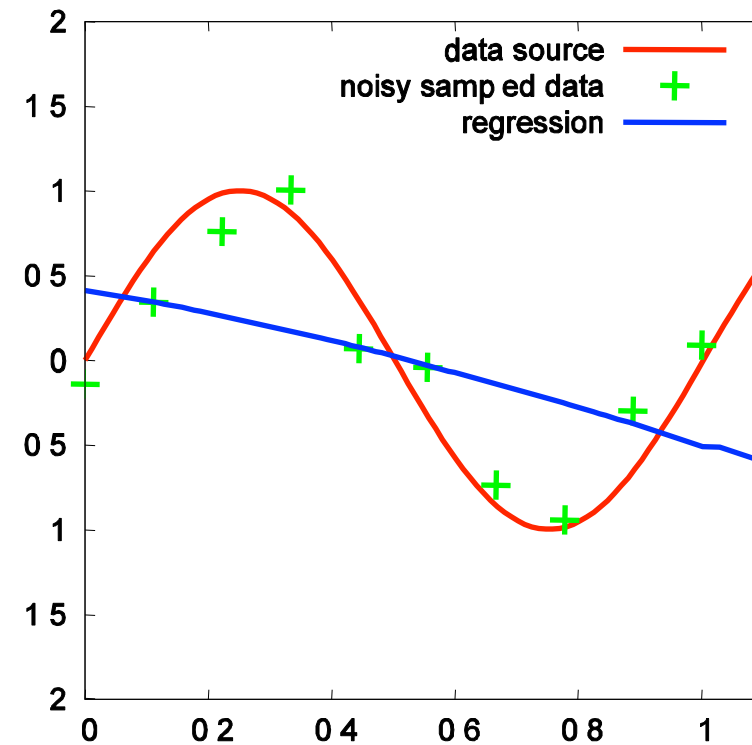
A Simple Example

$$\phi_j(x) = x^j$$



$$N = 10$$

$$M = 1$$



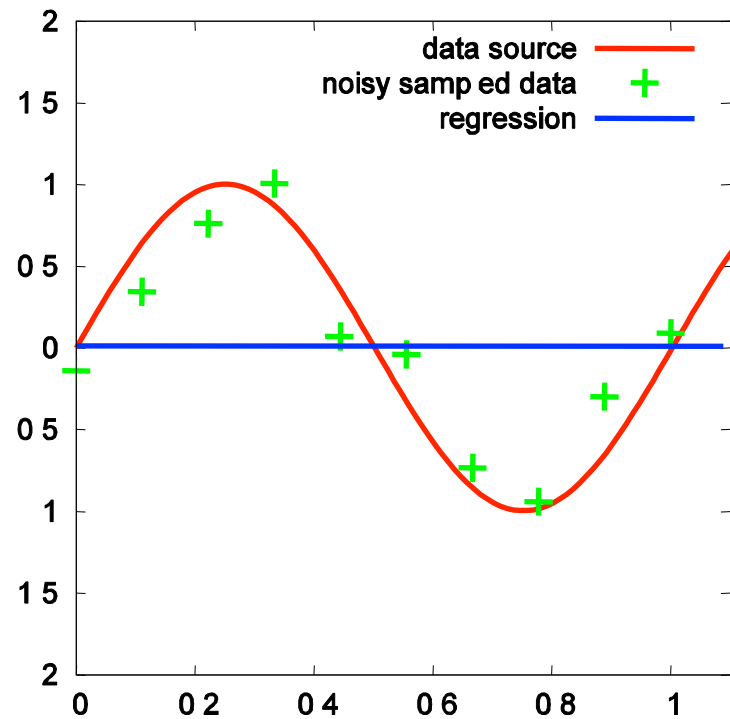
$$N = 10$$

$$M = 3$$



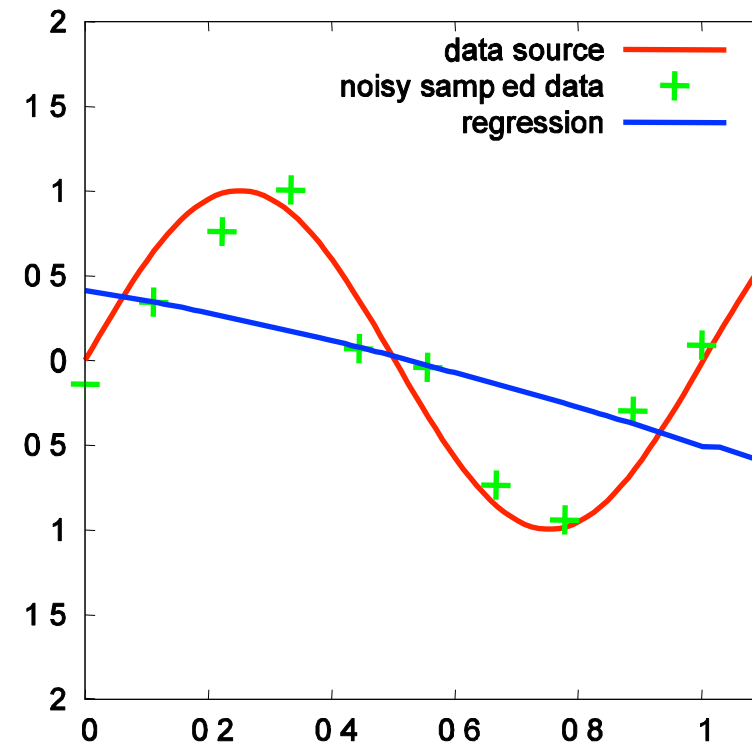
A Simple Example

$$\phi_j(x) = x^j$$



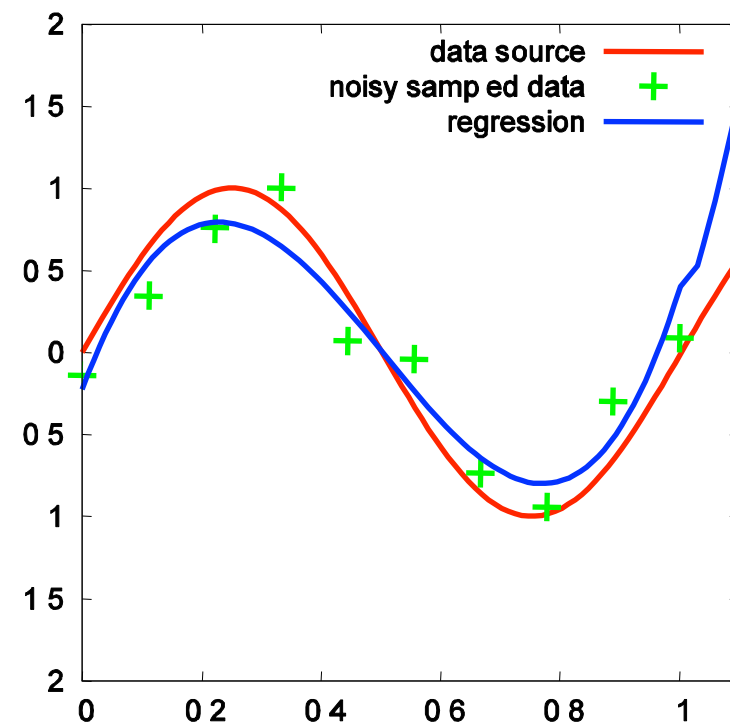
$$N = 10$$

$$M = 1$$



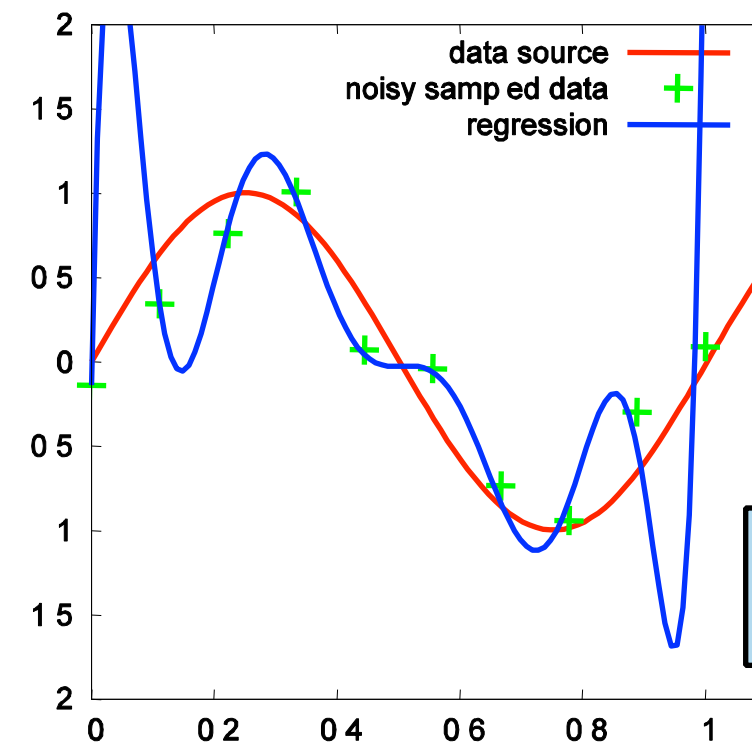
$$N = 10$$

$$M = 3$$



$$N = 10$$

$$M = 5$$



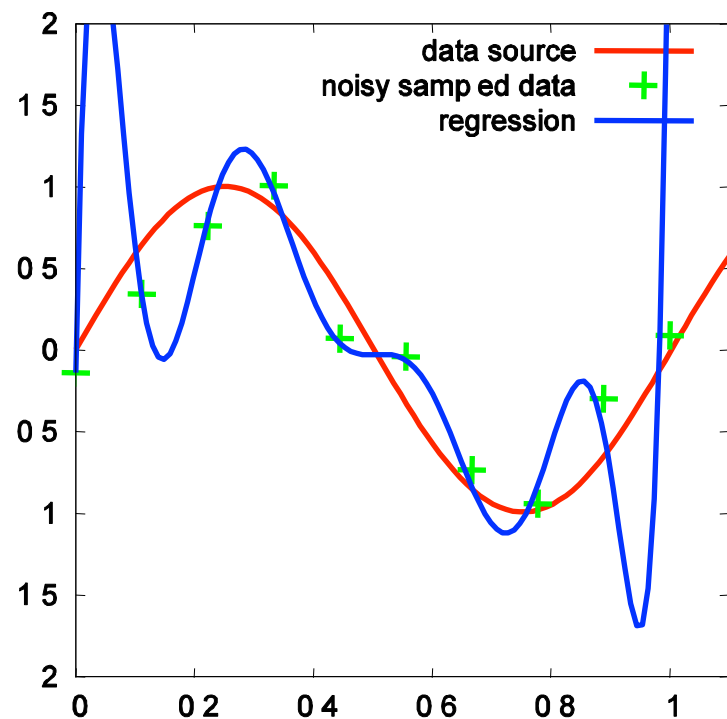
$$N = 10$$

$$M = 10$$

“Overfitting”

