Machine Learning

Linear Regression

Lesson 3

Linear Regression

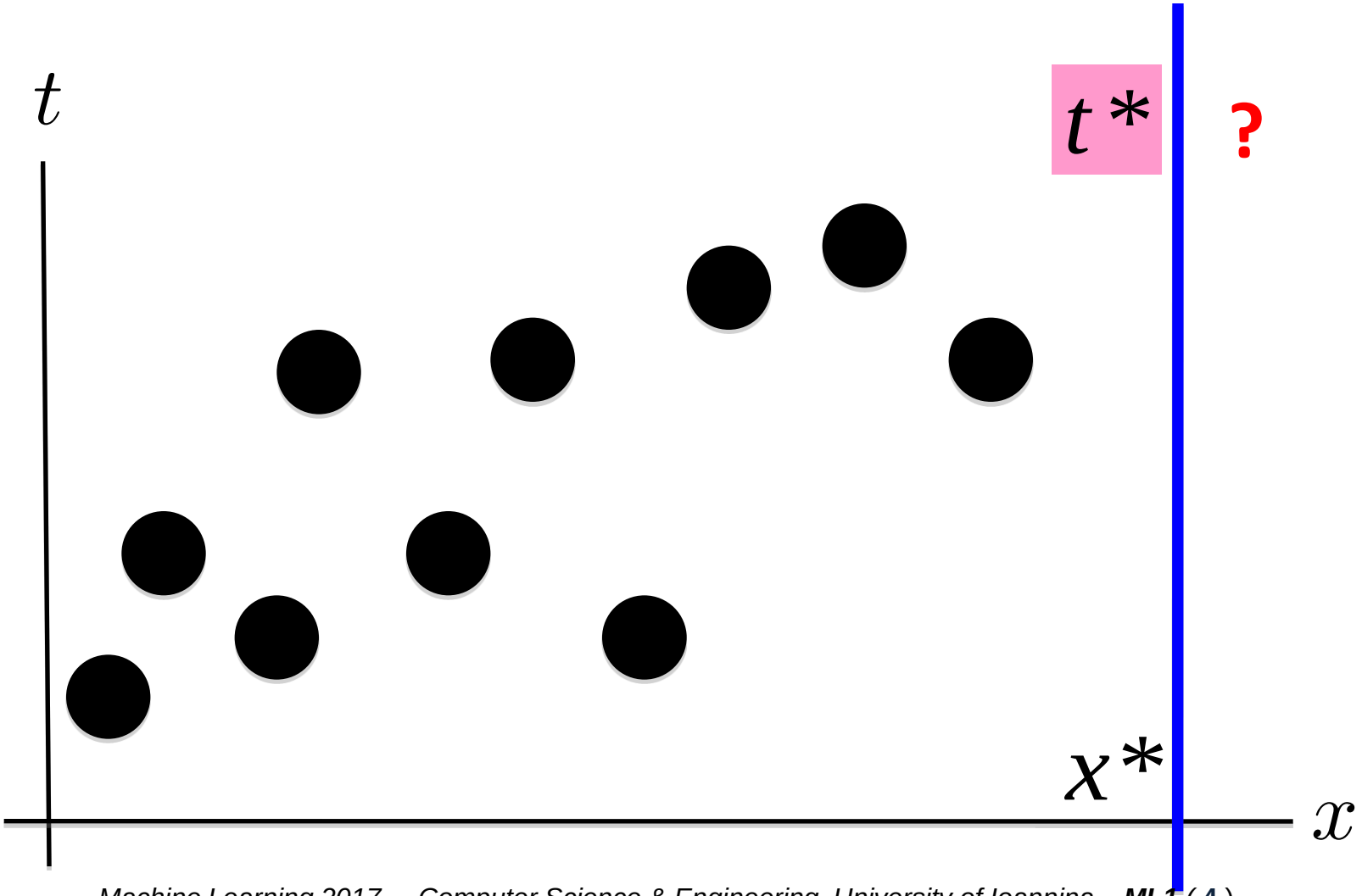
- **Basics of Regression**
- **Least Squares estimation**
- **Polynomial Regression**
- **Basis functions Regression model**
- **Regularized Regression**
- **Statistical Regression**
  - Maximum Likelihood (**ML**) estimation
  - Maximum A-Posteriori (**MAP**) estimation
  - **Bayesian** Regression

Basics in Regression

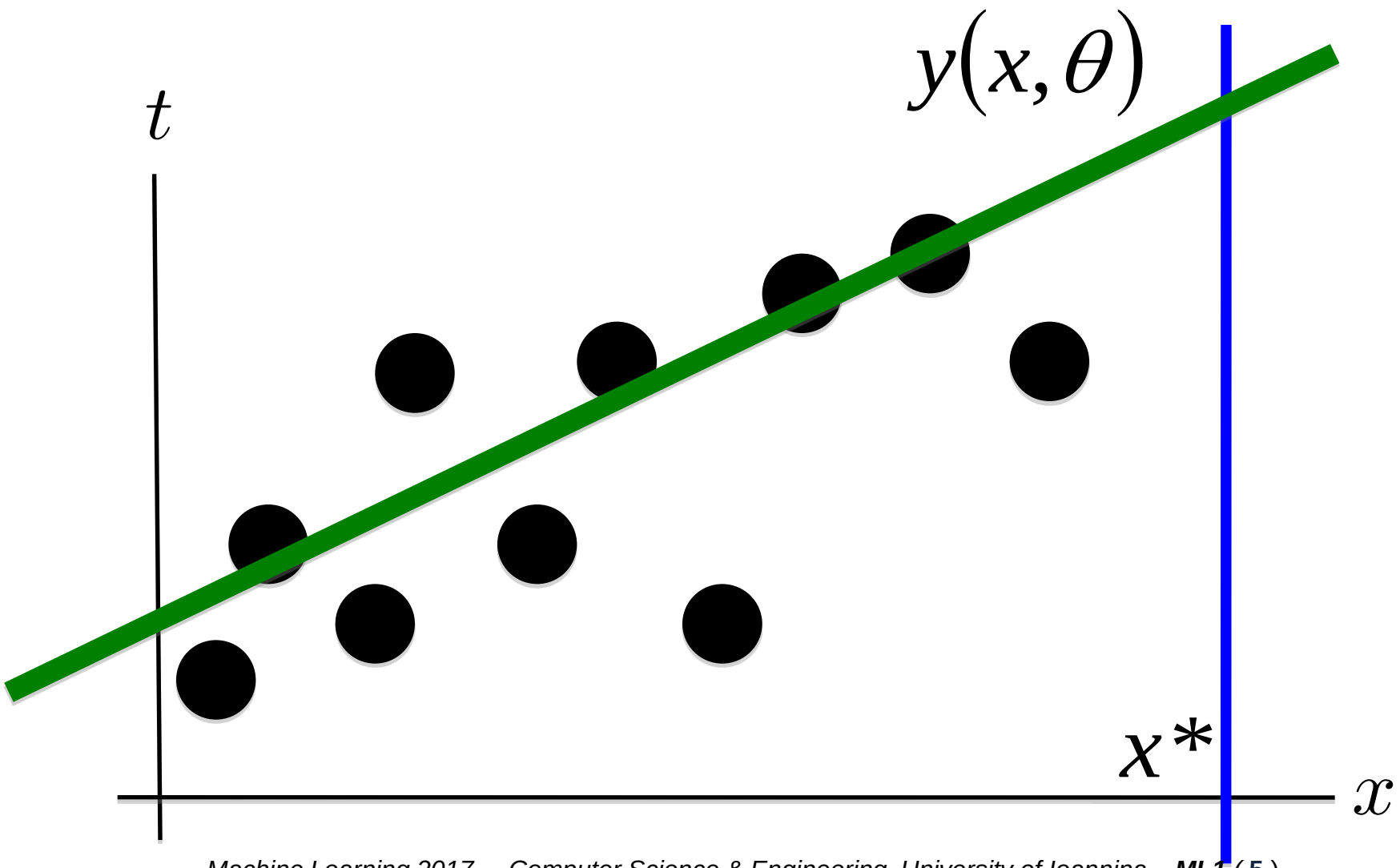
- Input: Set of N tuples: $D = \{(x_1, t_1), \ldots, (x_N, t_N)\}$
  - $x_i \in R^d$ is the sample
  - $t_i \in R$ ***prediction*** or a ***value of an unknown function*** over the features of x_i
- **Supervised** technique
- **Goal:** create a function ***y*** : $\forall x_i \in D \;:\; y(x_i, \theta) \approx t_i$
  - in one dimension $y : \mathbb{R} \rightarrow \mathbb{R}$
  - in d dimensions $y : \mathbb{R}^D \rightarrow \mathbb{R}$

