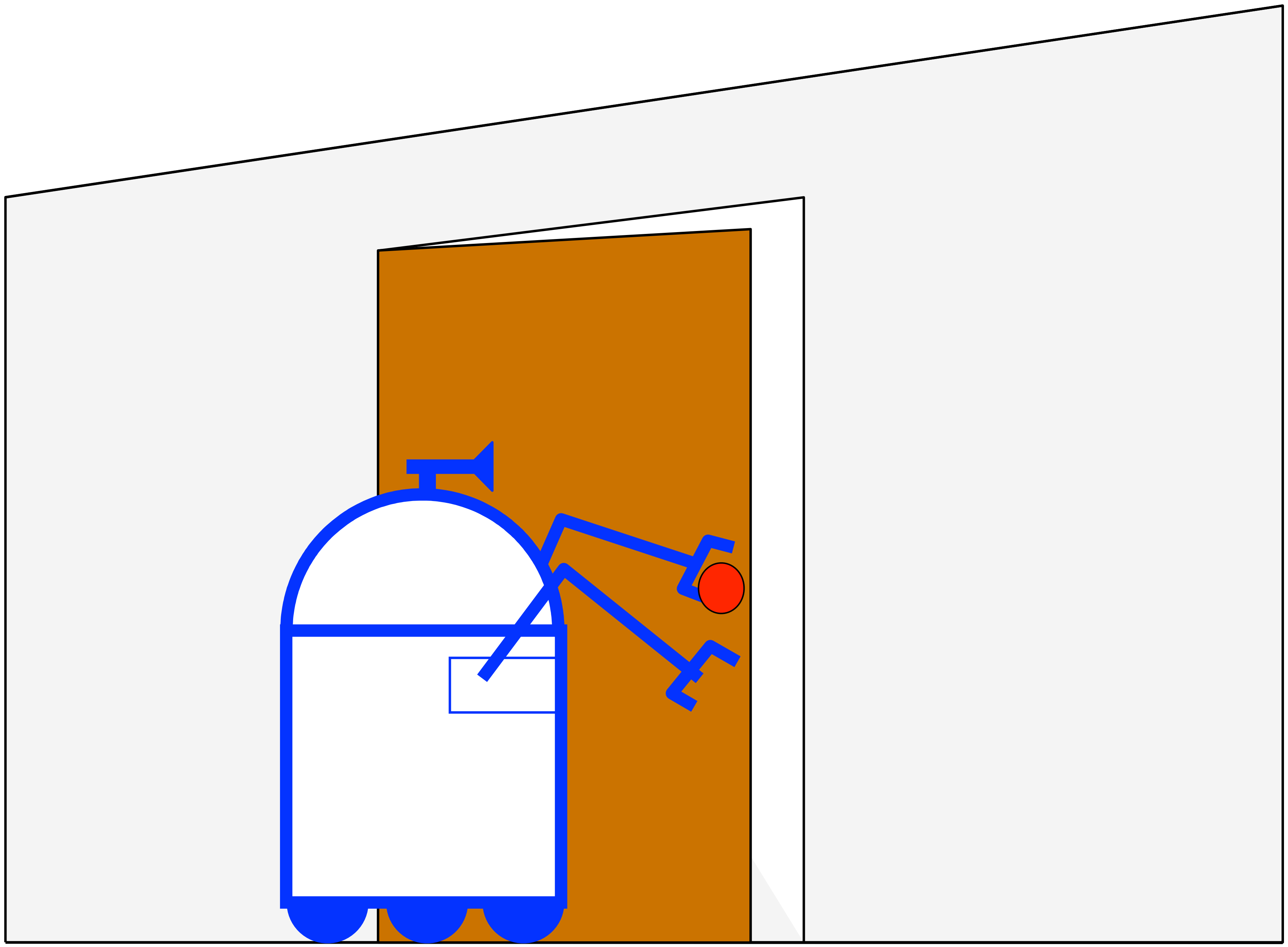
Example: Sensing and Acting

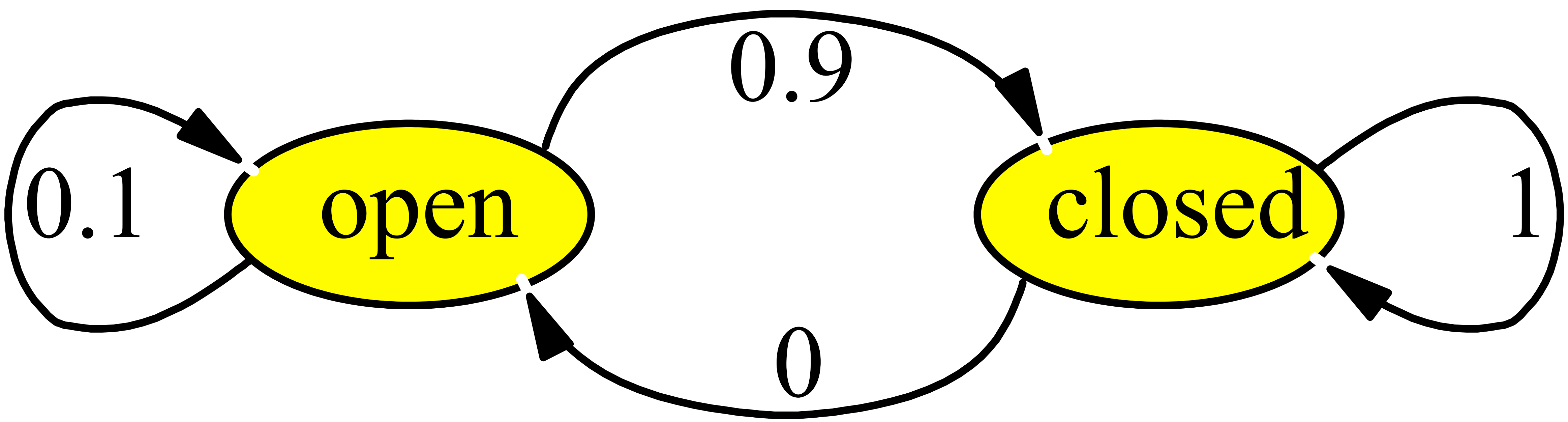
Now the robot ***senses*** the door state and ***acts***
(it opens or closes the door).



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:

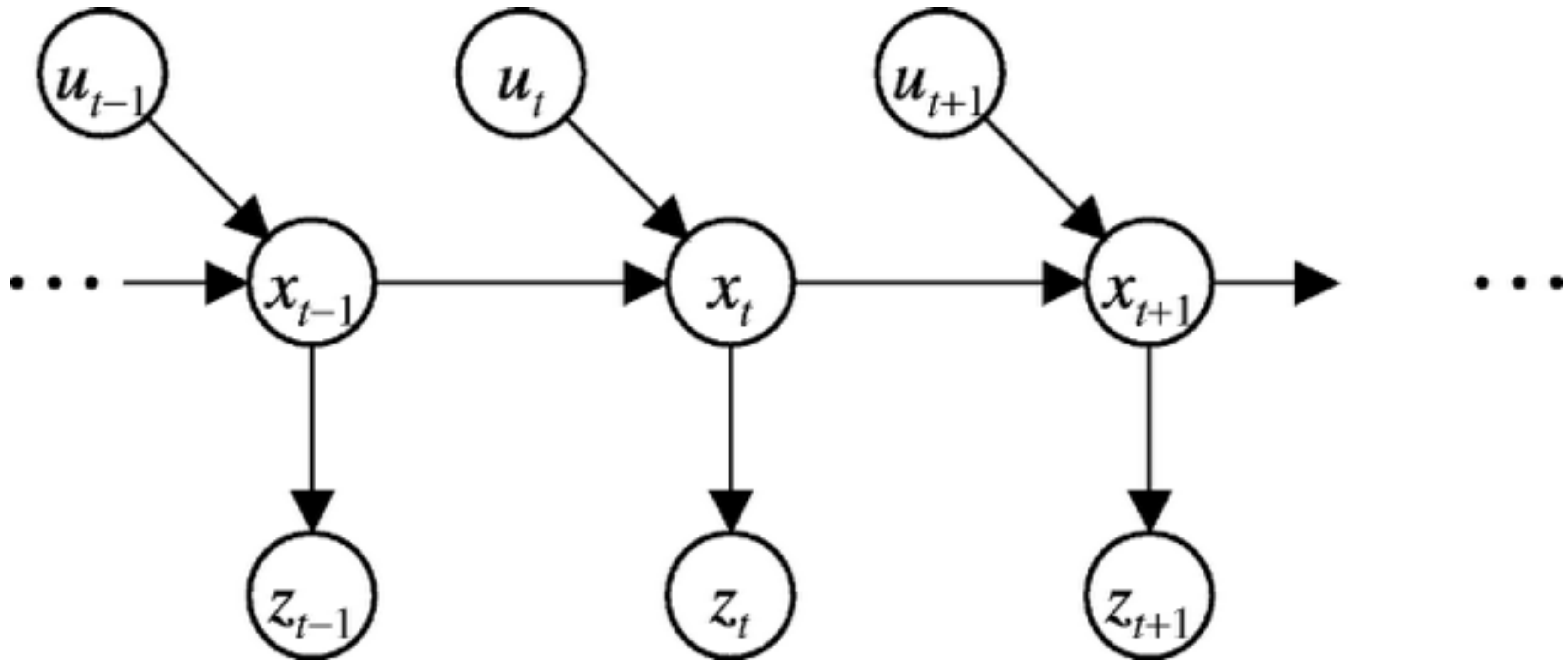
- Sensor update: $p(x \mid z)$
- Action update: $p(x \mid u)$
- Now we want to combine both:

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

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

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

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

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

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

The Bayes Filter Algorithm

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

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

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

Bayes Filter Variants

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

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.)

Learning

Unsupervised Learning

clustering, density estimation

Supervised Learning

learning from a training data set, **inference** on the test data

Reinforcement Learning

no supervision, but a **reward function**

Regression

target set is **continuous**, e.g.

$$\mathcal{Y} = \mathbb{R}$$

Classification

target set is **discrete**, e.g.

$$\mathcal{Y} = [1, \ldots, C]$$

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

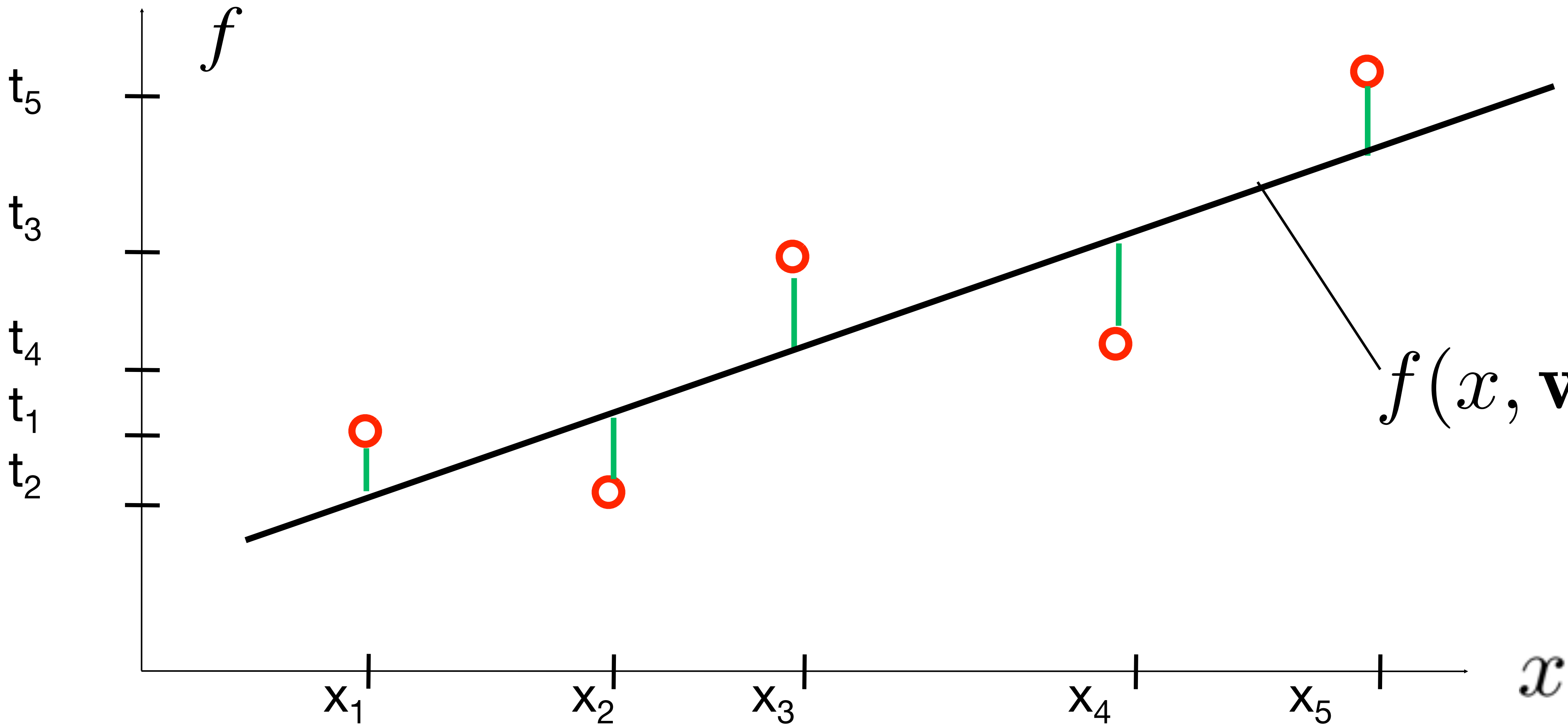
In principal, the elements of $\mathcal{X}$ can be anything (e.g. real numbers, graphs, 3D objects). To be able to treat these objects mathematically we need functions ϕ that map from $\mathcal{X}$ to $\mathbb{R}^M$. We call these the **basis functions**.