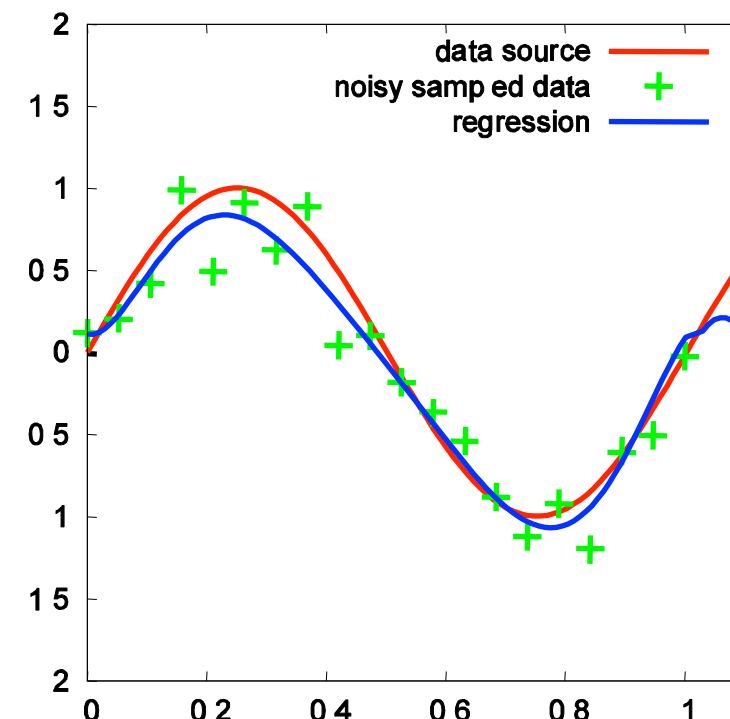


Varying the Sample Size



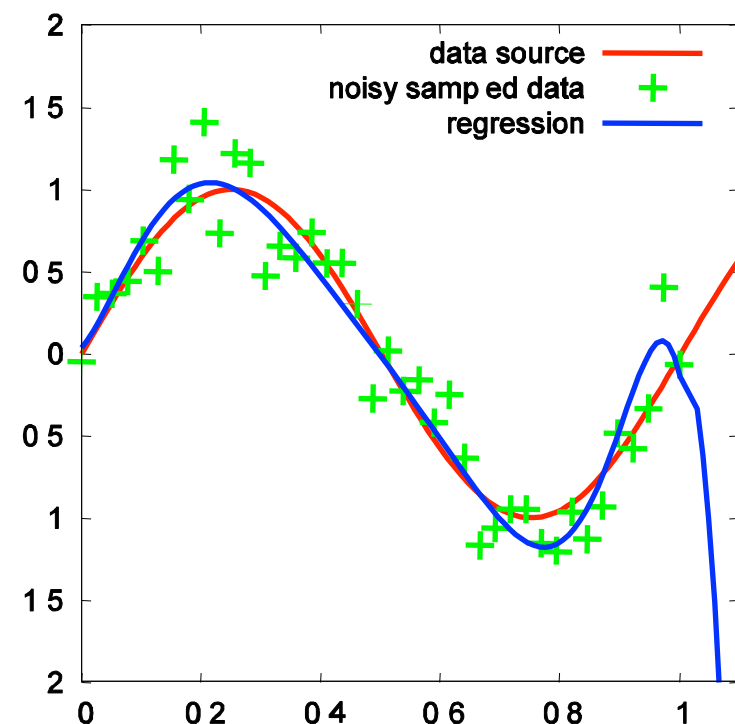
$$N = 10$$

$$M = 10$$



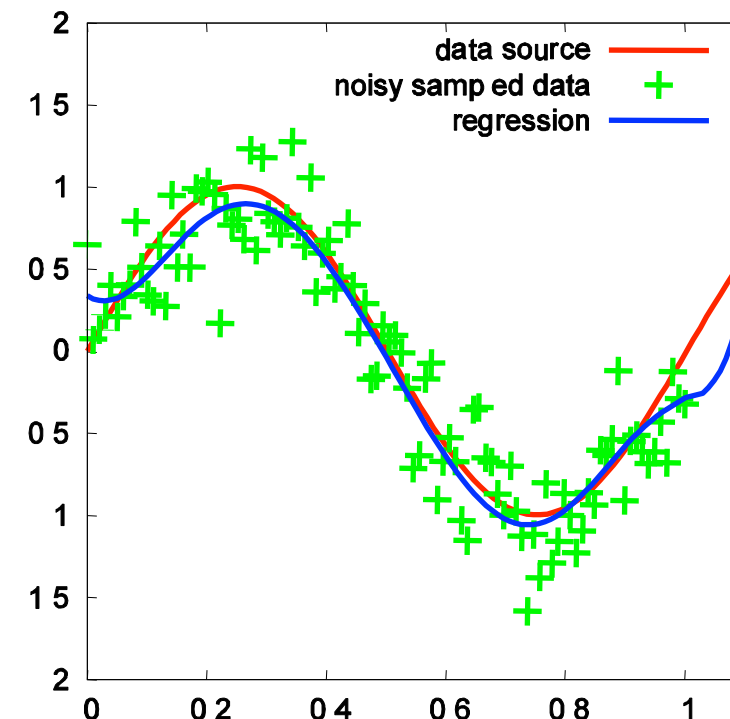
$$N = 20$$

$$M = 10$$



$$N = 40$$

$$M = 10$$

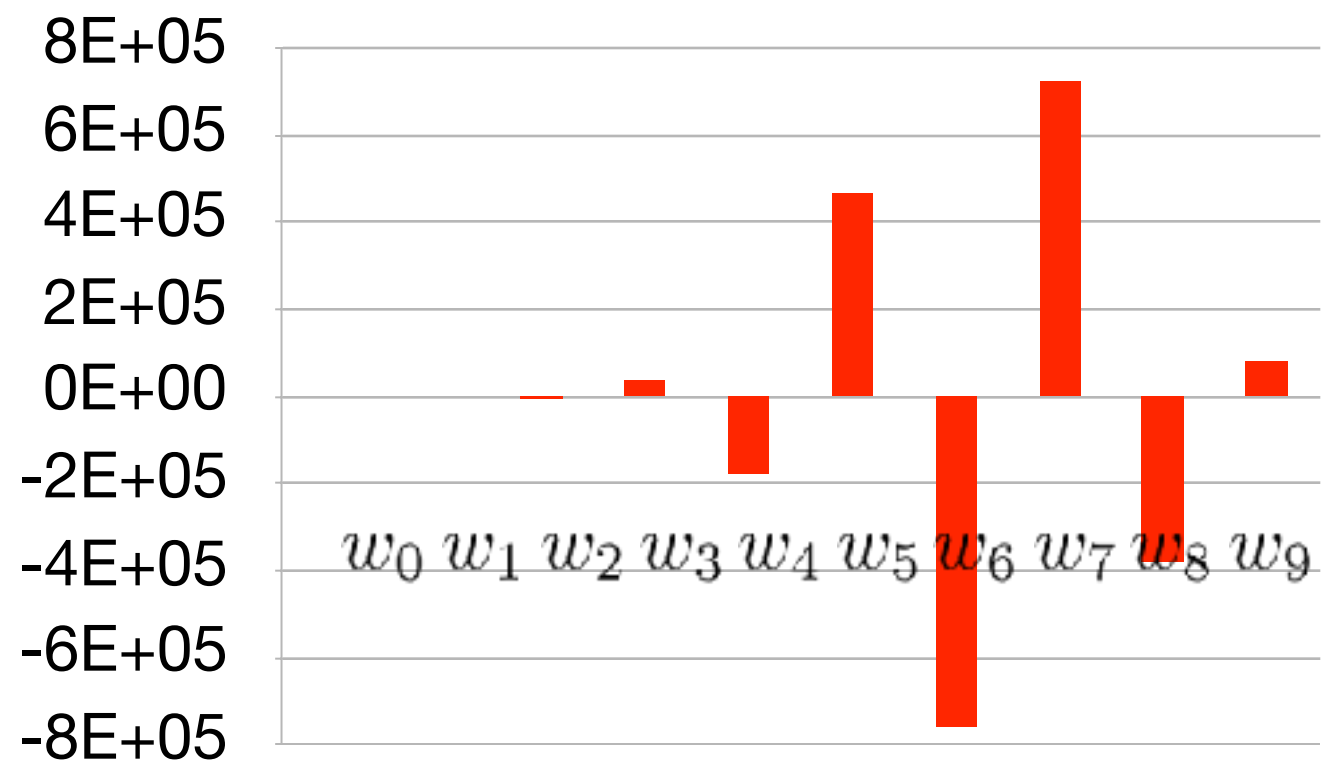
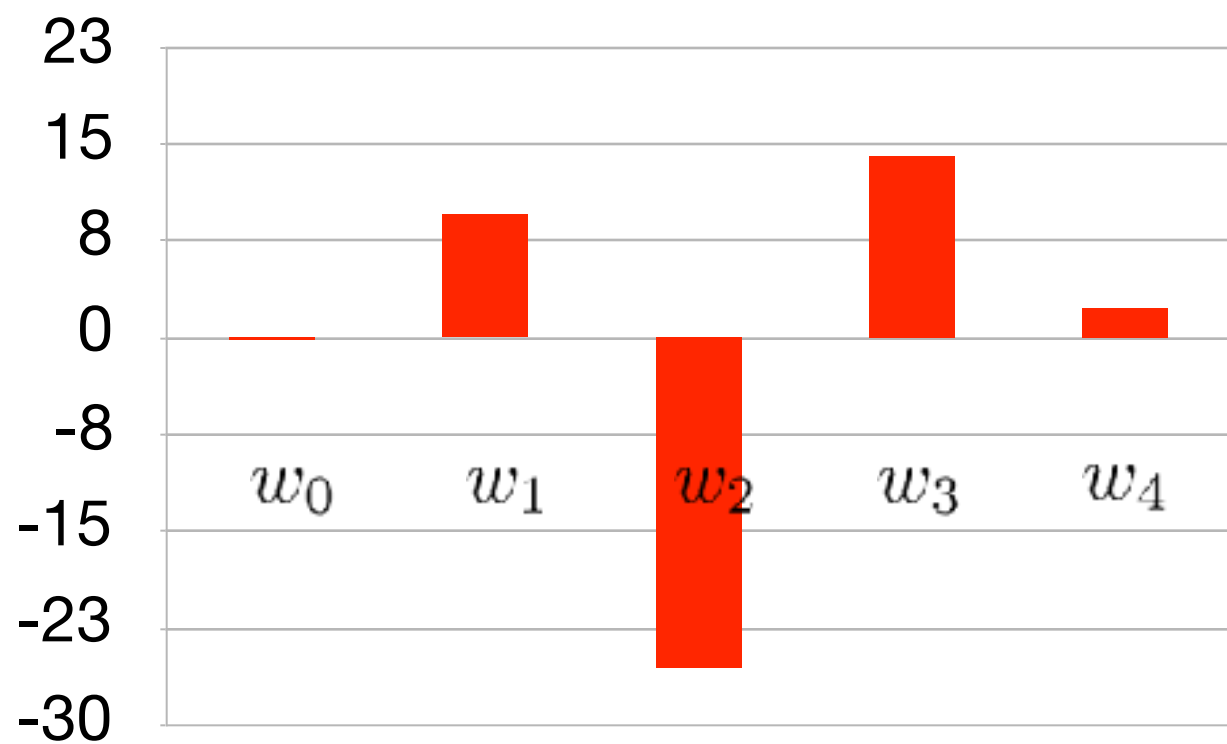
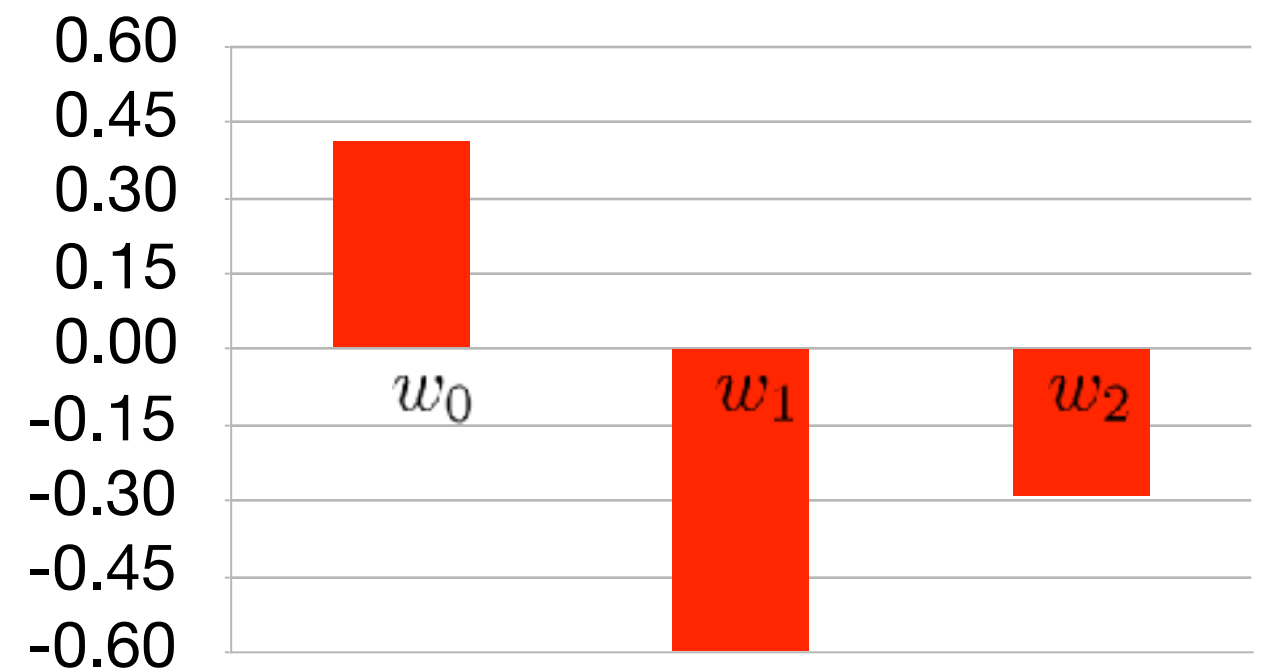
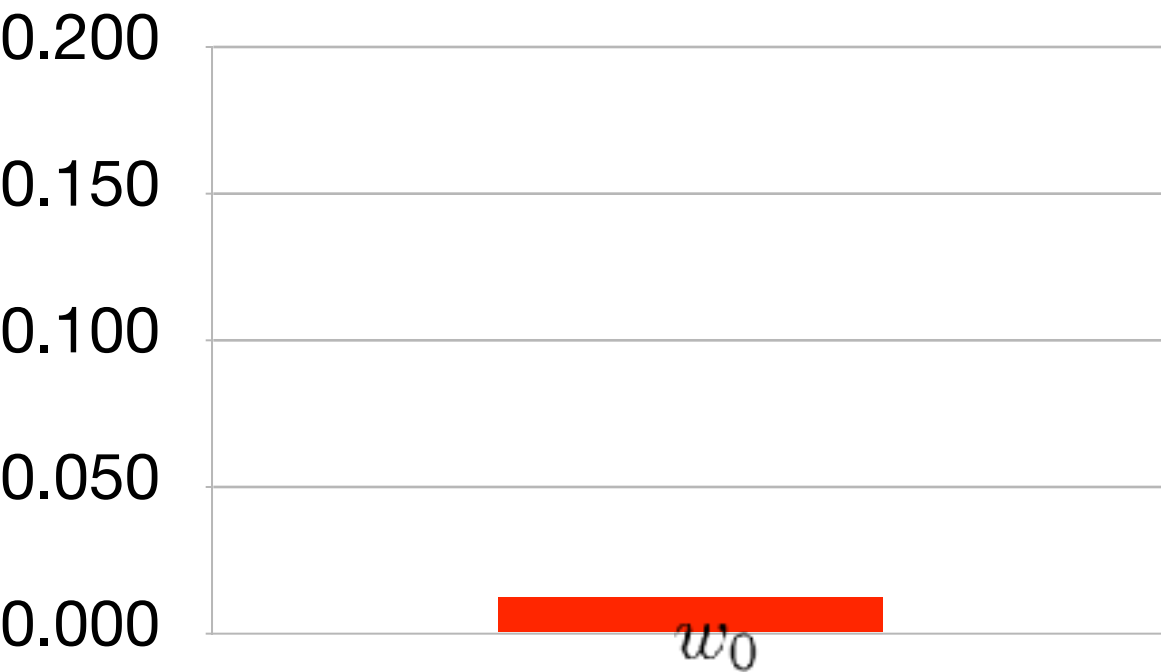


$$N = 100$$

$$M = 10$$



The Resulting Model Parameters



Observations

- The higher the model complexity grows, the better is the fit to the data
- If the model complexity is too high, all data points are explained well, but the resulting model oscillates very much. It can not generalize well. This is called *overfitting*.
- By increasing the size of the data set (number of samples), we obtain a better fit of the model
- More complex models have larger parameters

Problem: How can we find a good model complexity for a given data set with a fixed size?



Regularization

We observed that complex models yield large parameters, leading to oscillation. Idea:

Minimize the error function and the magnitude of the parameters simultaneously

We do this by adding a regularization term :

$$\tilde{E}(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (\mathbf{w}^T \phi(x_i) - t_i)^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

where λ rules the influence of the regularization.