θ is the (***unknown***) set of parameters

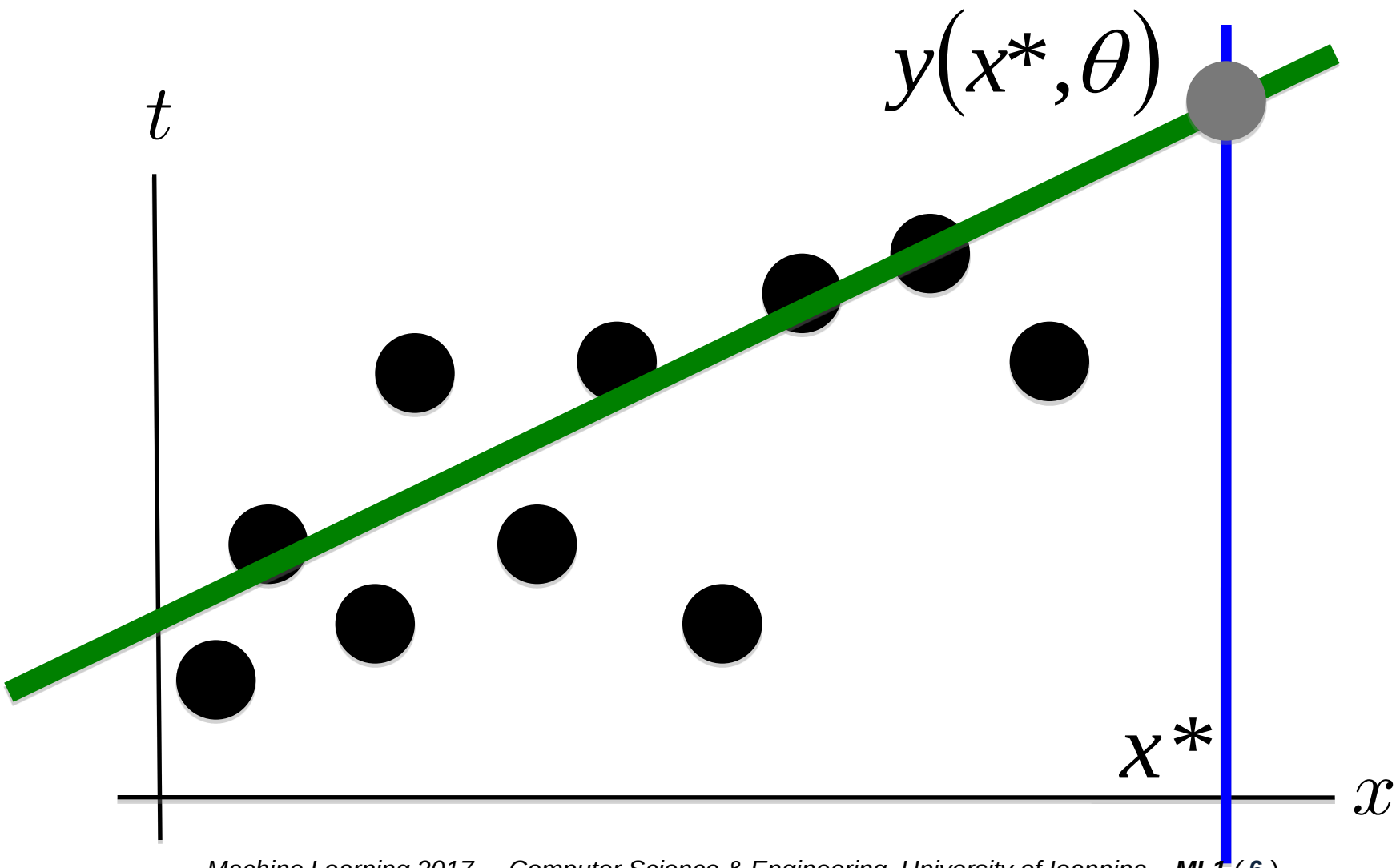
Graphical Example of Regression



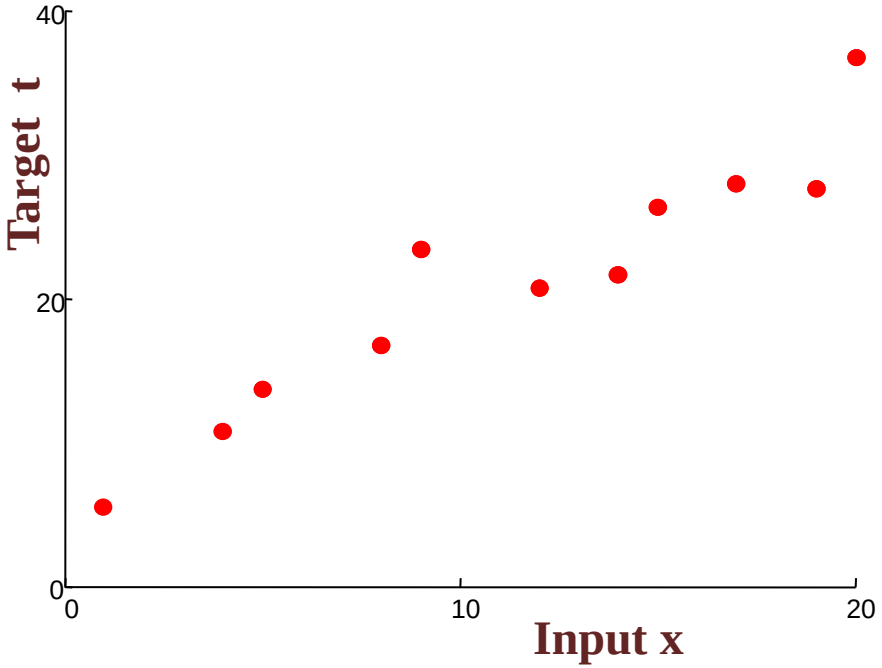
Graphical Example of Regression



Graphical Example of Regression



Linear regression:
The case of 1-dimensional data

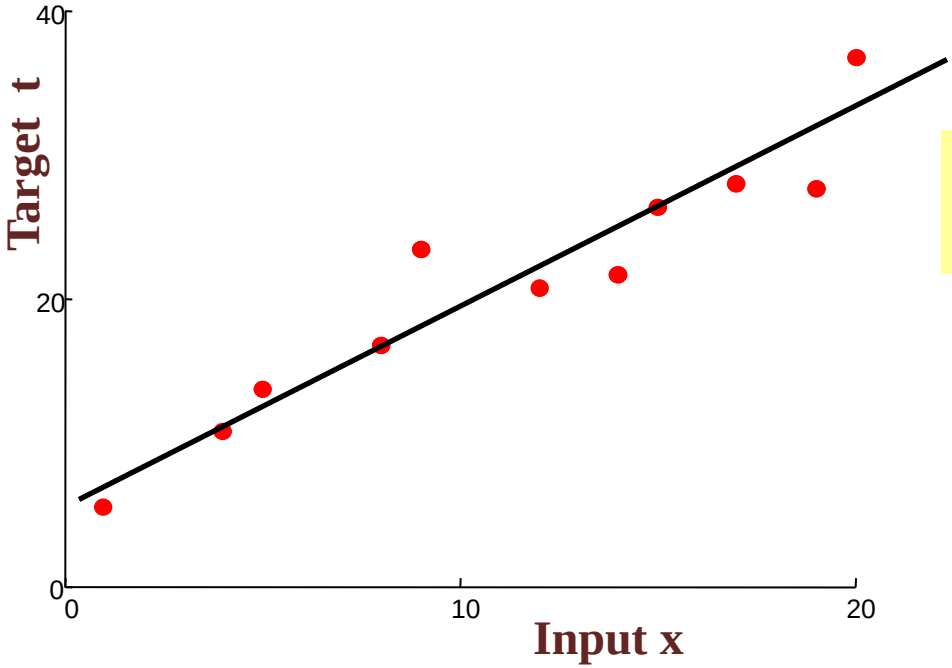


Input dataset

$$D = \{(x_1, t_1), \ldots, (x_N, t_N)\}$$

$x_i \in R$ (one feature)

Linear regression:
The case of 1-dimensional data



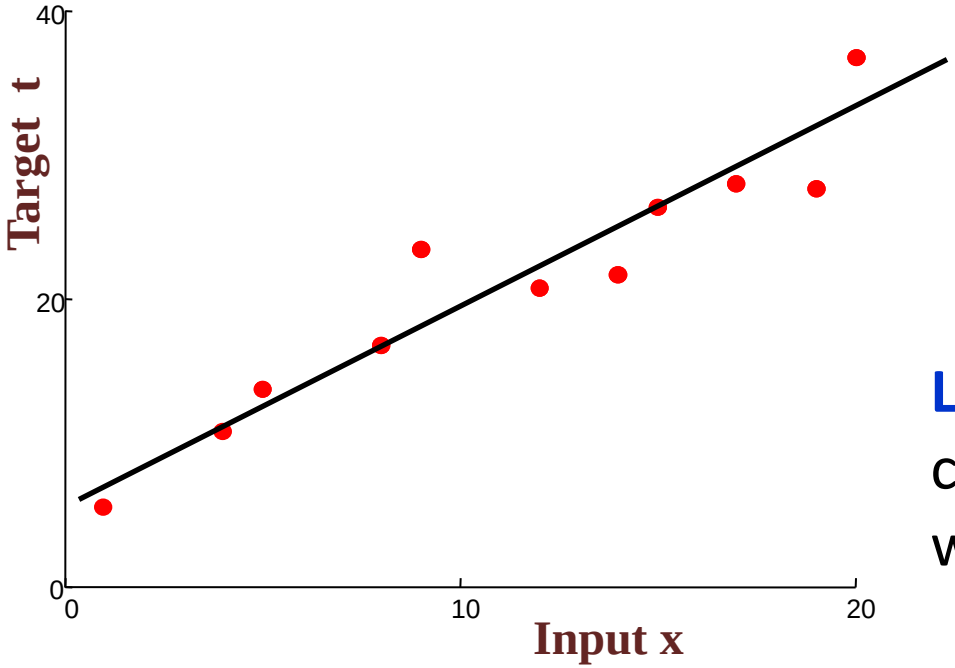
Input dataset

$$D = \{(x_1, t_1), \ldots, (x_N, t_N)\}$$

$x_i \in R$ (one feature)

Predictor: Evaluate line $y(x, w) = w_0 + w_1 x_1$

Linear regression:
The case of 1-dimensional data



Input dataset

$$D = \{(x_1, t_1), \ldots, (x_N, t_N)\}$$

Learning: Estimating the regression coefficients {w_0, w_1} which are the weights of the linear equation

$$y(x, w) = w_0 + w_1 x_1$$

so as $\forall x_i \in D \;:\; w_0 + w_1 x_i \approx t_i$

Least squares linear fit to data

- Most popular estimation method is **least squares**.
- Determine linear weights *w* that minimize the **sum of squared loss (SSL):**

$$J(w)=\frac{1}{2}\sum_{i=1}^{N}\left(y(x_i,w)-t_i\right)^2=\sum_{i=1}^{N}\left(w_0+w_1x_i-t_i\right)^2$$

- Use standard differential calculus:
  - differentiate SSL with respect to w_0 , w_1
  - find zeros of each partial differential equation
  - solve for w_0 , w_1

- Derivatives of parameters:

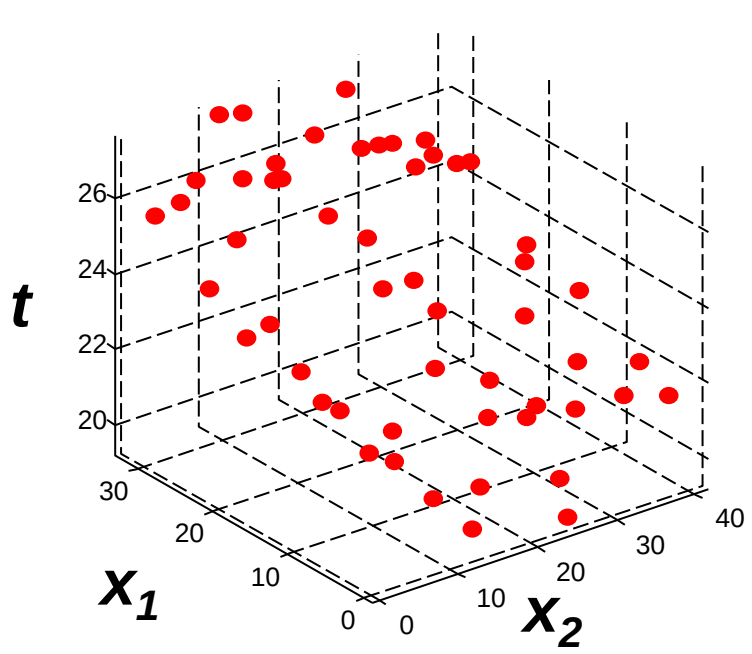
$$\frac{\partial J(w)}{\partial w_0} = \sum_{i=1}^{N} w_0 + w_1 x_i - t_i = 0 \Rightarrow N w_0 + w_1 \sum_{i=1}^{N} x_i = \sum_{i=1}^{N} t_i$$

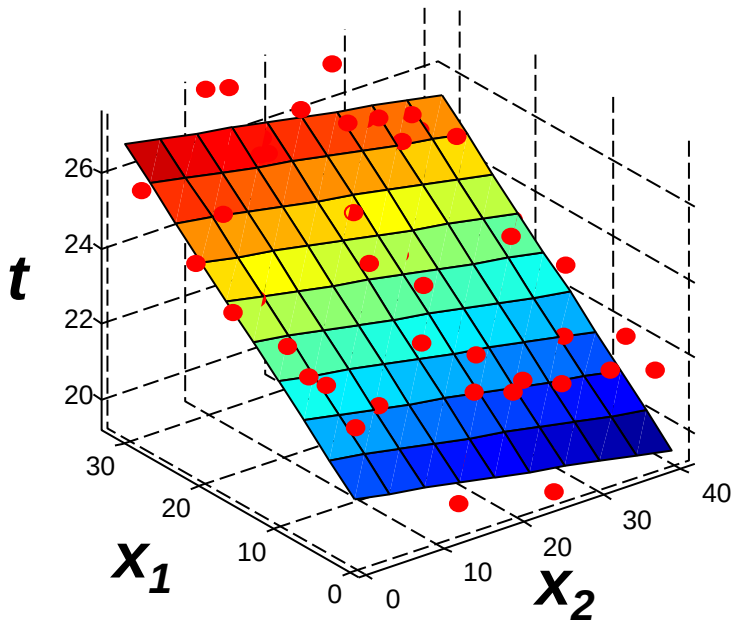
$$\frac{\partial J(w)}{\partial w_1} = \sum_{i=1}^{N} (w_0 + w_1 x_i - t_i) x_i = 0 \Rightarrow w_0 \sum_{i=1}^{N} x_i + w_1 \sum_{i=1}^{N} x_i^2 = \sum_{i=1}^{N} x_i t_i$$

$$\hat{w}_1 = \frac{\frac{1}{N}\sum_{i=1}^{N} x_i t_i - (\bar{x})(\bar{t})}{\overline{x^2} - (\bar{x})^2} = \frac{\mathrm{cov}(x,t)}{\mathrm{var}(x)}$$

$$\hat{w}_0 = \bar{t} - \hat{w}_1 \bar{x}$$

More dimensions (d>1) $x \in R^d$





- Input x has ***d* features**: $\vec{x} = [x_1 \ldots x_d]$

- There are ***d+1* linear weights** for describing the regression function $y(x, w)$

- Linear regression model

$$y(x, w) = w_0 + w_1 x_1 + \ldots + w_d x_d = w_0 + \sum_{j=1}^{d} w_j x_j$$

- Alternative representation

$$y(x, w) = w^T \tilde{x}$$

where $w = [w_0 \; w_1 \ldots w_d]$

$$\tilde{x} = [1 \, x_1 \ldots x_d]$$

Sum of Squared Error (SSE)

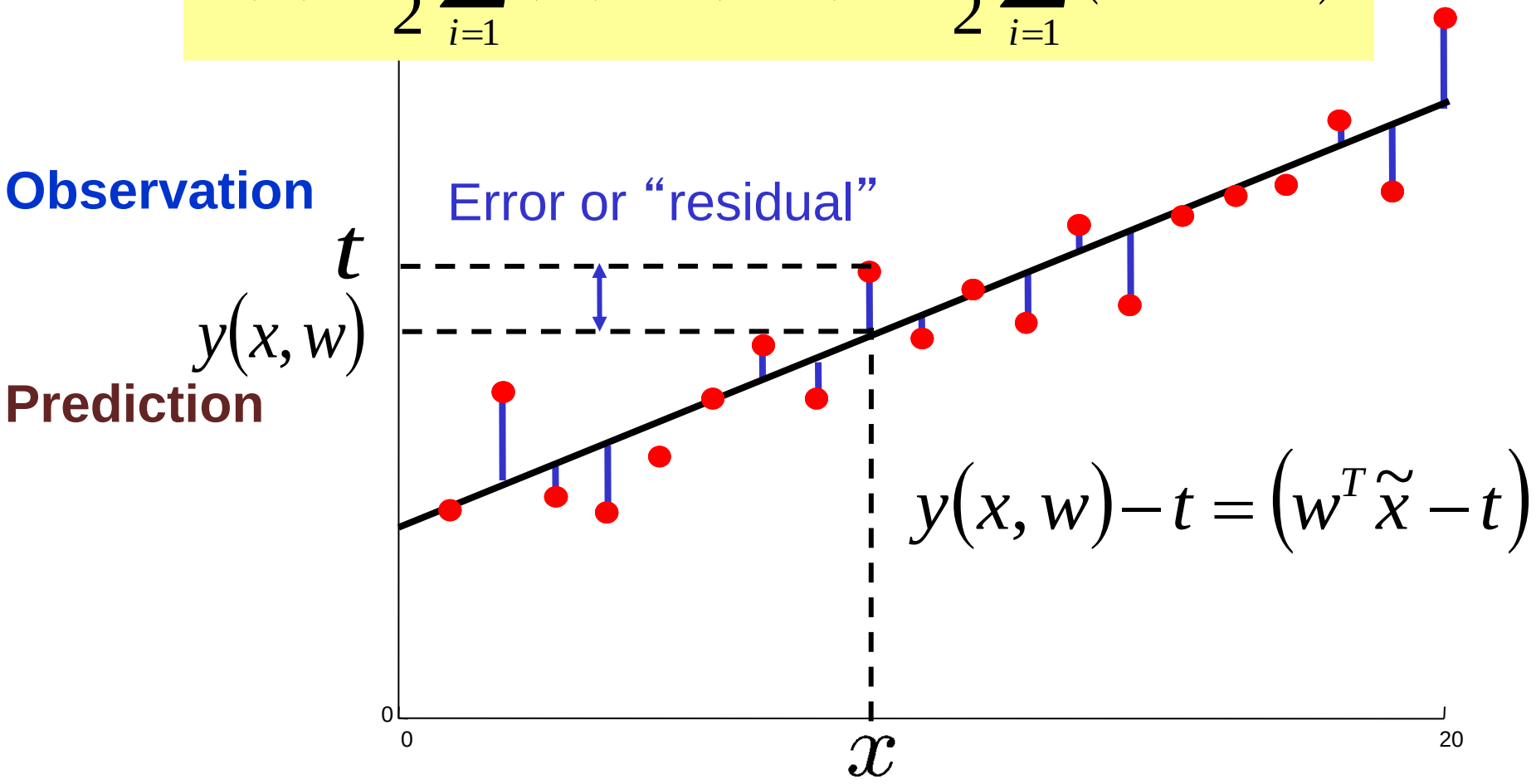
- How can we quantify the error?

$$J(w) = \frac{1}{2}\sum_{i=1}^{N}\left(y(x_i, w) - t_i\right)^2$$

$$J(w) = \frac{1}{2}\sum_{i=1}^{N}\left(w_0 + w^T x_i - t_i\right)^2 = \frac{1}{2}\sum_{i=1}^{N}\left(w^T \tilde{x}_i - t_i\right)^2$$

Sum of Squared Error (SSE)

$$J(w)=\frac{1}{2}\sum_{i=1}^{N}\left(y(x_i,w)-t_i\right)^2=\frac{1}{2}\sum_{i=1}^{N}\left(w^T\tilde{x}_i-t_i\right)^2$$



- How can we quantify the error?