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 $(x_1, t_1), (x_2, t_2), \ldots$
- **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:

$$\nabla E(\mathbf{w}) = \sum_{i=1}^{N}(f(x_i,\mathbf{w}) - t_i)\nabla f(x_i,\mathbf{w}) \doteq (0 \quad 0)$$

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

Linear Regression

To evaluate the function y, 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:

$$\nabla E(\mathbf{w}) = \sum_{i=1}^{N}(f(x_i, \mathbf{w}) - t_i)\nabla f(x_i, \mathbf{w}) \doteq (0 \quad 0)$$

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

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

Linear Regression

To evaluate the function y, 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:

$$\nabla E(\mathbf{w}) = \sum_{i=1}^{N}(f(x_i, \mathbf{w}) - t_i)\nabla f(x_i, \mathbf{w}) \doteq (0 \quad 0)$$

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

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

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

Polynomial Regression

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

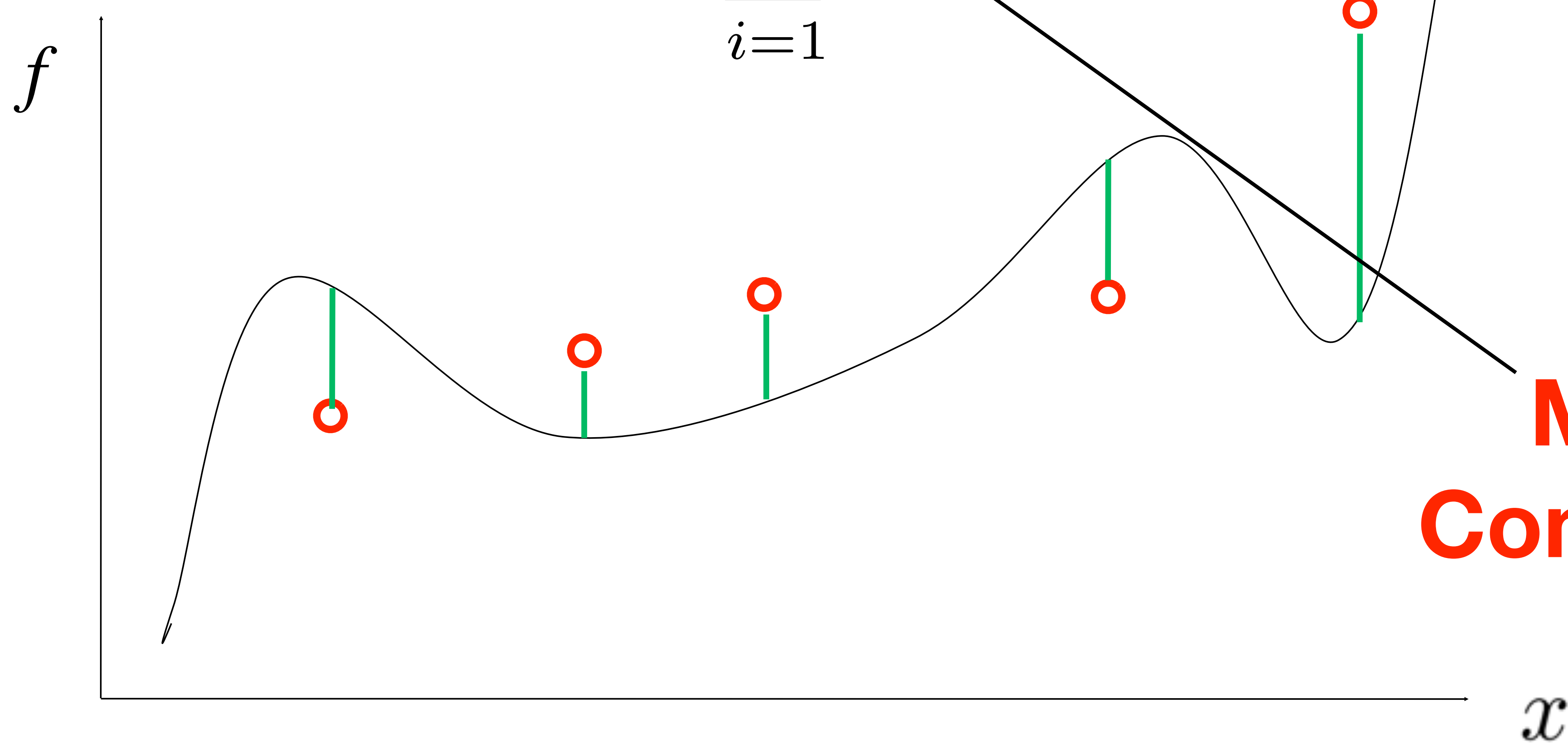
Given: data points $(x_1, t_1), (x_2, t_2), \ldots, (x_N, t_N)$

Assume we are given M **basis** functions

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

Data Set Size

Model Complexity



Polynomial Regression

We have defined:

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

“Basis functions”

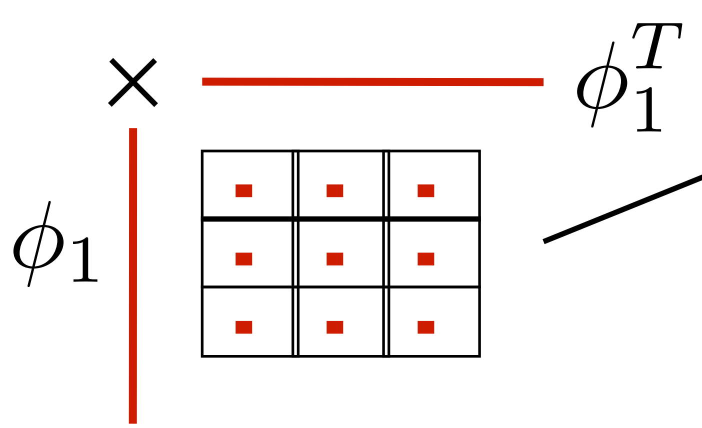
Therefore:

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

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

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

Outer Product



Polynomial Regression

We have defined:

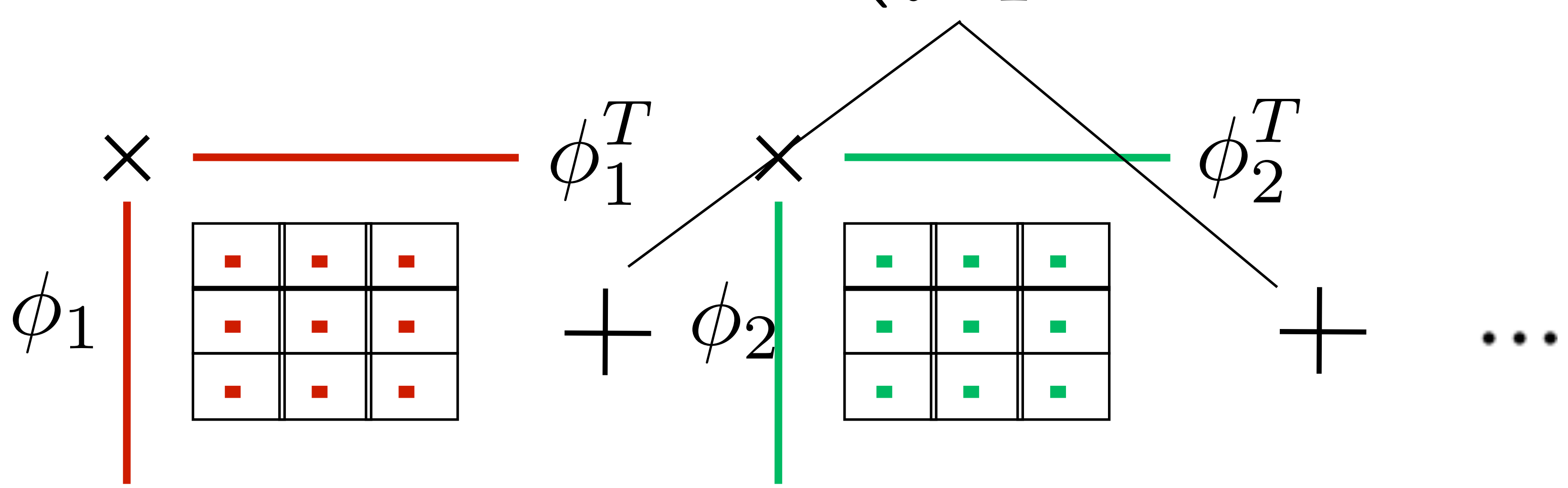
$$\boldsymbol{\phi}(x) := (1, \phi_1(x), \dots, \phi_{M-1}(x))^T$$