Regularization

As above, we set the derivative to zero:

$$\nabla \tilde{E}(\mathbf{w}) = \sum_{i=1}^N (\mathbf{w}^T \phi(x_i) - t_i) \phi(x_i)^T + \lambda \mathbf{w}^T \doteq \mathbf{0}^T$$

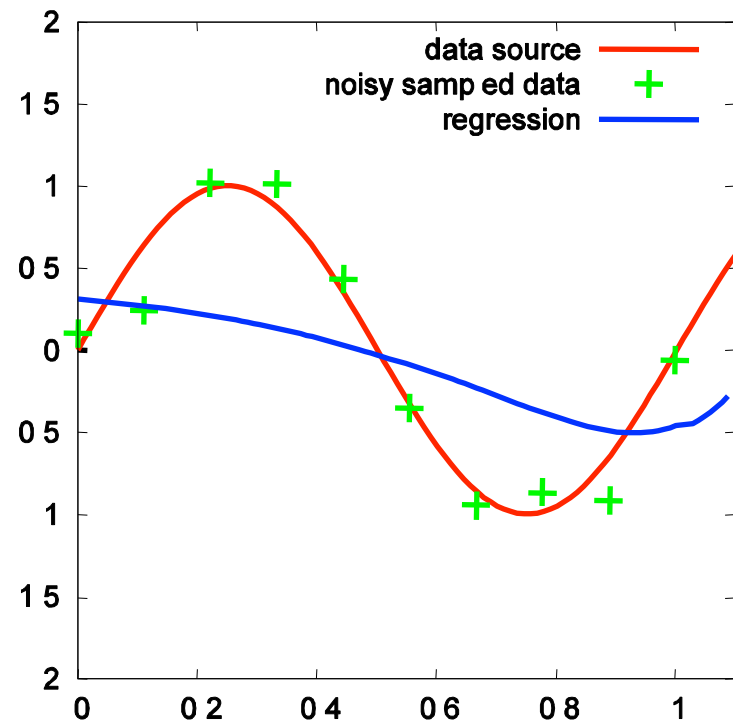
$$\mathbf{w}^T \Phi^T \Phi + \lambda \mathbf{w}^T = \mathbf{t}^T \Phi \quad \Rightarrow \quad (\lambda I + \Phi^T \Phi) \mathbf{w} = \Phi^T \mathbf{t}$$

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

With regularization, we can find a complex model for a small data set. However, the problem now is to find an appropriate regularization coefficient λ .