$$J(w)=\frac{1}{2}\sum_{i=1}^{N}\left(w^T\hat{x}_i-t_i\right)^2=\frac{1}{2}\left\|Xw-T\right\|^2$$

$$J(w)=\frac{1}{2}\left(Xw-T\right)^T\left(Xw-T\right)$$

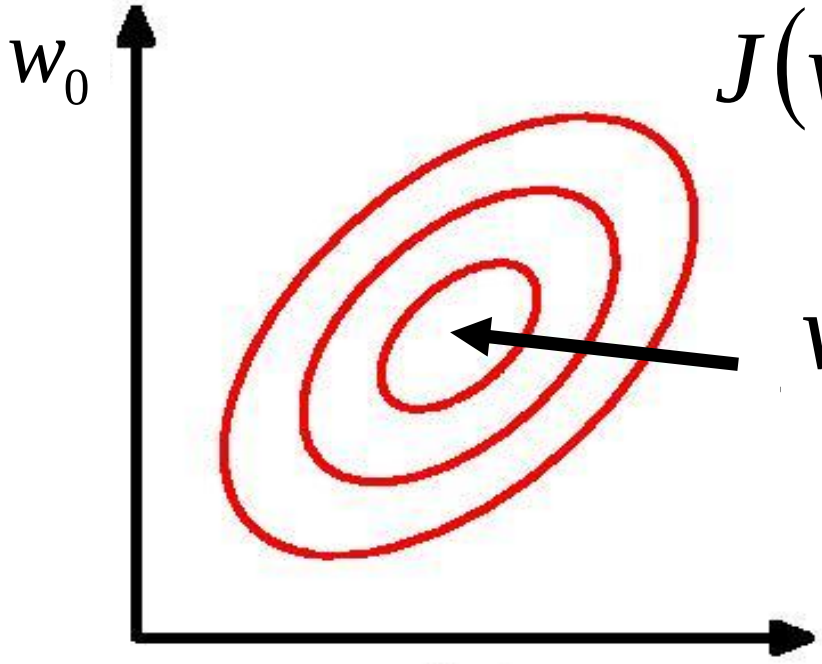
$$X=\begin{bmatrix}1 & x_{11} & \cdots & x_{1d}\\ \vdots & \vdots & \ddots & \vdots\\ 1 & x_{N1} & \cdots & x_{Nd}\end{bmatrix}$$

$$T=\begin{bmatrix}t_1 & \cdots & t_N\end{bmatrix}^T$$

$$w=\begin{bmatrix}w_0 & w_1 \ldots & w_d\end{bmatrix}^T$$

Finding good parameters

- Want to find parameters **w** which minimize the error
- Think of a cost "surface": error residual for that **w**…



$$J(w)$$

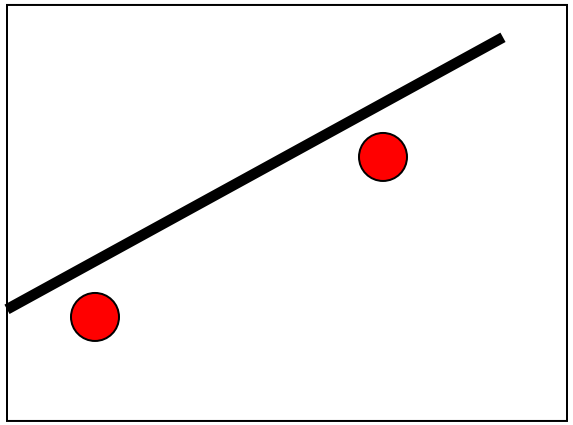
$$\hat{w} = \arg\min_{w} J(w)$$

SSE Minimization

- Consider a **simple problem**
  - One feature (**d=1**), two data points (**N>2**)
  - Two unknowns: w_0, w_1
  - Two equations:

$$t_1 = w_0 + w_1 x_1$$

$$t_2 = w_0 + w_1 x_2$$

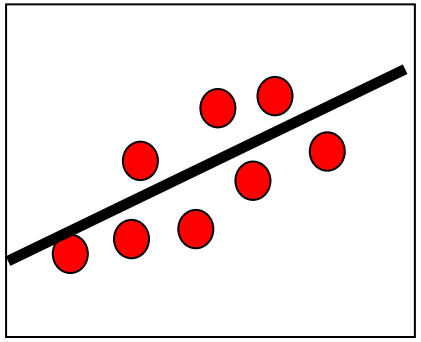


- Can solve this system directly ($\boldsymbol{X}$: 2x2):

$$T = Xw \quad \Rightarrow \quad \hat{w} = X^{-1}T$$

- However, most of the time, N > d+1
  - There may be no linear function that hits all the data exactly
  - Instead, solve directly for minimum of SSE function

The general case



$$J(w)=\frac{1}{2}\|Xw-T\|^2=\frac{1}{2}(Xw-T)^T(Xw-T)$$

$$\nabla_w J(w)=0\Rightarrow X^T(Xw-T)=0$$

- Reordering, we take

$$X^T X\hat{w}=X^T T\Rightarrow \hat{w}_{LS}=(X^T X)^{-1}X^T T$$

Least Squares estimator

- $(X^T X)^{-1}X^T$ is called the "**pseudo-inverse**"
- If X^T is square and independent, this is the inverse X^{-1}

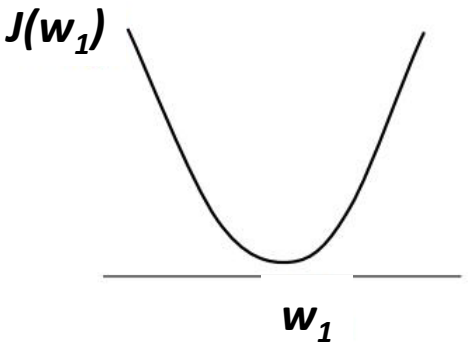
Optimization methods of SSE

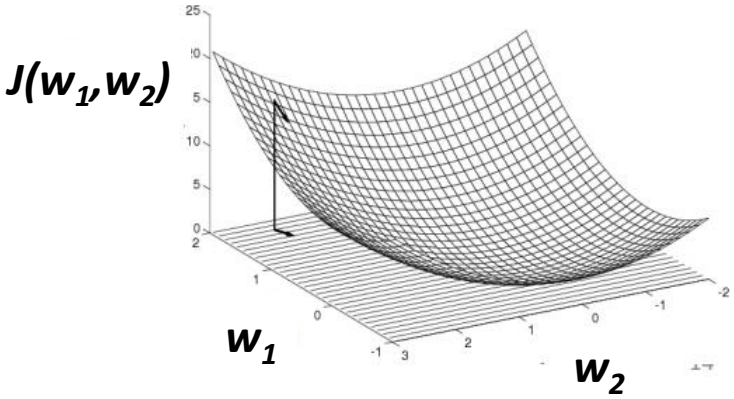
- Even when ($X^T X$) is invertible, might be **computationally expensive** if X is huge.
- Treat as an **optimization problem**:

$$\hat{w} = \arg\min_{w} J(w) = \arg\min_{w} \frac{1}{2} \|Xw - T\|^2$$

- **How to find an estimator?**

J(w) is convex in w

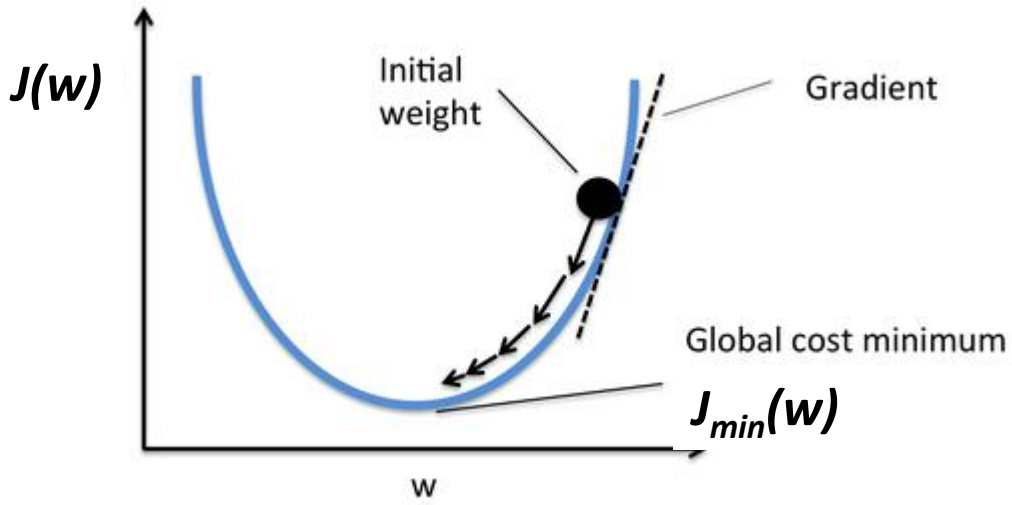




Gradient Descent Optimization

- **Steepest Descent**

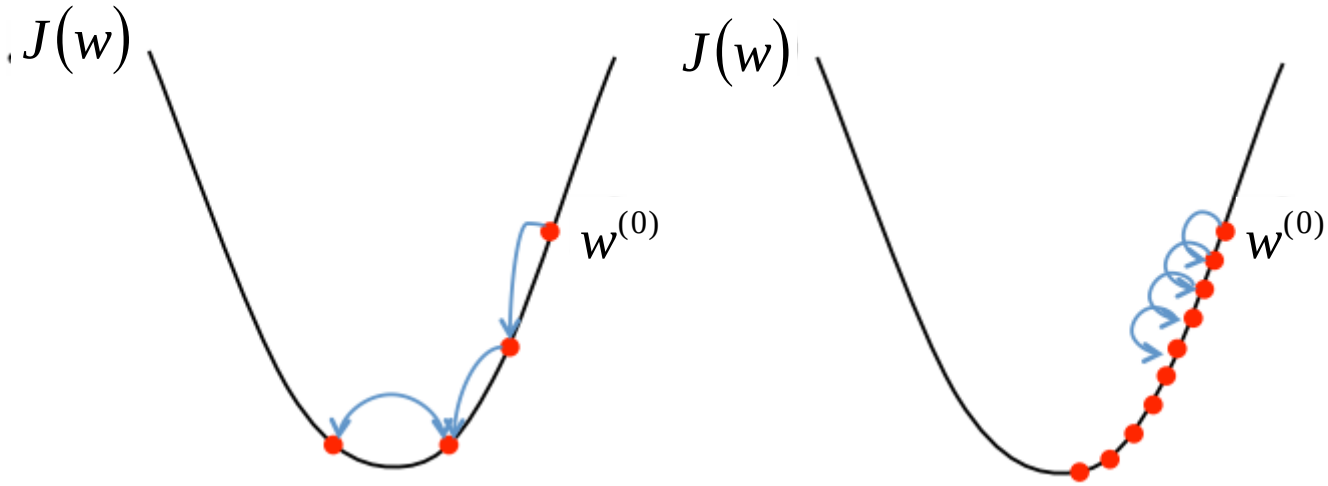
Function decreases most quickly in the direction of the negative gradient.



η: **learning rate (or step-size)**

$$w^{(new)} = w^{(old)} - \eta \frac{\partial J(w)}{\partial w}$$

Effect of step-size (η)



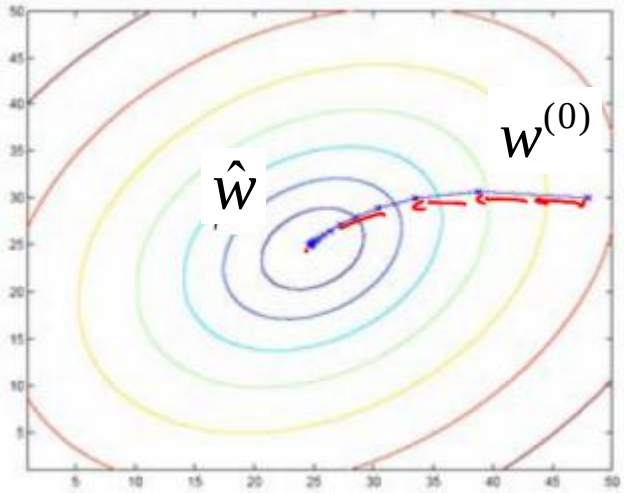
- **Large η :** fast convergence but larger residual error. May cause oscillations

- **Small η :** slow convergence but small residual error.