Therefore:

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

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

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



Polynomial Regression

We have defined:

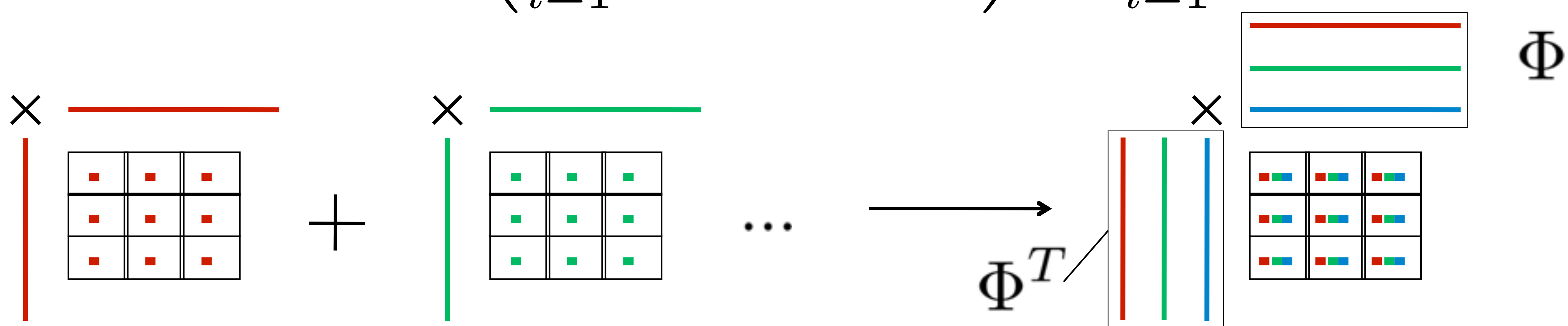
$$\phi(x) := (1, \phi_1(x), \ldots, \phi_{M-1}(x))^T$$

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

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

"Normal Equation"

It follows:

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

"Pseudoinverse" Φ^+

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.

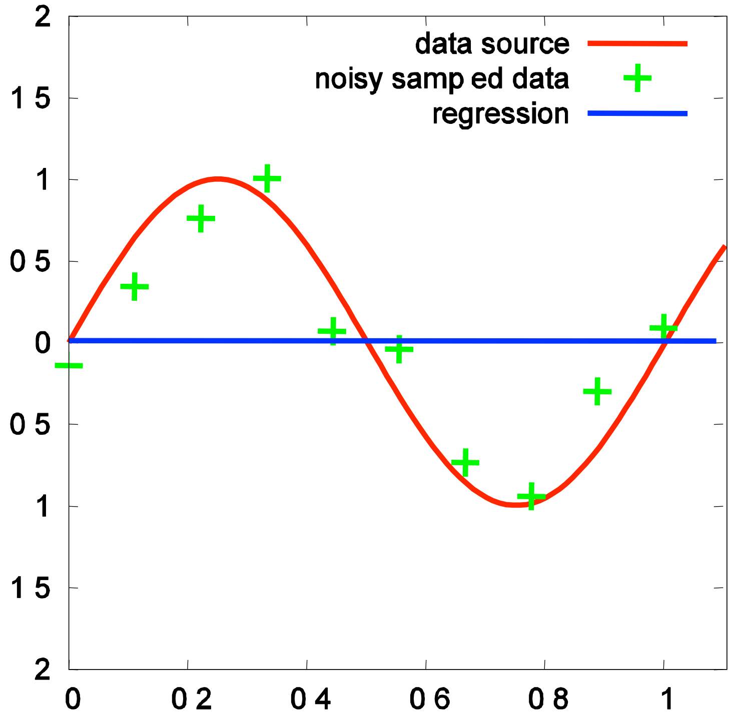
Therefore: Singular Value Decomposition (SVD) is used: $\Phi = UDV^T$ where