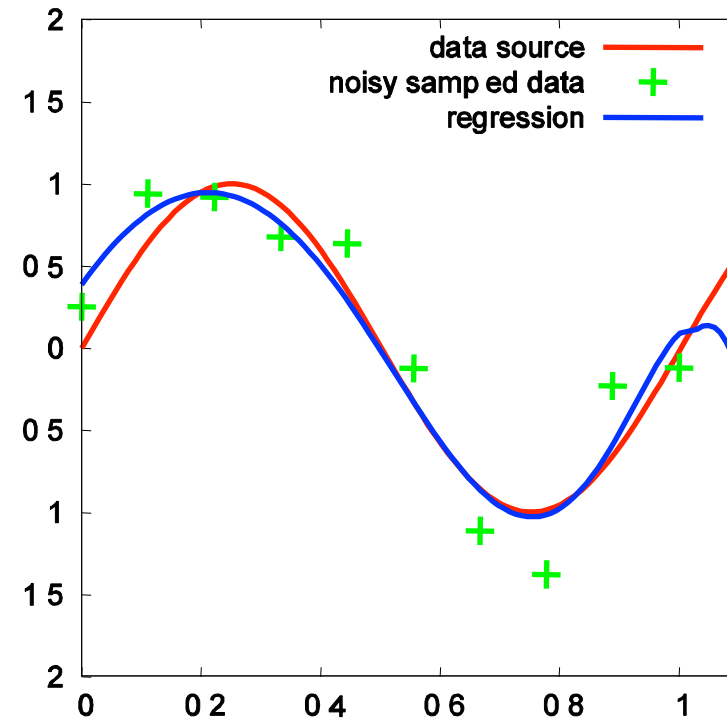
Regularized Results



$$N = 10$$

$$M = 10$$

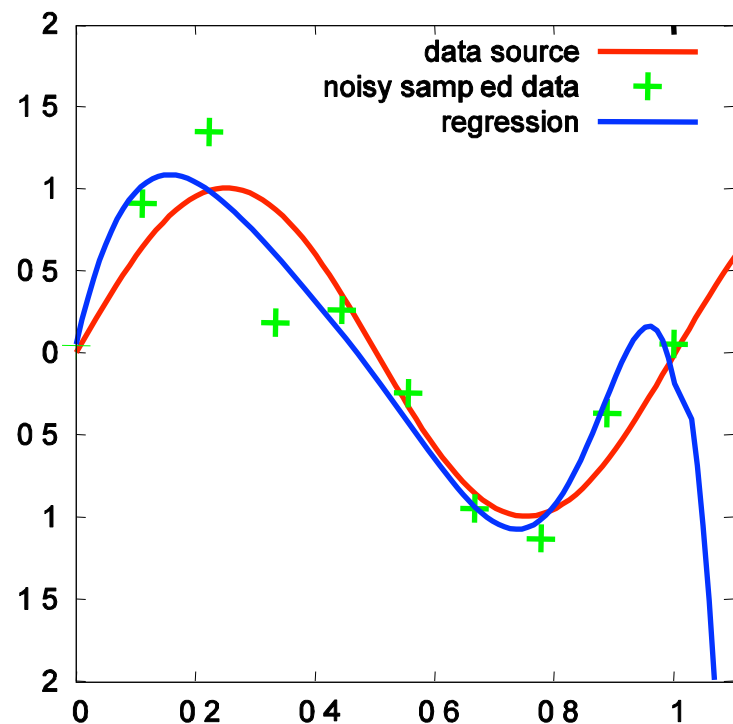
$$\lambda = 1$$



$$N = 10$$

$$M = 10$$

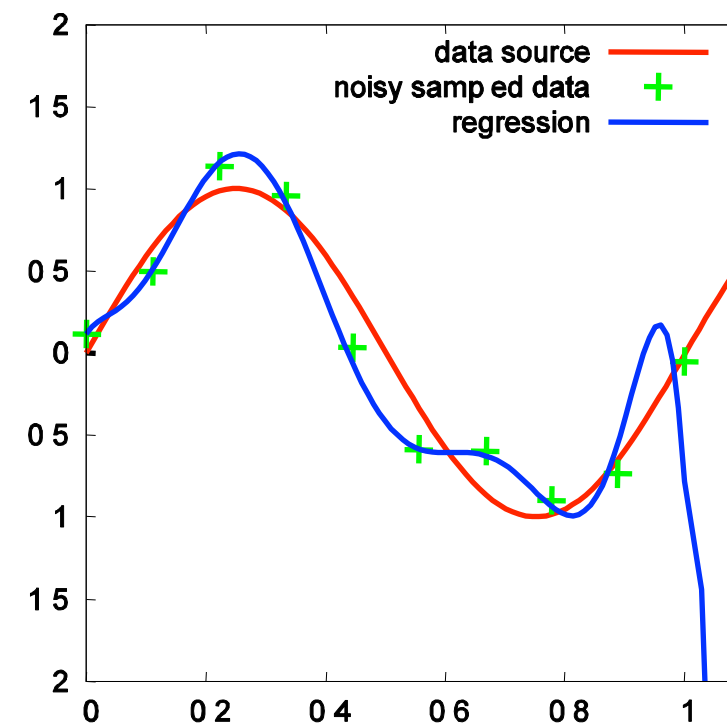
$$\lambda = 10^{-3}$$



$$N = 10$$

$$M = 10$$

$$\lambda = 10^{-6}$$



$$N = 10$$

$$M = 10$$

$$\lambda = 10^{-11}$$

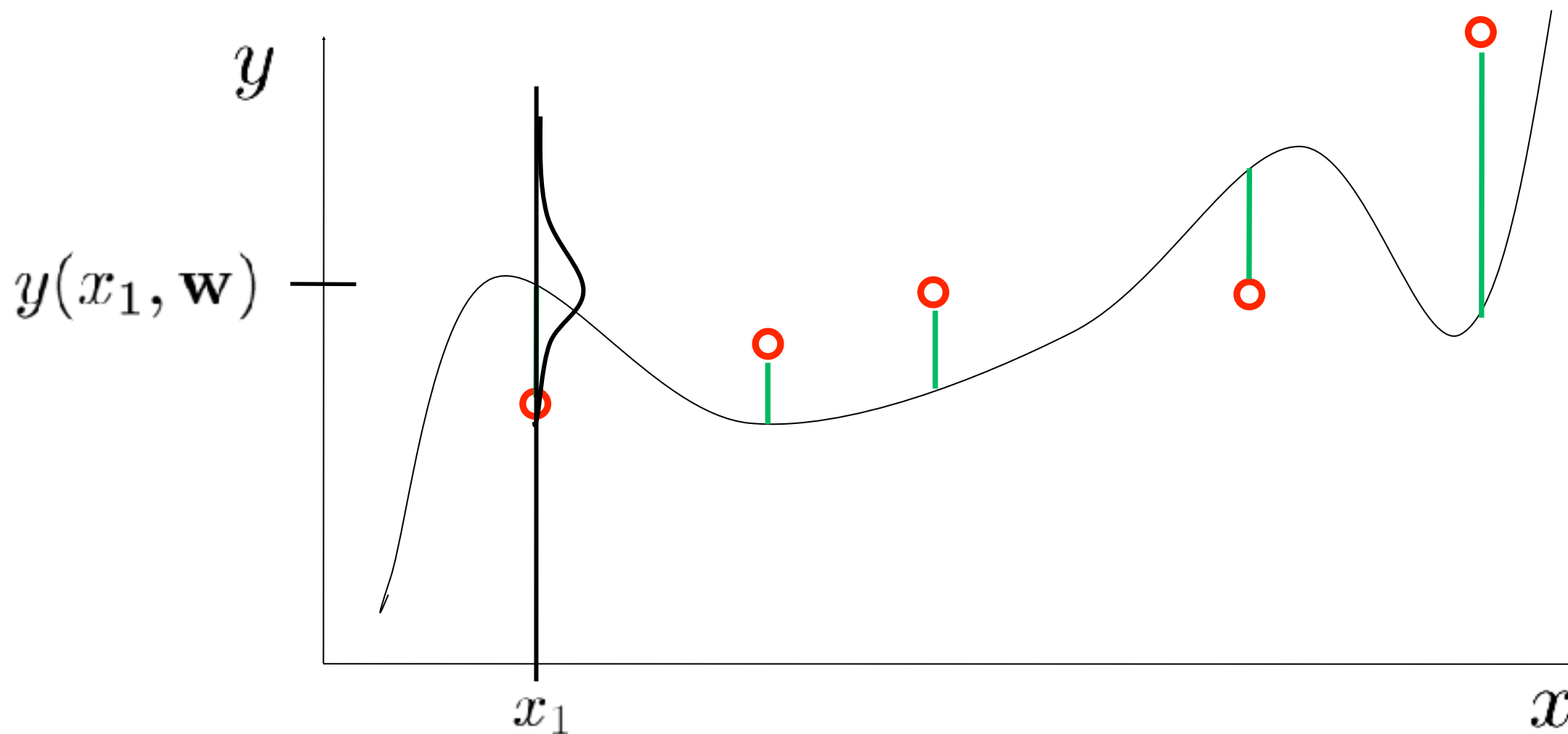


The Problem from a Different View Point

Assume that y is affected by Gaussian noise :

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

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



Maximum Likelihood Estimation

Aim: we want to find the $\mathbf{w}$ that maximizes p .

$p(t \mid x, \mathbf{w}, \sigma)$ is the *likelihood* of the measured data given a model. Intuitively:

Find parameters $\mathbf{w}$ that maximize the probability of measuring the already measured data t .

“Maximum Likelihood Estimation”

We can think of this as fitting a model $\mathbf{w}$ to the data t .

Note: σ is also part of the model and can be estimated.

For now, we assume σ is known.



Maximum Likelihood Estimation

Given data points: $(x_1, t_1), (x_2, t_2), \dots, (x_N, t_N)$

Assumption: points are drawn independently from p :

$$\begin{aligned} p(\mathbf{t} \mid \mathbf{x}, \mathbf{w}, \sigma) &= \prod_{i=1}^N p(t_i \mid x_i, \mathbf{w}, \sigma) \\ &= \prod_{i=1}^N \mathcal{N}(t_i; \mathbf{w}^T \phi(x_i), \sigma^2) \end{aligned}$$

where:

$$\mathbf{x} = (x_1, x_2, \dots, x_N)$$

$$\mathbf{t} = (t_1, t_2, \dots, t_N)$$

Instead of maximizing p we
can also maximize its
logarithm (monotonicity of
the logarithm)



Maximum Likelihood Estimation

$$\begin{aligned}
 \ln p(\mathbf{t} \mid \mathbf{x}, \mathbf{w}, \sigma) &= \sum_{i=1}^N \ln p(t_i \mid x_i, \mathbf{w}, \sigma) \\
 &= \frac{1}{2} \sum_{i=1}^N -\ln(\sigma^2) - \ln(2\pi) - \frac{1}{\sigma^2} (\mathbf{w}^T \phi(x_i) - t_i)^2 \\
 &= \underbrace{\frac{-N(\ln(\sigma^2) + \ln(2\pi))}{2}}_{\text{Constant for all } \mathbf{w}} - \frac{1}{2\sigma^2} \underbrace{\sum_{i=1}^N (\mathbf{w}^T \phi(x_i) - t_i)^2}_{\text{Is equal to } E(\mathbf{w})}
 \end{aligned}$$

$\mathcal{N} \rightarrow \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2} \frac{(x-\mu)^2}{\sigma^2}}$

The parameters that maximize the likelihood are equal to the minimum of the sum of squared errors



Maximum Likelihood Estimation

$$\begin{aligned}\ln p(\mathbf{t} \mid \mathbf{x}, \mathbf{w}, \sigma) &= \sum_{i=1}^N \ln p(t_i \mid x_i, \mathbf{w}, \sigma) \\ &= \frac{1}{2} \sum_{i=1}^N -\ln(\sigma^2) - \ln(2\pi) - \frac{1}{\sigma^2} (\mathbf{w}^T \phi(x_i) - t_i)^2 \\ &= \frac{-N(\ln(\sigma^2) + \ln(2\pi))}{2} - \frac{1}{\sigma^2} \sum_{i=1}^N (\mathbf{w}^T \phi(x_i) - t_i)^2\end{aligned}$$

$$\mathcal{N} \rightarrow \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2} \frac{(x-\mu)^2}{\sigma^2}}$$

$$\mathbf{w}_{ML} := \arg \max_{\mathbf{w}} \ln p(\mathbf{t} \mid \mathbf{x}, \mathbf{w}, \sigma) = \arg \min_{\mathbf{w}} E(\mathbf{w}) = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{t}$$

The ML solution is obtained using the Pseudoinverse

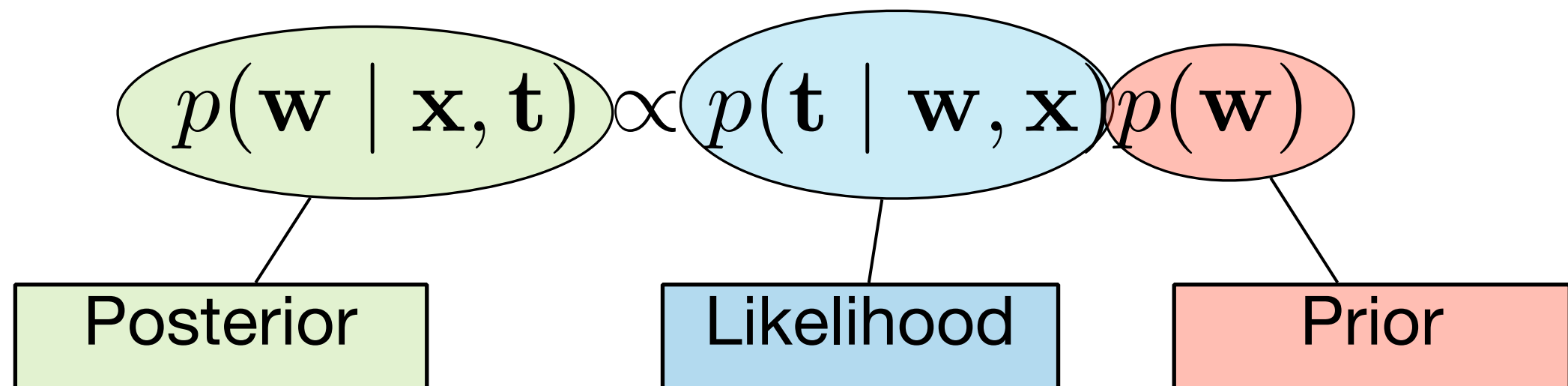


Maximum A-Posteriori Estimation

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):



“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

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

$$\text{const.} - \frac{1}{2\sigma_1^2} \sum_{i=1}^N (\mathbf{w}^T \phi(x) - t_i)^2$$

$$\text{const.} - \frac{1}{2\sigma_2^2} \mathbf{w}^T \mathbf{w}$$

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

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:

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

The same in vector notation:

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

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

**“Feature
Matrix”**



Summary: MAP Estimation

To summarize, we have the following optimization problem:

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

The same in vector notation:

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

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

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:

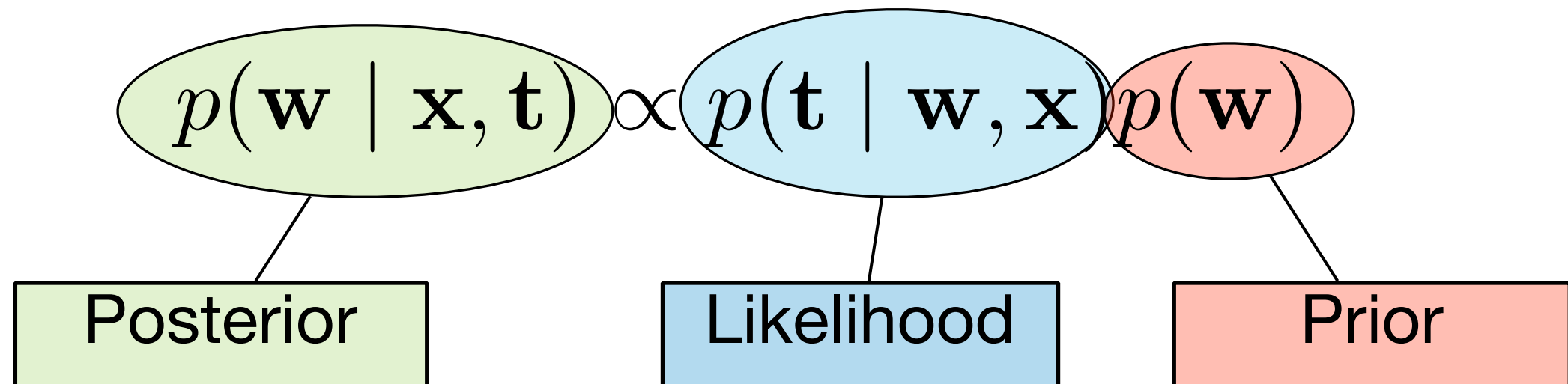
If the prior and the likelihood are Gaussian then the **posterior** and the **normalizer** are also Gaussian and we can compute them in closed form.