1. Gradient descent Optimization scheme

- Since ***J(w)*** is convex, move along negative of gradient

- **Initialize:** (*t=0*) $w^{(0)}$



- **Update** rule:

$$w^{(new)} = w^{(old)} - \eta \left. \frac{\partial J(w)}{\partial w} \right|_t$$

$$w^{(new)} = w^{(old)} - \eta \, X^T \left(X w^{(old)} - T \right)$$

- **Stop:** when some criterion is met (e.g. fixed # iterations) or when

$$\left. \frac{\partial J(w)}{\partial w} \right|_{w^{(old)}} < \varepsilon$$

2. Stochastic Gradient descent

- Previous scheme was a **batch** gradient descent: all training examples are participated at every step

$$w^{(new)} = w^{(old)} - \eta\, X^T \left(X w^{(old)} - T \right) = w^{(old)} - \eta \sum_{i=1}^{N} \left(w^{(old)^T} x_i - t_i \right) x_i$$

- **Stochastic gradient descent** scheme repeatedly examines **a single** example at every step:

$$for\ \ i = 1, \ldots, N$$

$$w^{(new)} = w^{(old)} - \eta \left(w^{(old)^T} x_i - t_i \right) x_i$$

- **Advantage**: often gets w close to the minimum much faster than batch mode.
- It is preferred over batch when training set is large.

3. *Newton-Raphson* Optimization scheme

- **Update rule:** $$w^{(new)} = w^{(old)} - H^{-1}\nabla J(w)$$

where ***H***: Hessian matrix (second derivatives of J(w))

$$\nabla J(w) = X^T X w - X^T T$$

$$H = \nabla\nabla J(w) = X^T X$$

- By substituting the previous we obtain the rule:

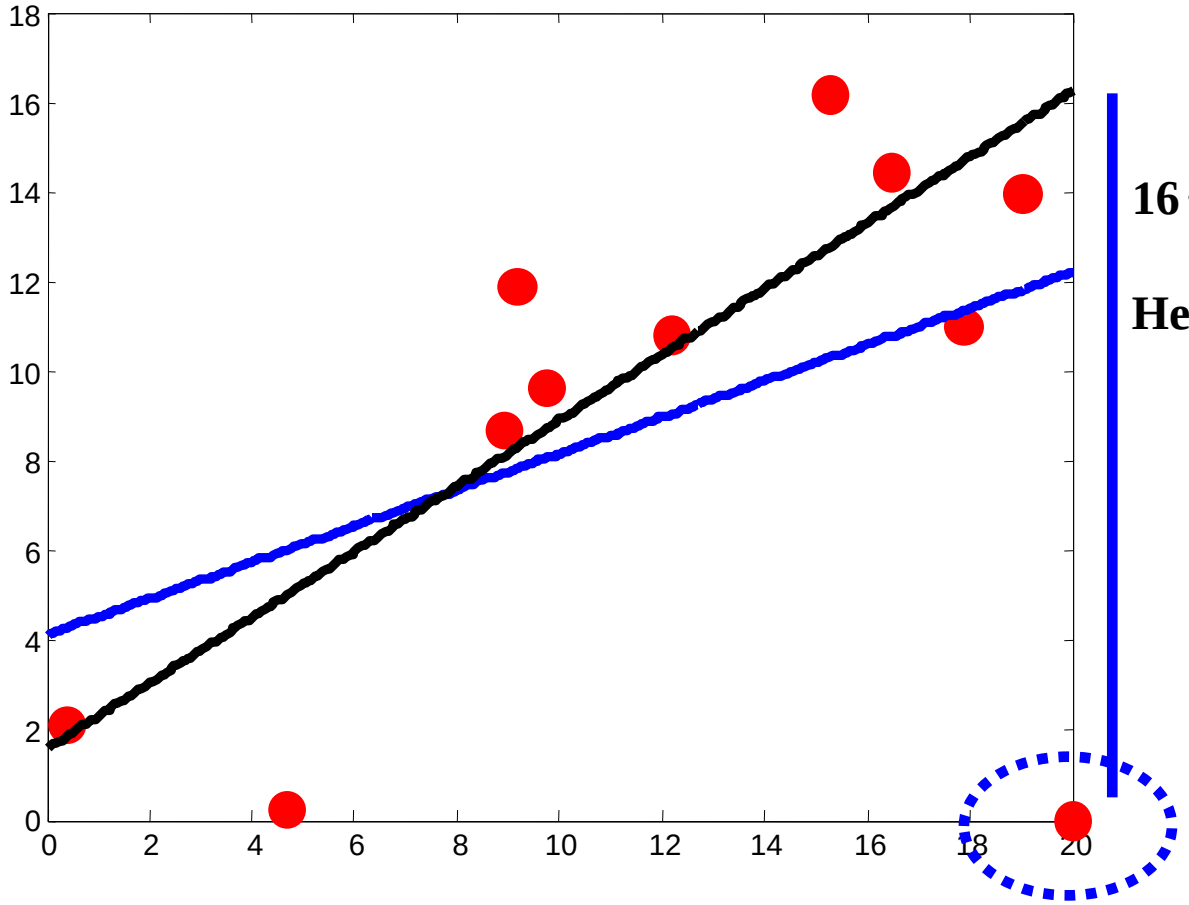
$$w^{(new)} = w^{(old)} - \left(X^T X\right)^{-1}\left(X^T X w^{(old)} - X^T T\right) =$$

$$= \left(X^T X\right)^{-1} X^T T$$

- This is **the least-squares solution** and is exact solution in one step

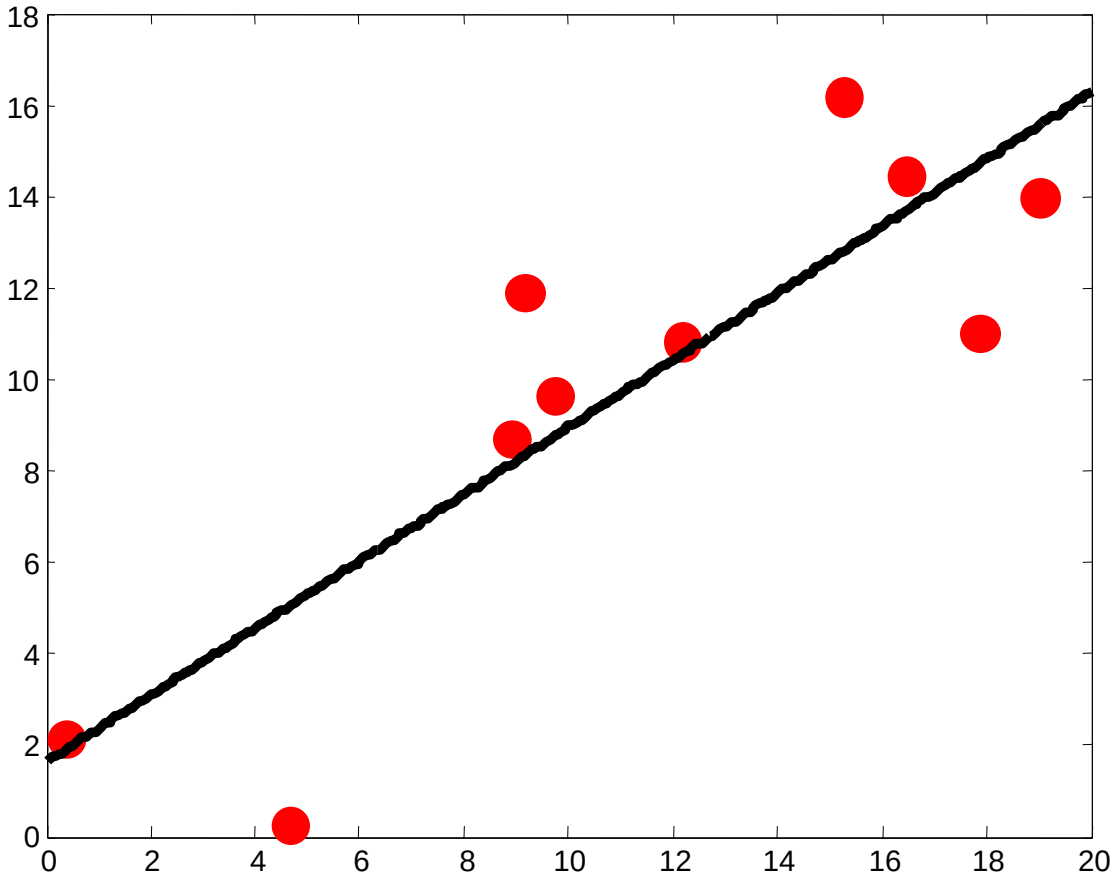
Effects of **SSE** (***l2* error**) choice

- Sensitivity to outliers



Use the sum of absolute error (SAE) (*l1* error)