- 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

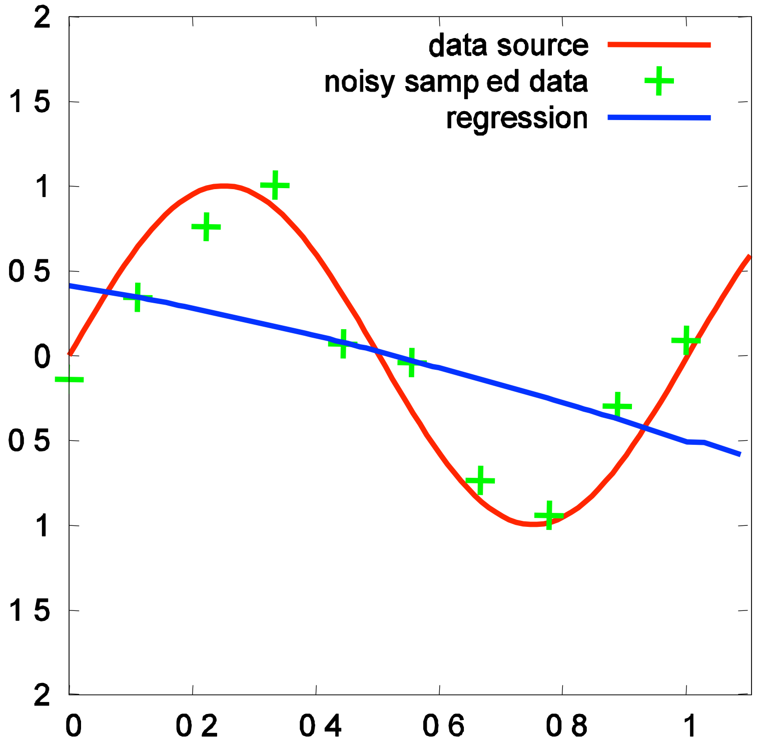
A Simple Example

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



$$N = 10$$

$$M = 1$$

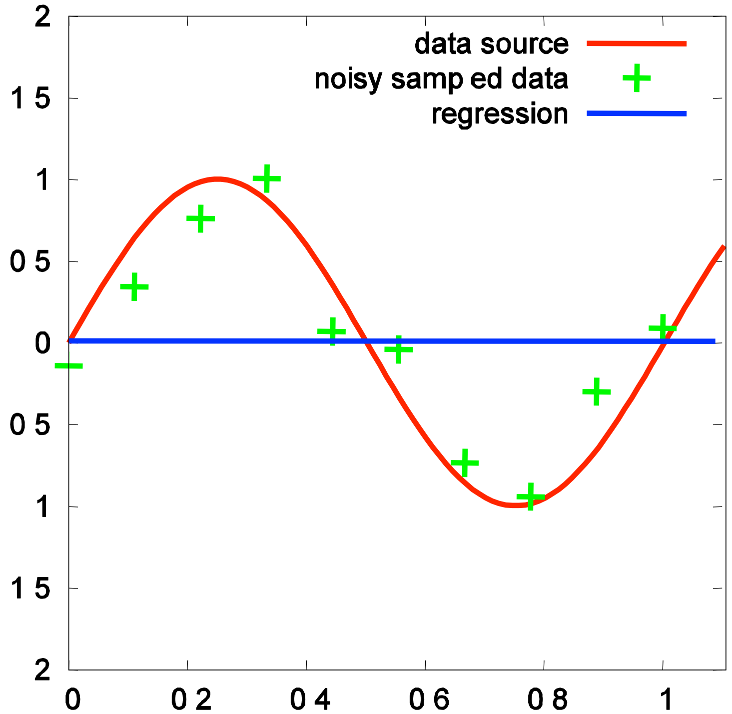


$$N = 10$$

$$M = 3$$

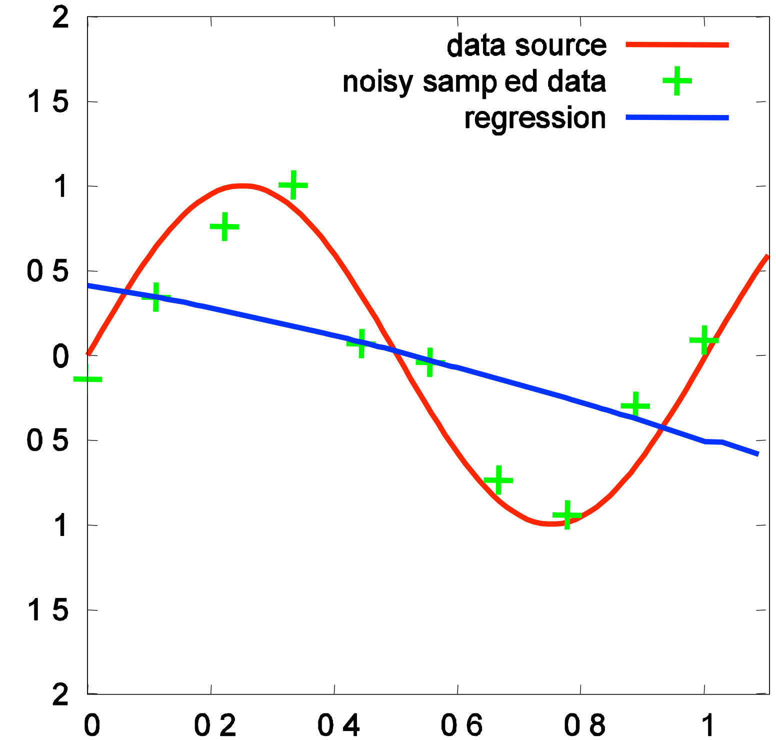
A Simple Example