This gives us a natural way to compute uncertainty!



The Posterior Distribution

Remember Bayes Rule:



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

So the entire posterior distribution is

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

Note: So far we
only used the
training data!

$(\mathbf{x}, \mathbf{t})$



The Predictive Distribution

We obtain the **predictive distribution** by integrating over all possible model parameters (“inference”):

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

Test data

Test data likelihood

Parameter posterior

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

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

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

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



The Predictive Distribution

We obtain the **predictive distribution** by integrating over all possible model parameters (“inference”):

$$p(\underline{t}^* | x^*, \underline{\mathbf{t}}, \underline{\mathbf{x}}) = \int \underbrace{p(\underline{t}^* | x^*, \underline{\mathbf{w}})}_{\text{Test data likelihood}} \underbrace{p(\underline{\mathbf{w}} | \underline{\mathbf{x}}, \underline{\mathbf{t}})}_{\text{Parameter posterior}} d\underline{\mathbf{w}}$$

Test data

Test data likelihood

Parameter posterior

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

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

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

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

MAP solution



The Predictive Distribution

Using formula III. from above (linear Gaussian),

$$\begin{aligned} p(t^* | x^*, \mathbf{t}, \mathbf{x}) &= \int p(t^* | x^*, \mathbf{w}) p(\mathbf{w} | \mathbf{x}, \mathbf{t}) d\mathbf{w} \\ &= \int \mathcal{N}(t^*; \phi(x^*)^T \mathbf{w}, \sigma) \mathcal{N}(\mathbf{w}; \boldsymbol{\mu}, \Sigma) d\mathbf{w} \\ &= \mathcal{N}(t^*; \phi(x^*)^T \boldsymbol{\mu}, \sigma_N^2(x^*)) \end{aligned}$$

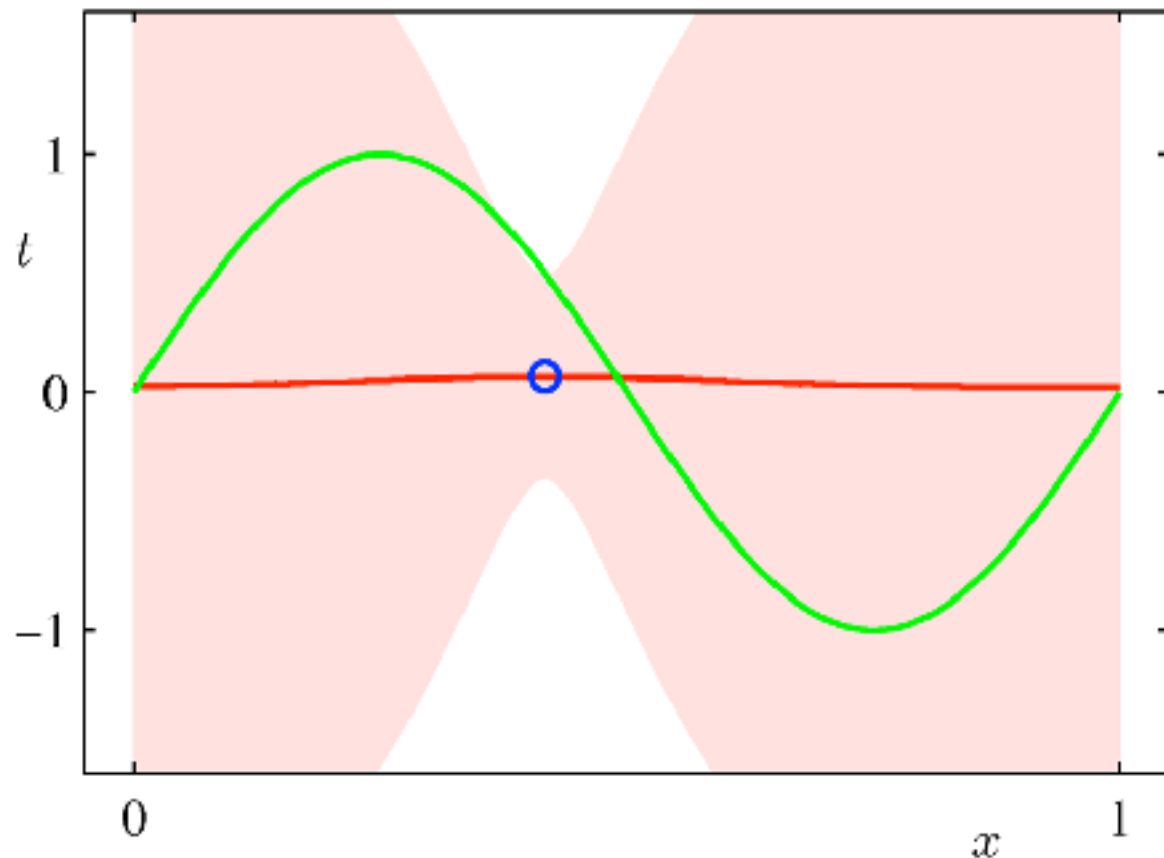
where

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

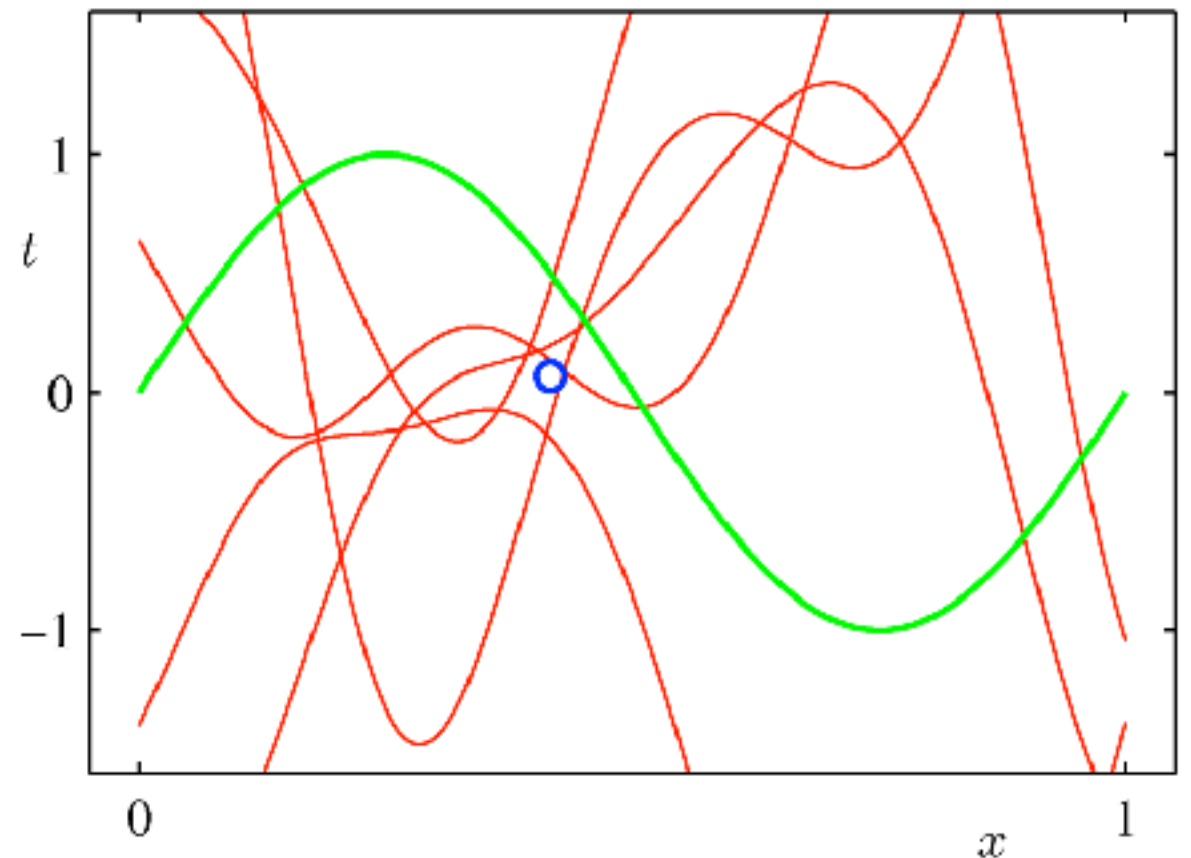


The Predictive Distribution (2)