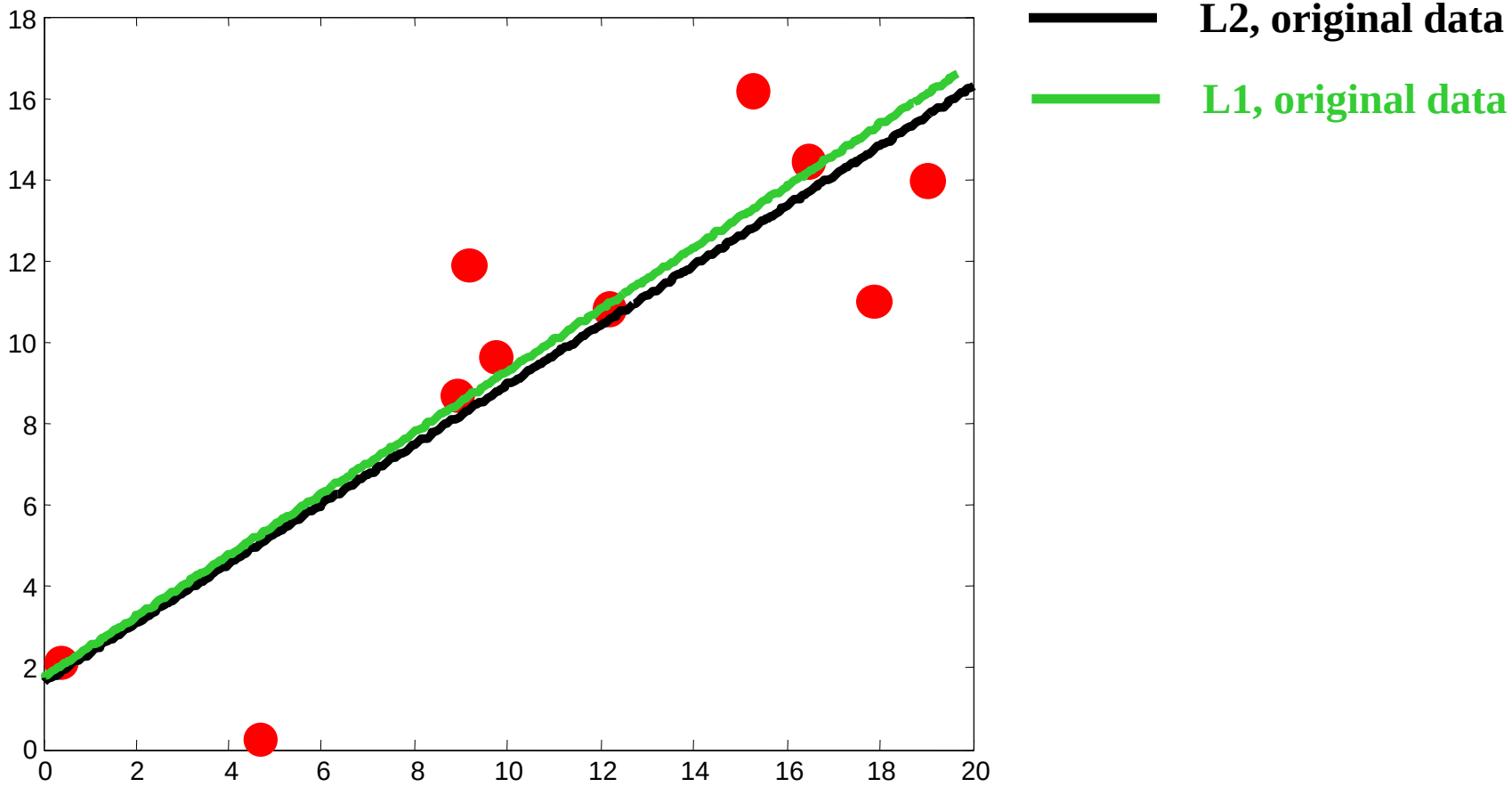
$$J(w)=\sum_{i=1}^{N}\left|w^t \breve{x}_i - t_i\right|$$



L2, original data

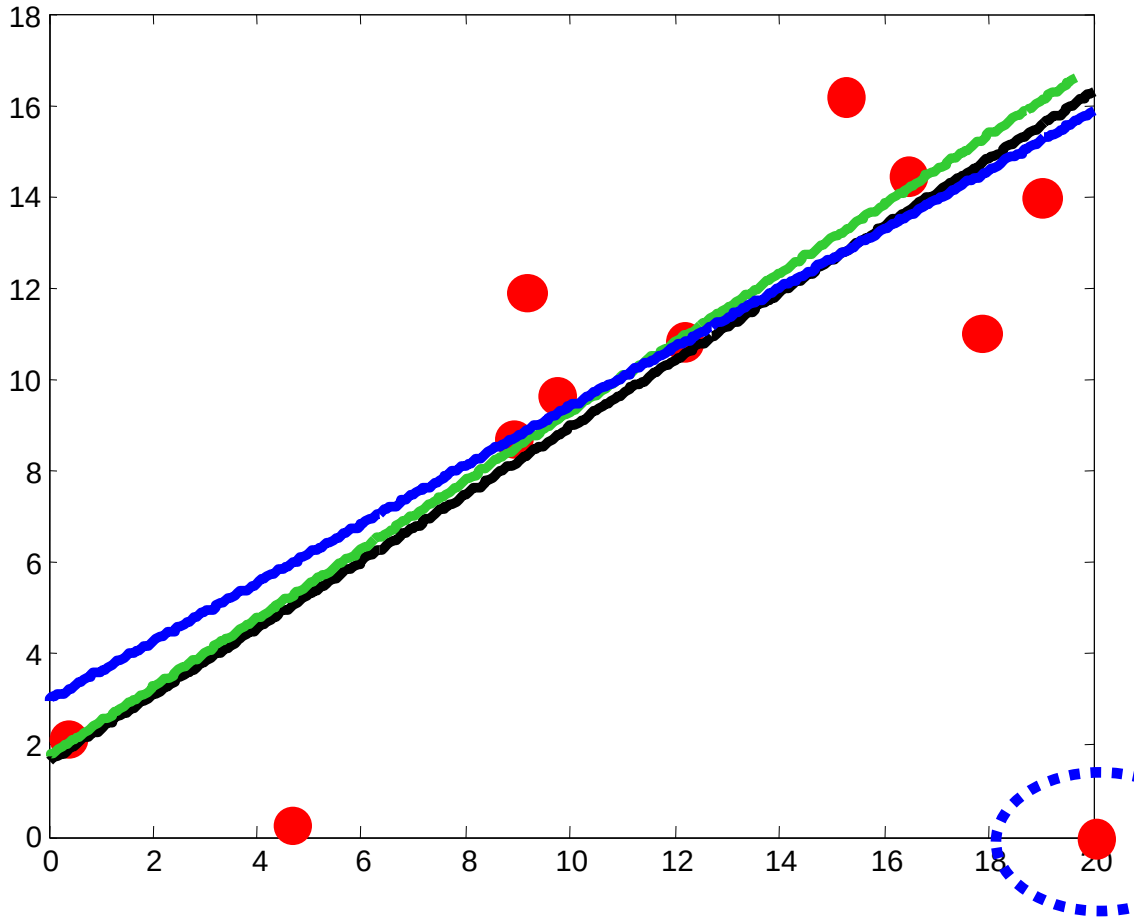
Use the sum of absolute error (SAE) (*l1* error)

$$J(w)=\sum_{i=1}^{N}\left|w^t \breve{x}_i - t_i\right|$$



Use the sum of absolute error (SAE) (*l1* error)

$$J(w)=\sum_{i=1}^{N}\left|w^t\breve{x}_i-t_i\right|$$



L2, original data

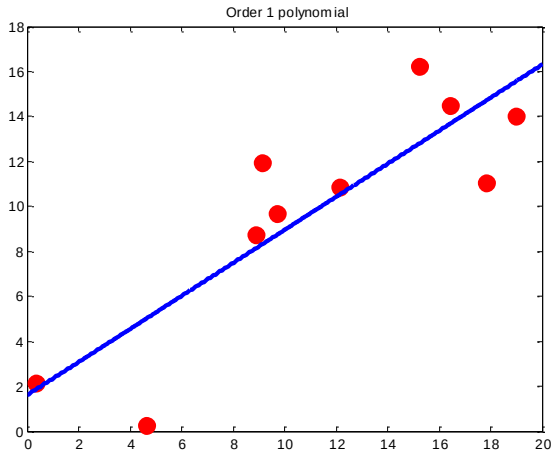
L1, original data

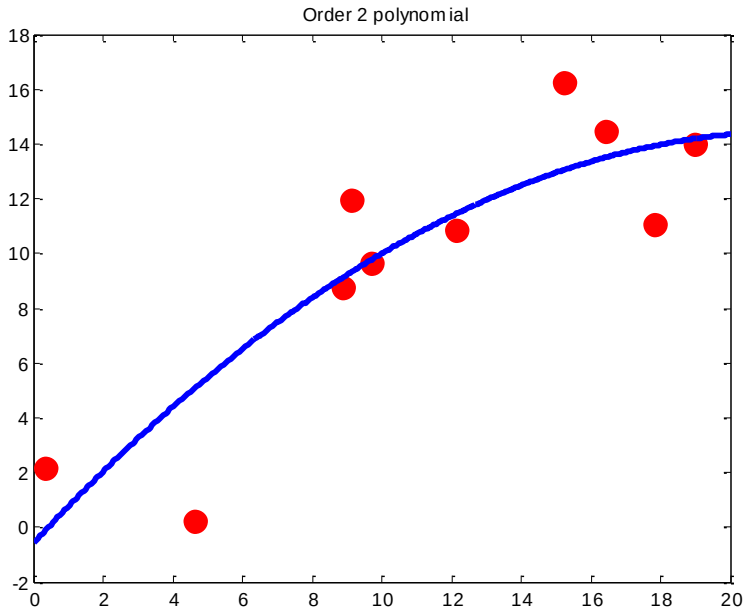
L1, outlier data

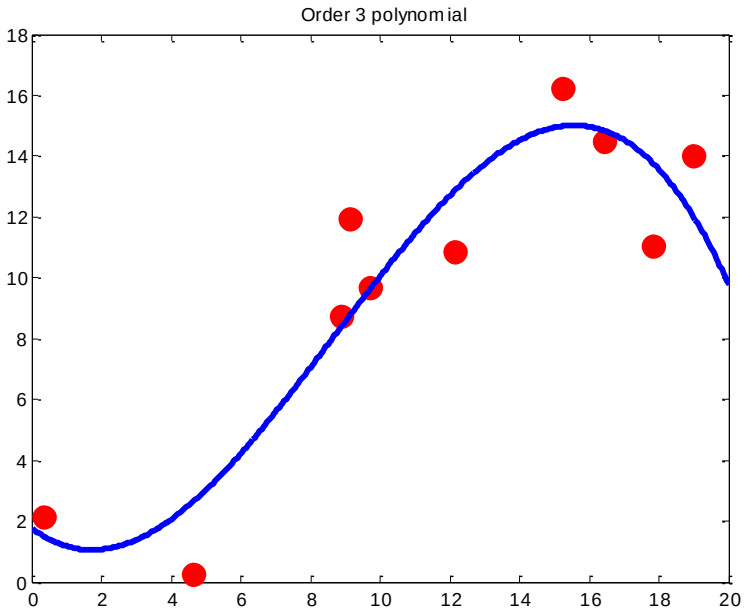
L1 is more **robust** to outliers. However, the solution is **unstable** and the optimization problem is **harder** (use pattern search schemes, e.g. SIMPLEX).