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



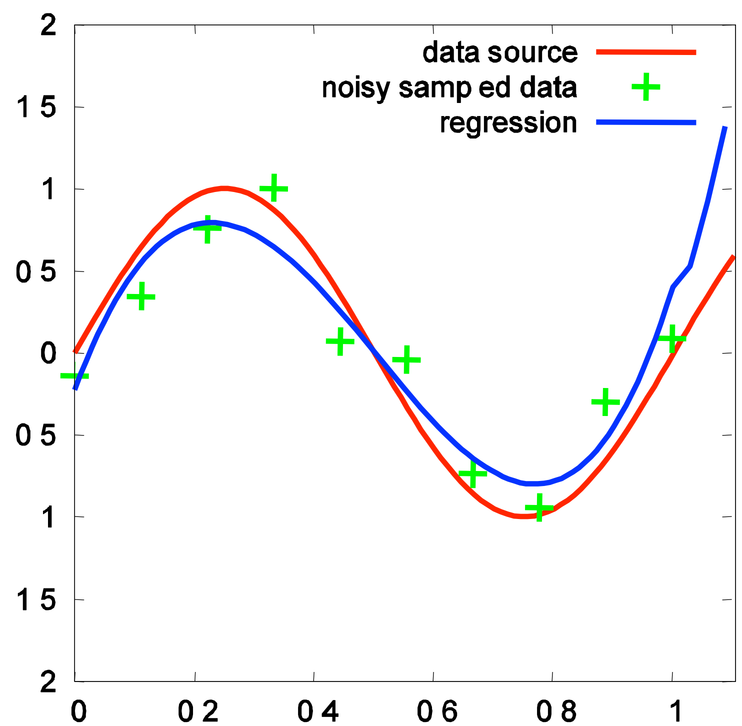
$N = 10$

$M = 1$



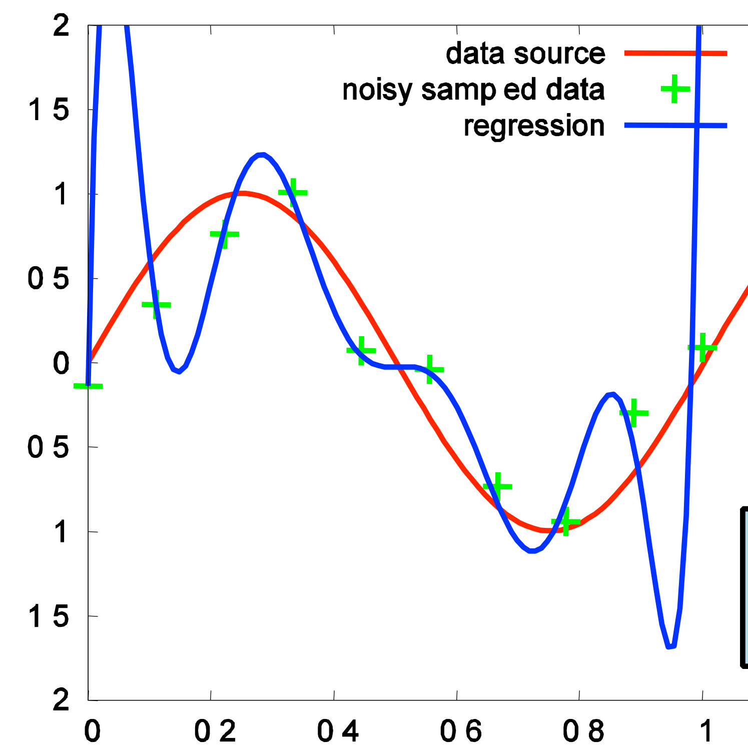
$N = 10$

$M = 3$



$N = 10$

$M = 5$

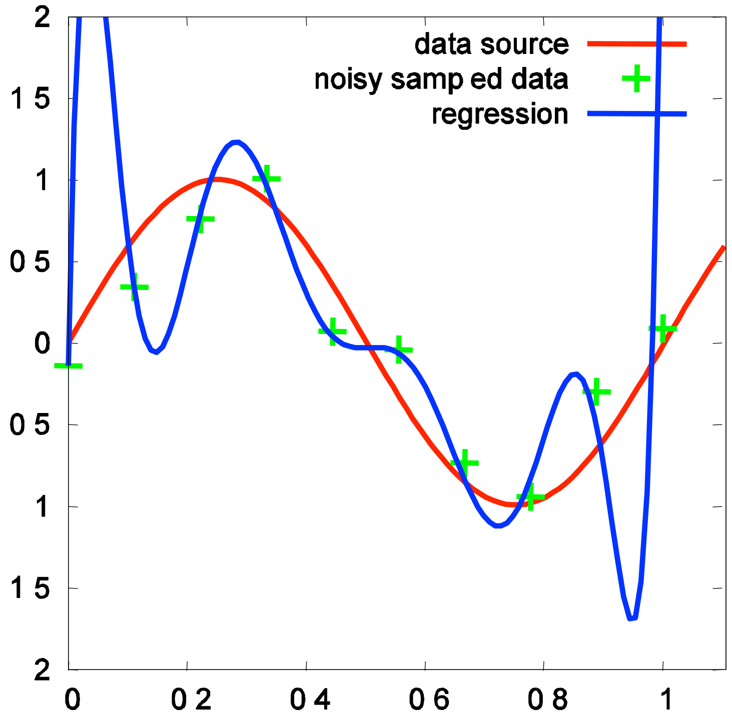


$N = 10$

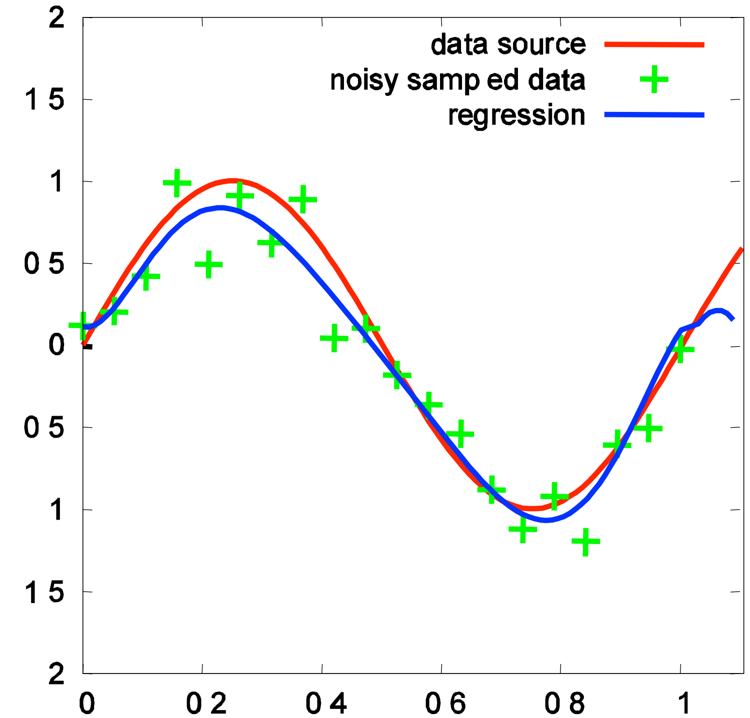
$M = 10$

“Overfitting”