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



The predictive distribution



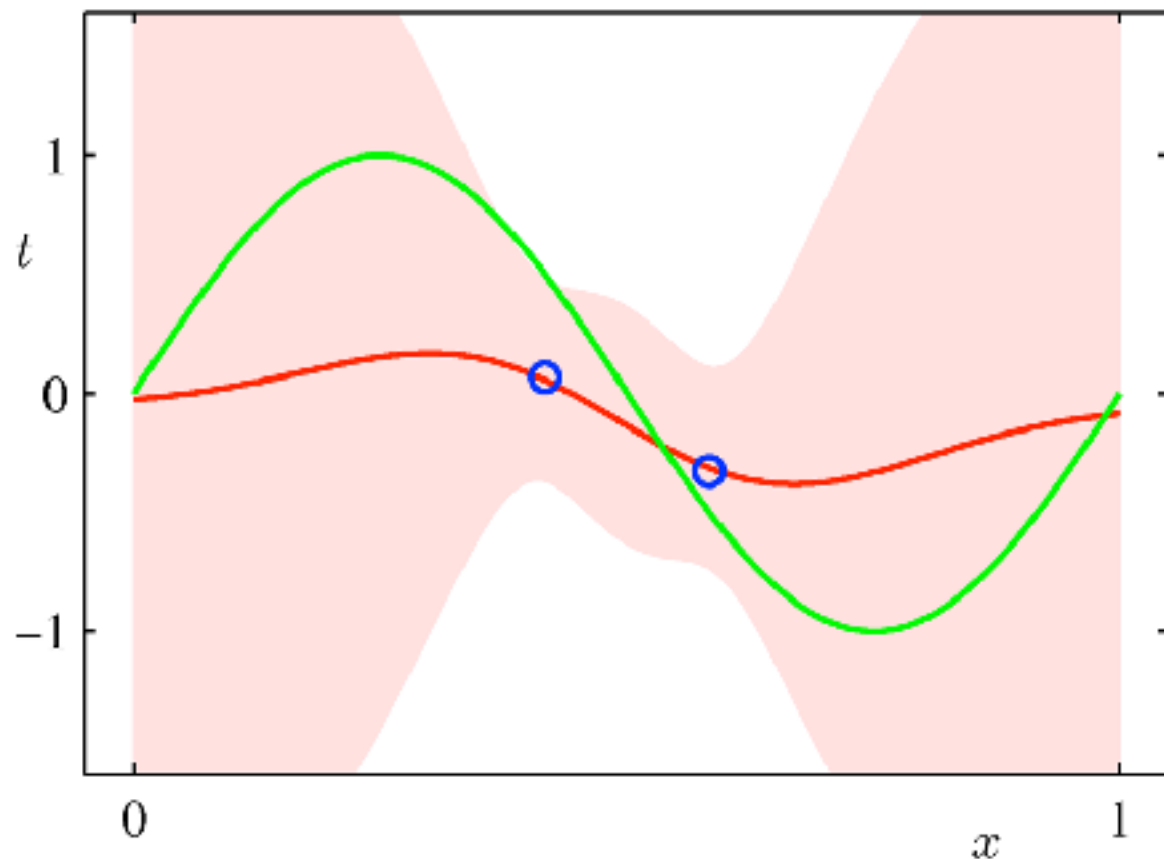
Some samples from
the posterior

From: C.M. Bishop

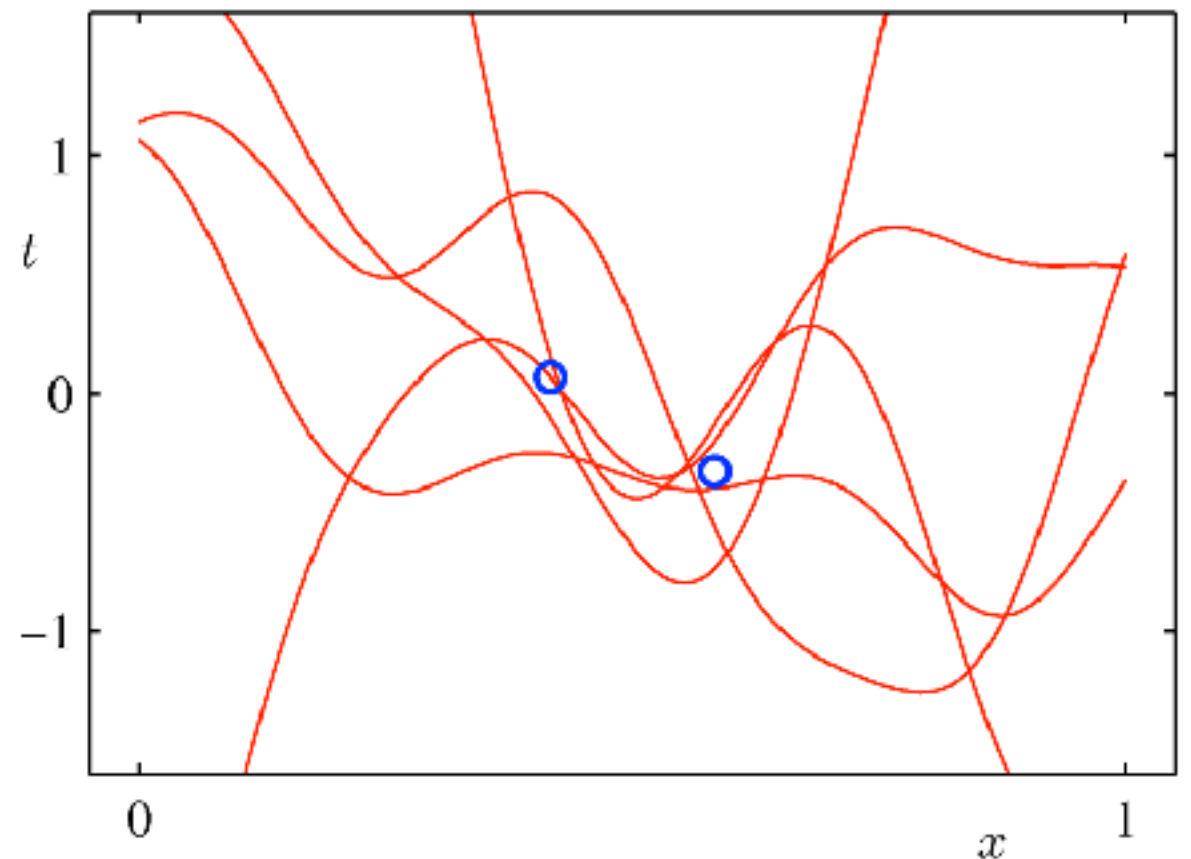


Predictive Distribution (3)