Non-linear Regression

- Consider **non-linear regression**
  - Ex: **higher-order polynomials**







Polynomial Regression

$$y(x,w) = w_0 + w_1 x + \ldots + w_\rho x^\rho = w_0 + \sum_{k=1}^{\rho} w_k x^k = w^T \phi(x)$$

$$J(w) = \frac{1}{2}\sum_{i=1}^{N}\left(w^T\phi(x_i) - t_i\right)^2 = \frac{1}{2}\left\|\Phi w - T\right\|^2$$

$$\Phi = \begin{bmatrix} 1 & x_1 & \cdots & x_1^\rho \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & \cdots & x_N^\rho \end{bmatrix}$$

$$T = \begin{bmatrix} t_1 & \cdots & t_N \end{bmatrix}^T$$

$$w = \begin{bmatrix} w_0 & w_1 \ldots w_d \end{bmatrix}^T$$

- **Example:**

  Use 3d polynomial. Single feature ***x***, predict target ***t***:

$$D = \{(x_i, t_i)\}$$

$$y(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

⇓ **Add features:**

$$D = \{(\langle x_i, (x_i)^2, (x_i)^3 \rangle, t_i)\}$$

⇓

$$y(x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3$$

Linear regression in **new features**

- Sometimes useful to think of "**feature transform**"

$$y(x) = w^T \phi(x)$$

$$\phi(x) = [1, x, x^2, x^3]$$

Learning polynomial regression coefficients

- **Using Least-squares**:

$$\hat{w}_{LS} = \left(\Phi^T \Phi\right)^{-1} \Phi^T \mathrm{T}$$

- **Using Gradient descent update rule:**

$$w^{(new)} = w^{(old)} - \eta\, \Phi^T \left(\Phi w^{(old)} - T\right)$$

Linear Basis Function Models

- In general, can use **any feature set** we think is useful

$$\phi(x) = [\phi_1(x), \ldots, \phi_m(x)]$$

- Other **information about the problem**
  - e.g. location, age, …
- **Polynomial functions**
  - Features [1, x, x^2, x^3, …]
- **Other functions**
  - 1/x, sqrt(x), $x_1 * x_2$, …

$$\phi_j(x): R^d \rightarrow R$$

basis functions

$$y(x) = \sum_{j=1}^{m} w_j \phi_j(x) = w^T \phi(x)$$

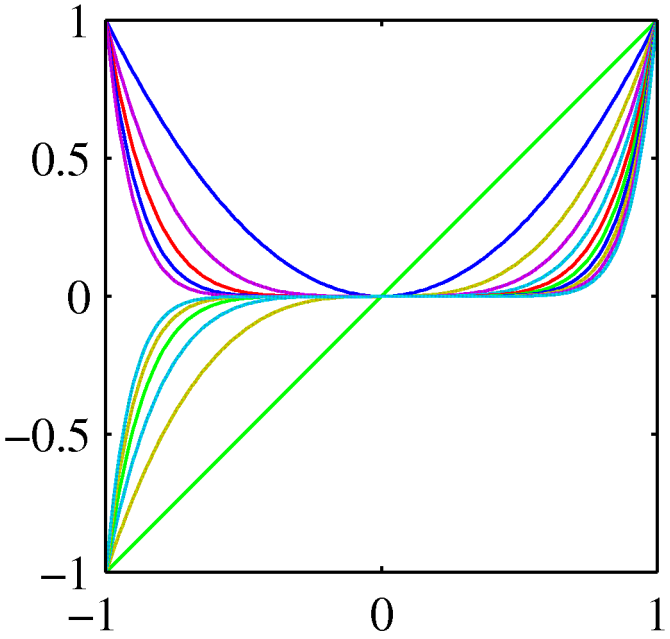
- **Regression remains "Linear" = linear in the parameters**
  - Features we can make as complex as we want!

Examples of Basis Function

Polynomial basis functions

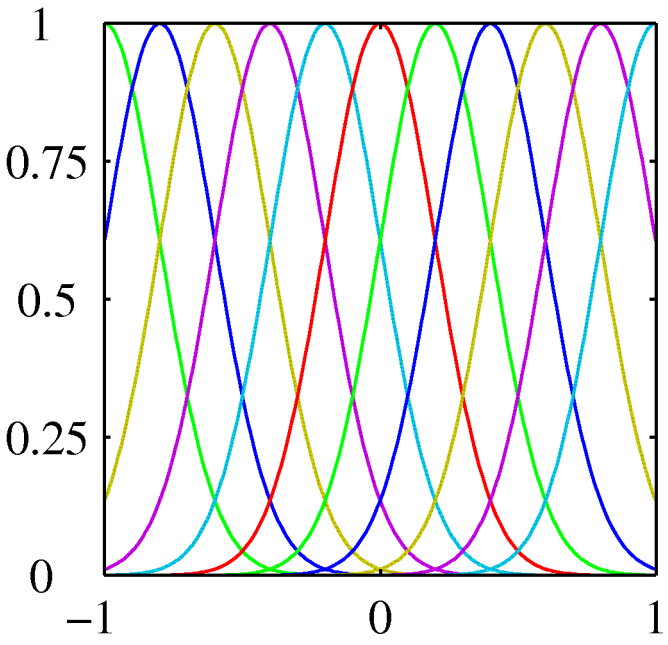
$$\phi_j(x) = x^j.$$

- These are **global**
- A small change in *x* affect all basis functions.



Gaussian basis functions

$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$$



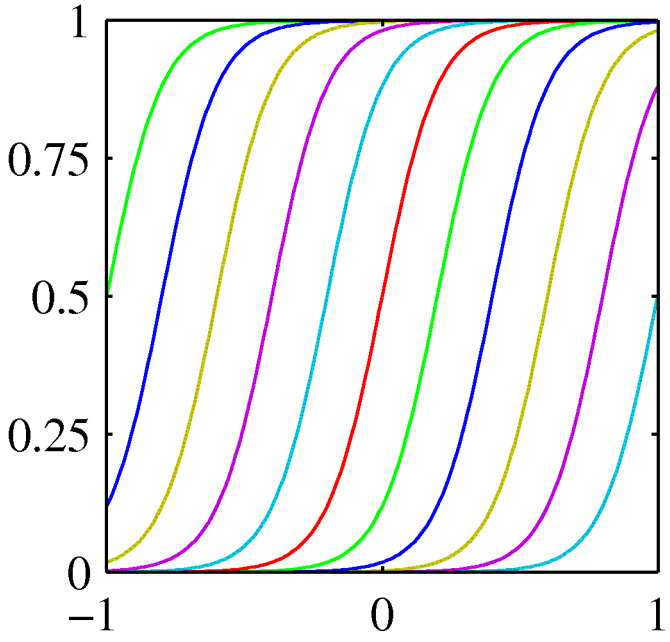
- These are **local**
- A small change in x only affect nearby basis functions.
- Parameters μ_j and s control location and scale (width).
- Related to **kernel methods**.

Sigmoid basis functions

$$\phi_j(x) = \sigma\left(\frac{x - \mu_j}{s}\right)$$

where

$$\sigma(a) = \frac{1}{1 + \exp(-a)}.$$



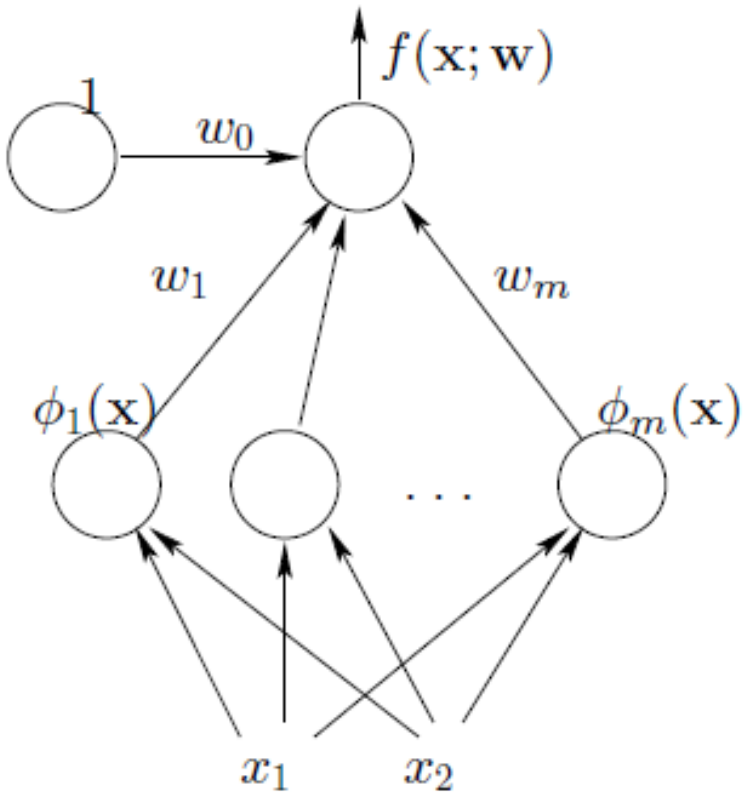
- Also these are **local**: a small change in x only affect nearby basis functions.
- Parameters $\boldsymbol{\mu_j}$ and **s** control location and scale (slope).