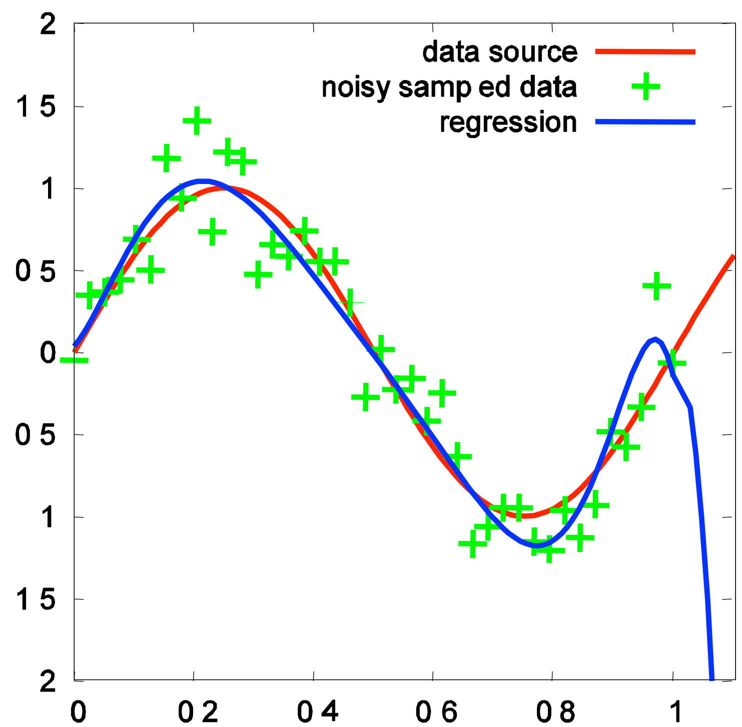
Varying the Sample Size



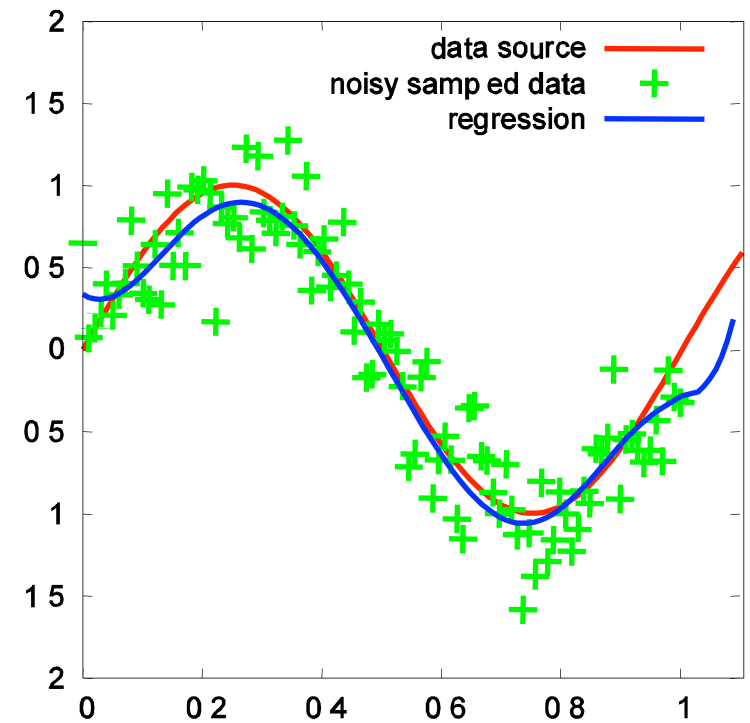
$$N = 10$$
$$M = 10$$



$$N = 20$$
$$M = 10$$

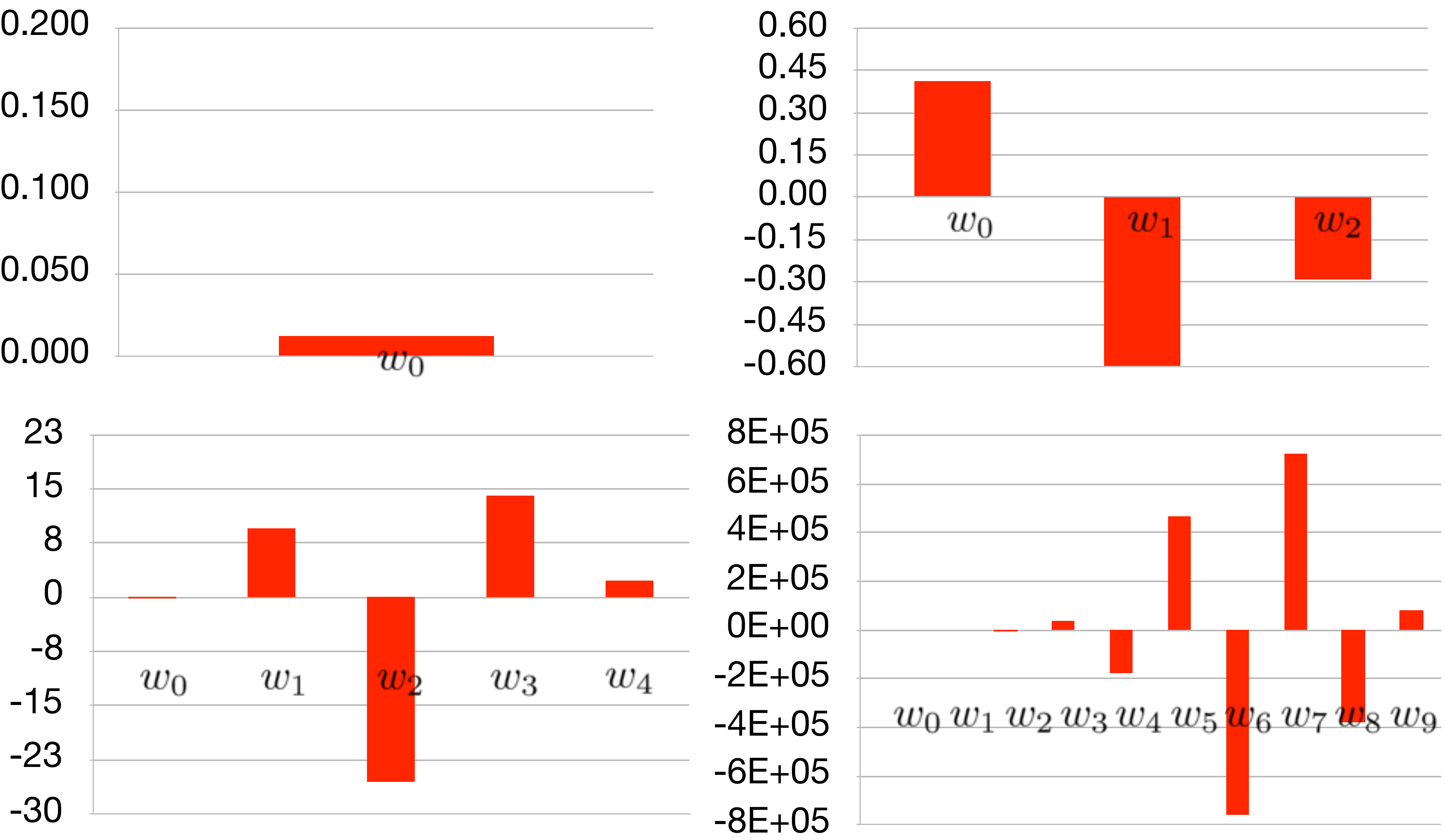


$$N = 40$$
$$M = 10$$



$$N = 100$$
$$M = 10$$

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:

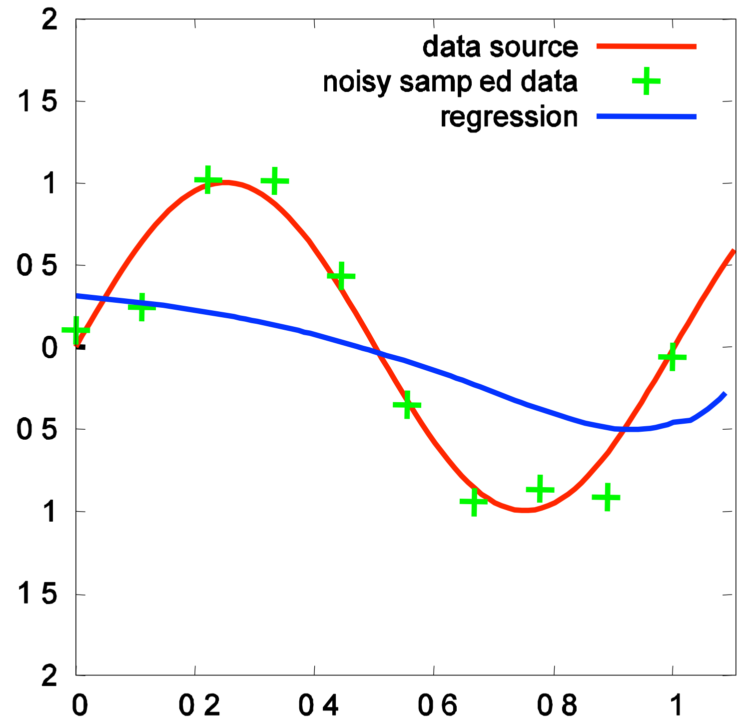
$$\nabla \tilde{E}(\mathbf{w}) = \sum_{i=1}^{N} (\mathbf{w}^T \phi(x_i) - t_i)\phi(x_i)^T + \lambda \mathbf{w}^T \doteq \mathbf{0}^T$$

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

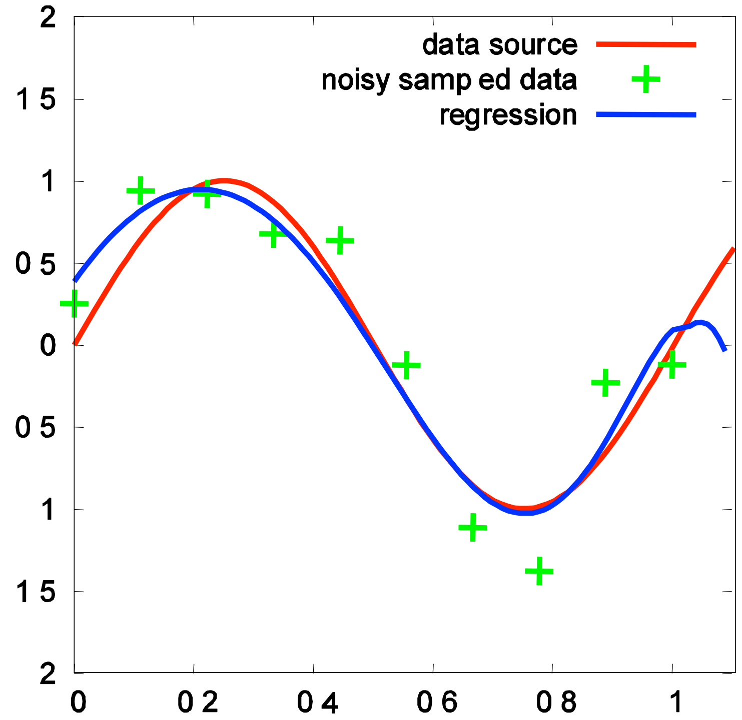
$$\mathbf{w} = (\lambda I + \Phi^T \Phi)^{-1} \Phi^T \mathbf{t}$$

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.