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



The predictive distribution



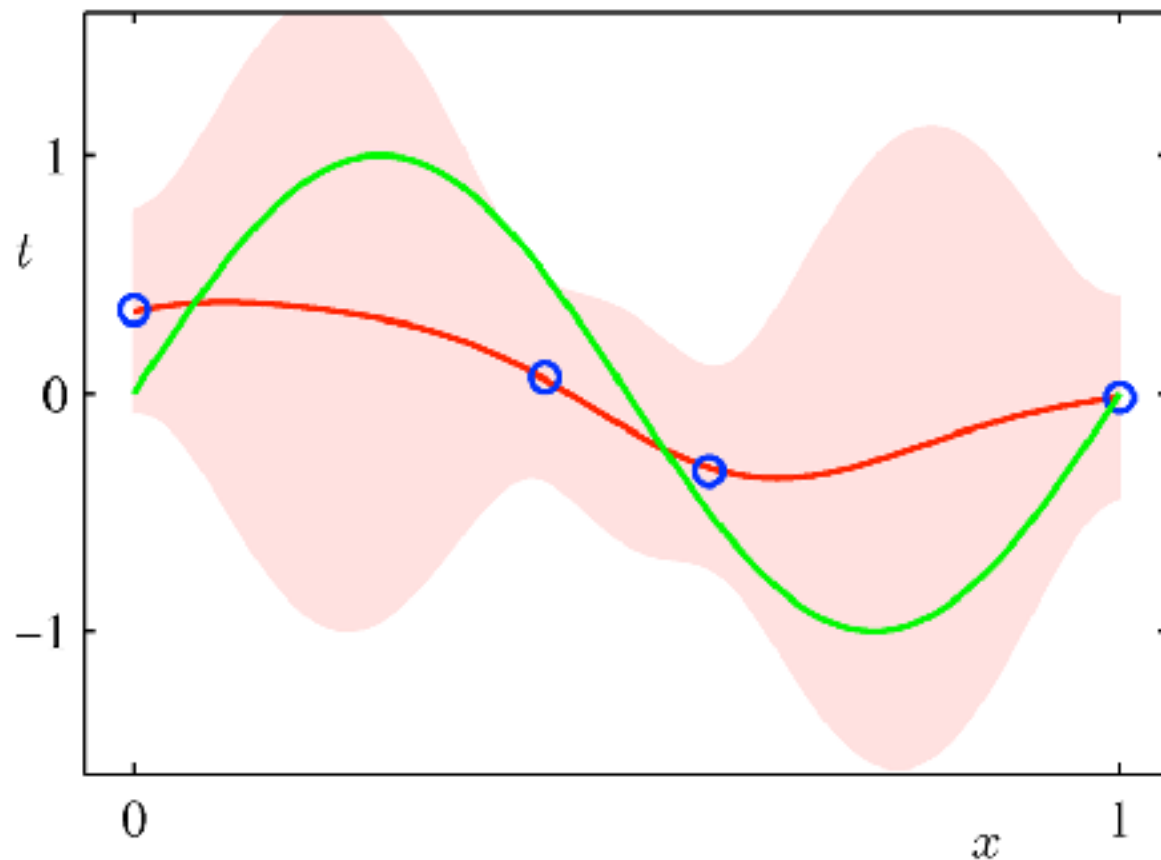
Some samples from
the posterior

From: C.M. Bishop

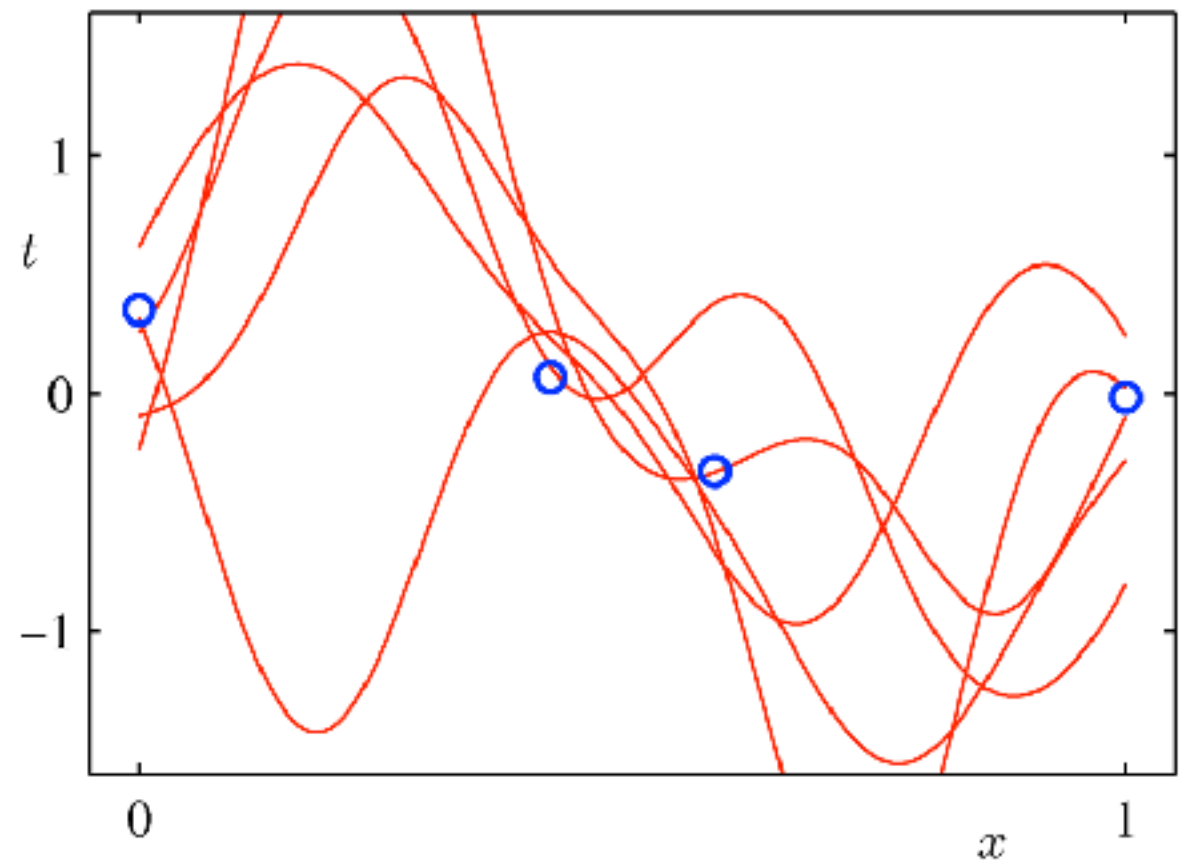


Predictive Distribution (4)

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



The predictive distribution



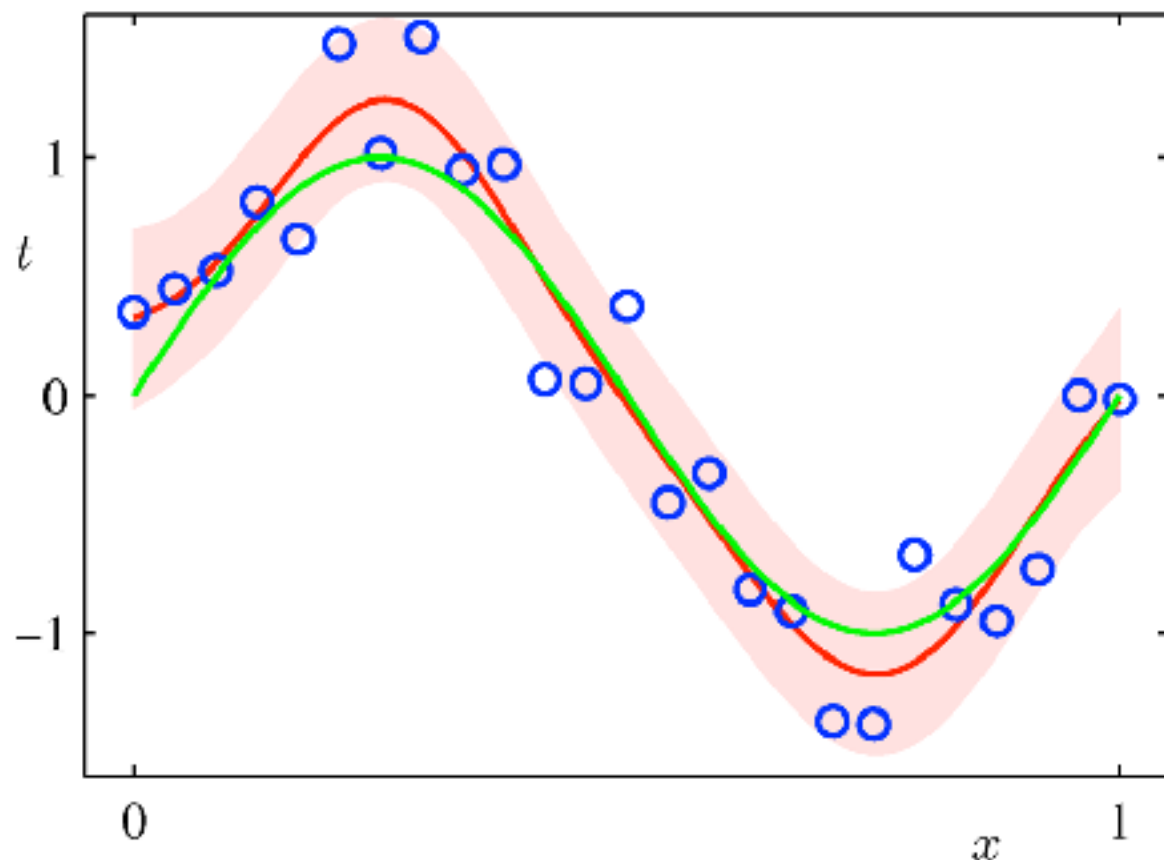
Some samples from
the posterior

From: C.M. Bishop

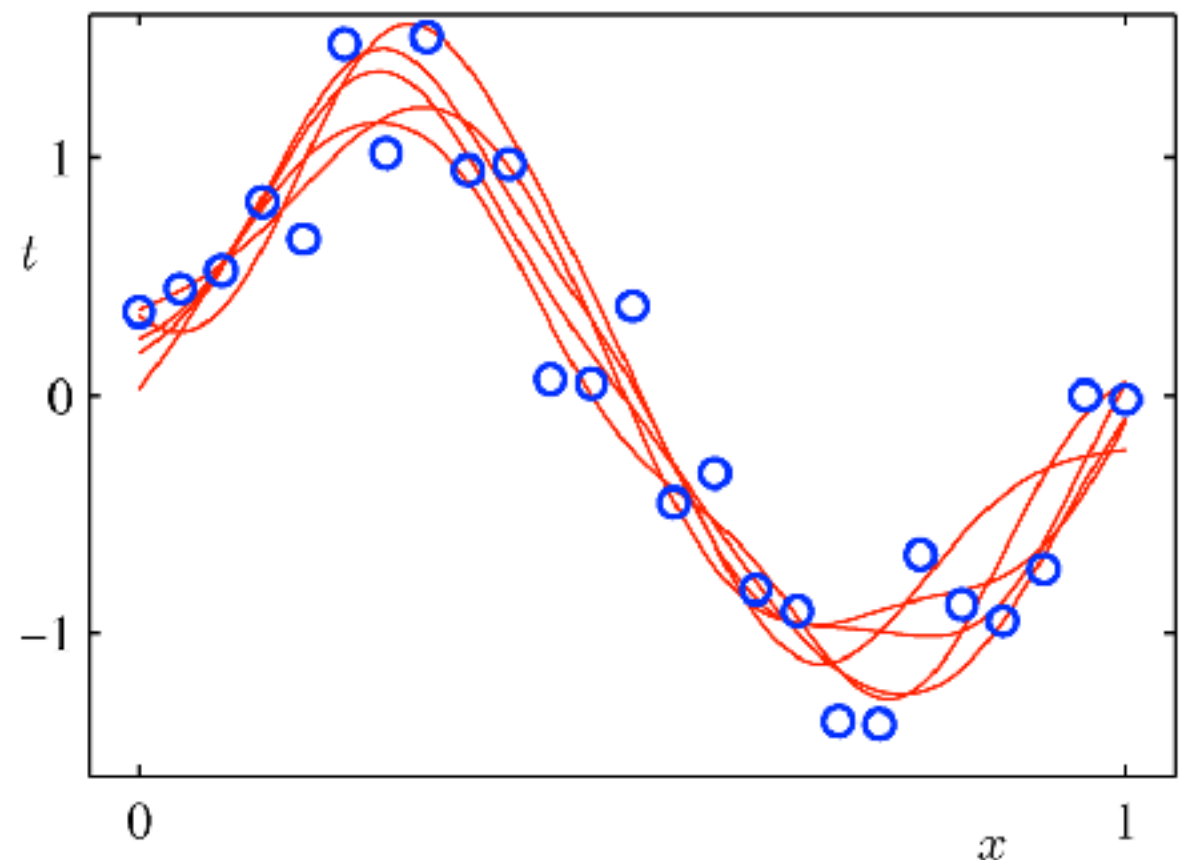


Predictive Distribution (5)

- 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**