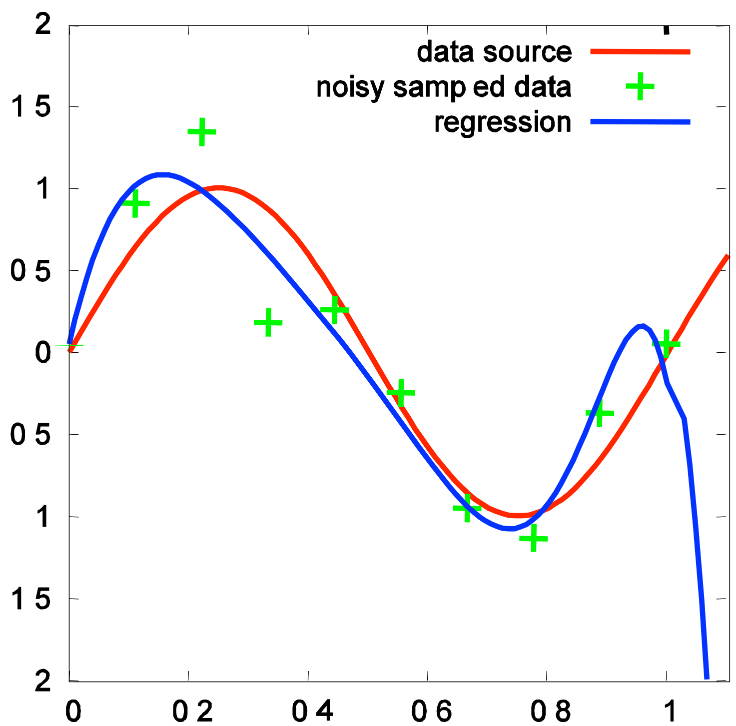
Regularized Results



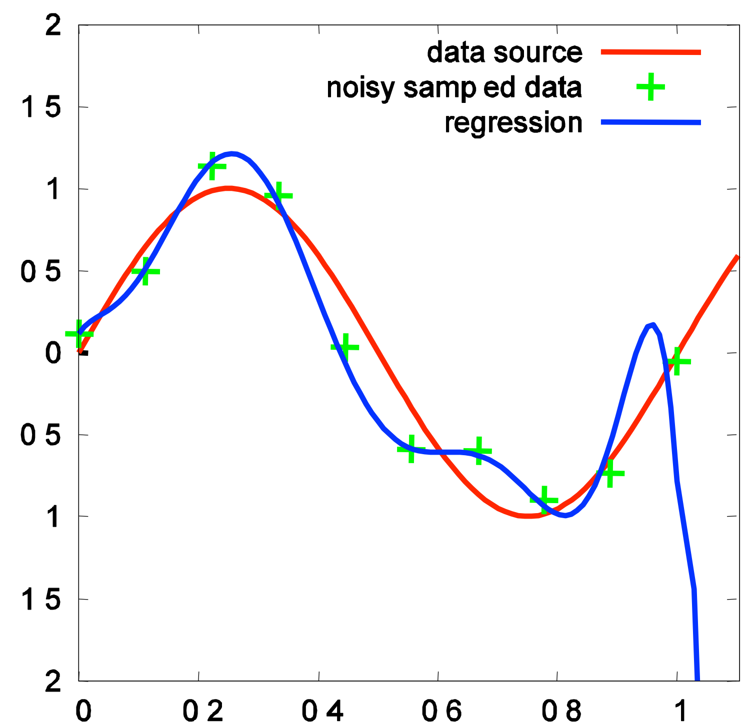
$$N = 10$$
$$M = 10$$
$$\lambda = 1$$



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



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



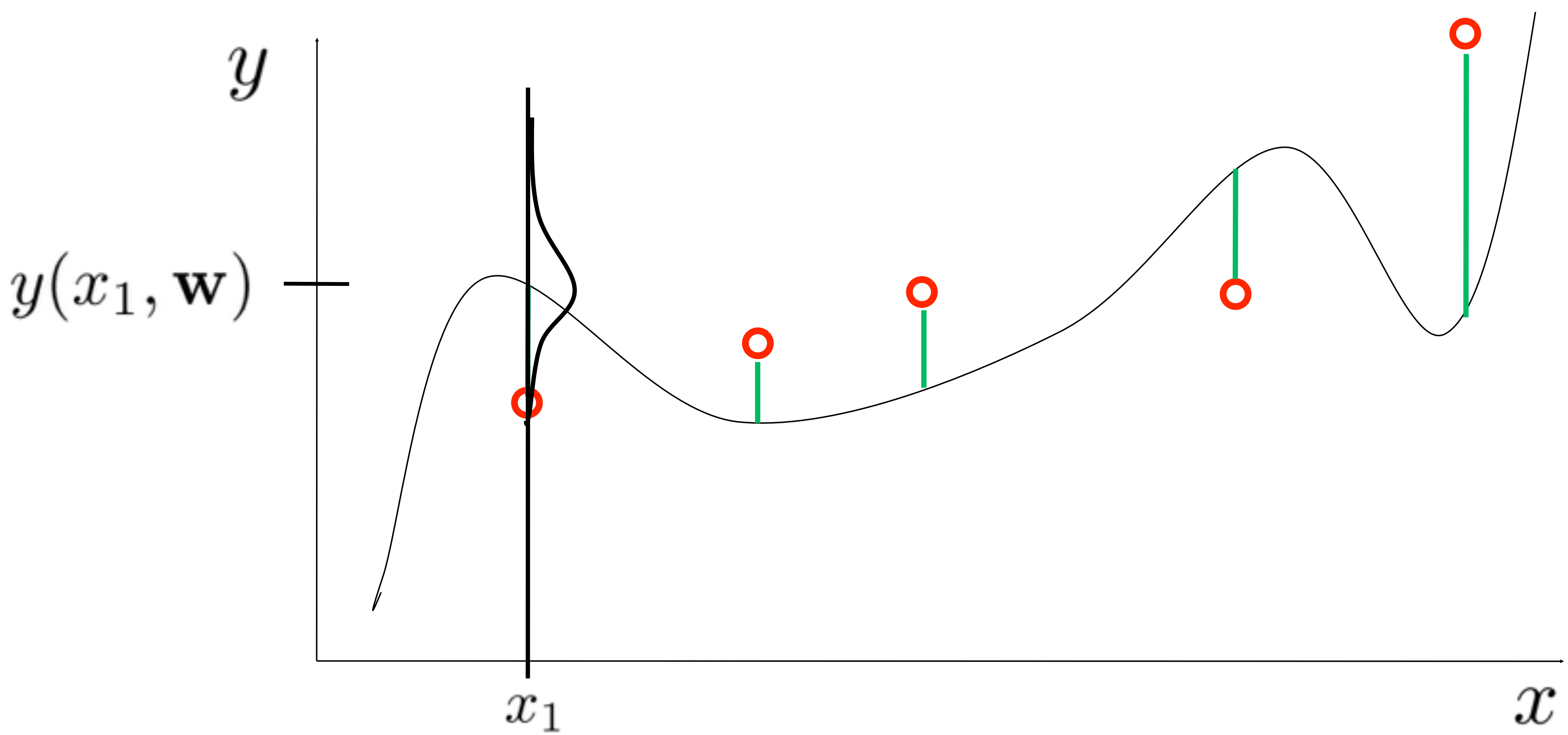
$$N = 10$$
$$M = 10$$
$$\lambda = 10^{-11}$$

The Problem from a Different View Point

Assume that y is affected by Gaussian noise :

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

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



Maximum 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:

$$p(\mathbf{t} \mid \mathbf{x}, \mathbf{w}, \sigma) = \prod_{i=1}^{N} p(t_i \mid x_i, \mathbf{w}, \sigma)$$

$$= \prod_{i=1}^{N} \mathcal{N}(t_i; \mathbf{w}^T \phi(x_i), \sigma^2)$$

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

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

$$\ln p(\mathbf{t} \mid \mathbf{x}, \mathbf{w}, \sigma) = \sum_{i=1}^{N} \ln p(t_i \mid x_i, \mathbf{w}, \sigma)$$

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

$$= \frac{-N(\ln(\sigma^2) + \ln(2\pi))}{2} - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(\mathbf{w}^T \boldsymbol{\phi}(x_i) - t_i)^2$$

Constant for all $\mathbf{w}$

Is equal to $E(\mathbf{w})$

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

Maximum Likelihood Estimation

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

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

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

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):

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

“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

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

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

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

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

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:

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

The same in vector notation:

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

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

“Feature Matrix”