Additive models

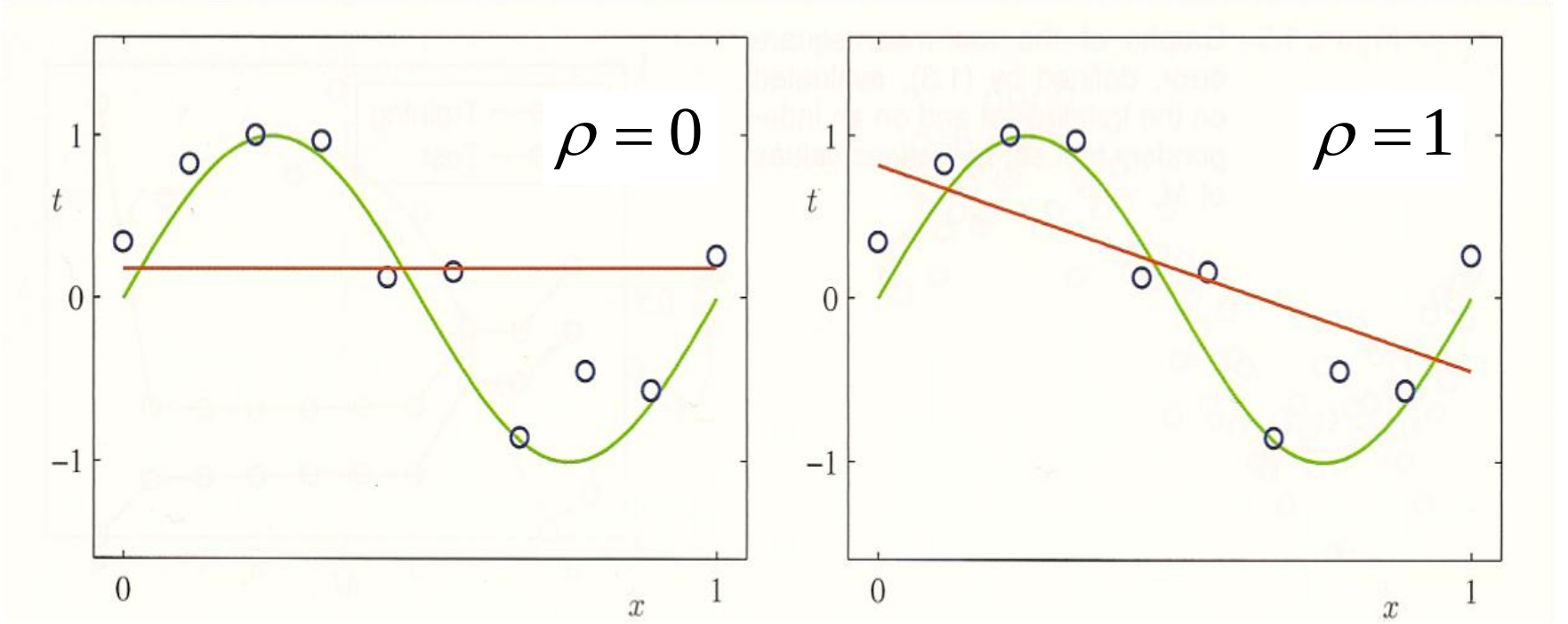
- The basis functions can capture various **properties** of the inputs (e.g. **qualitative**)
- These are called "**Additive models**"
- For example: we can try to rate documents based on text descriptors

$$\begin{aligned} \mathbf{x} &= \text{text document (collection of words)} \\ \phi_i(\mathbf{x}) &= \begin{cases} 1 \text{ if word } i \text{ appears in the document} \\ 0 \text{ otherwise} \end{cases} \\ f(\mathbf{x};\mathbf{w}) &= w_0 + \sum_{i \in \text{words}} w_i \phi_i(\mathbf{x}) \end{aligned}$$

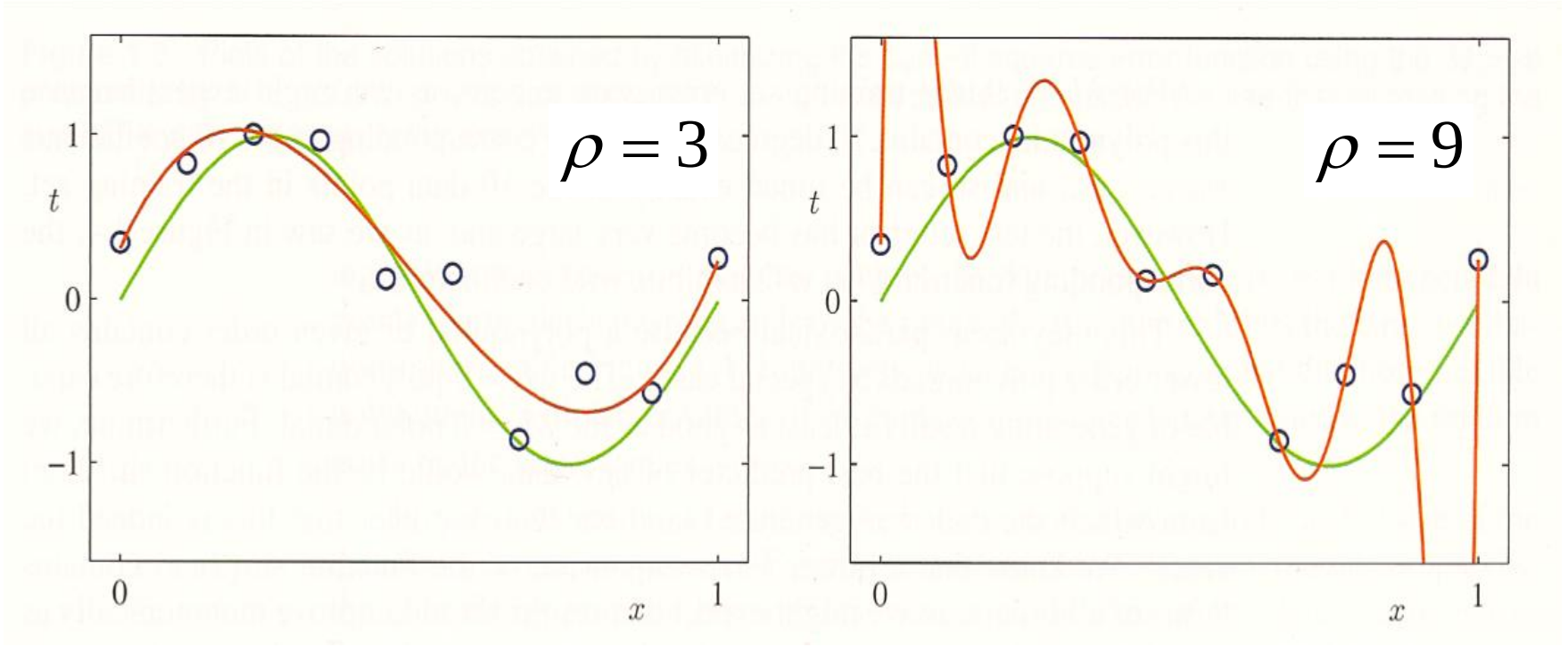
- We can view the additive models **graphically** in terms of simple "**units**" and "**weights**".



Overfitting

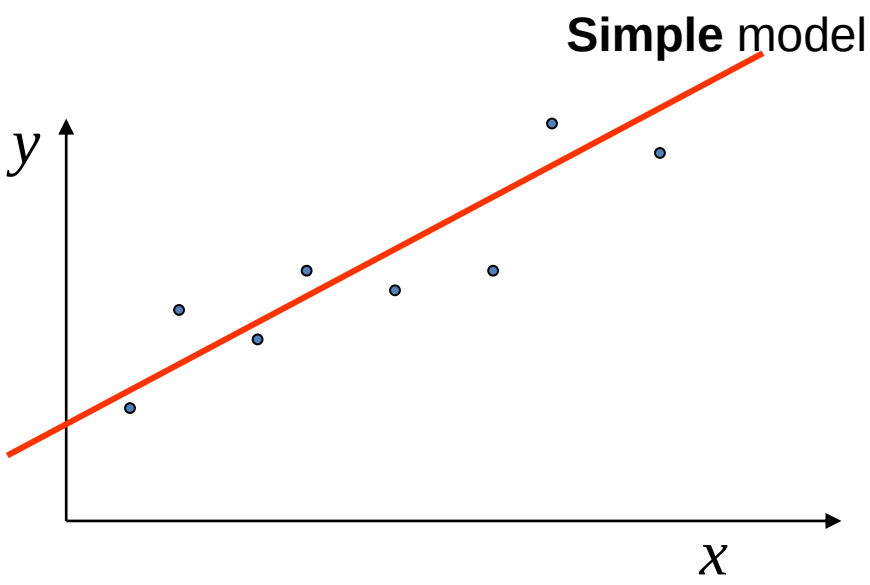


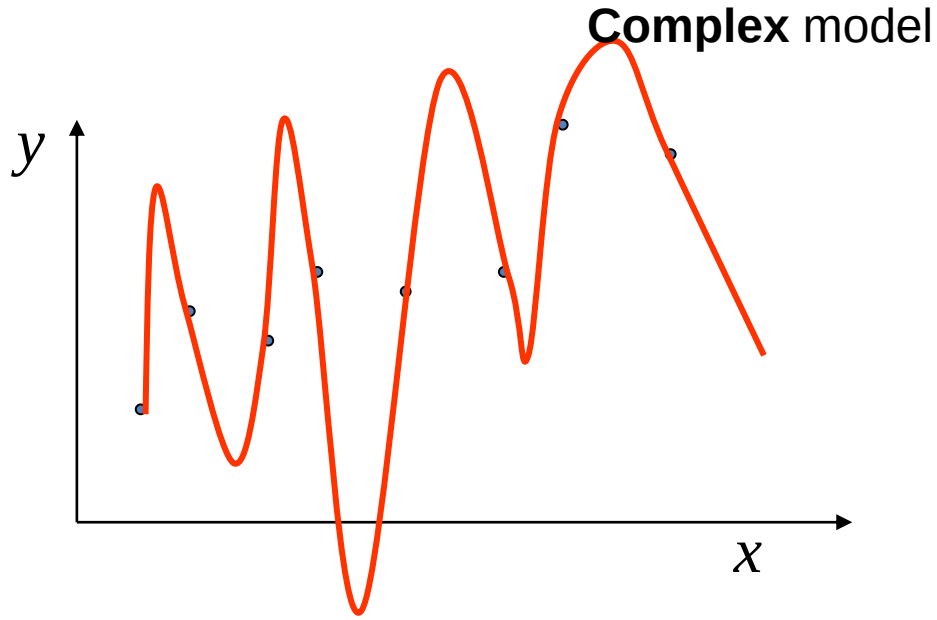
Overfitting



Overfitting and complexity

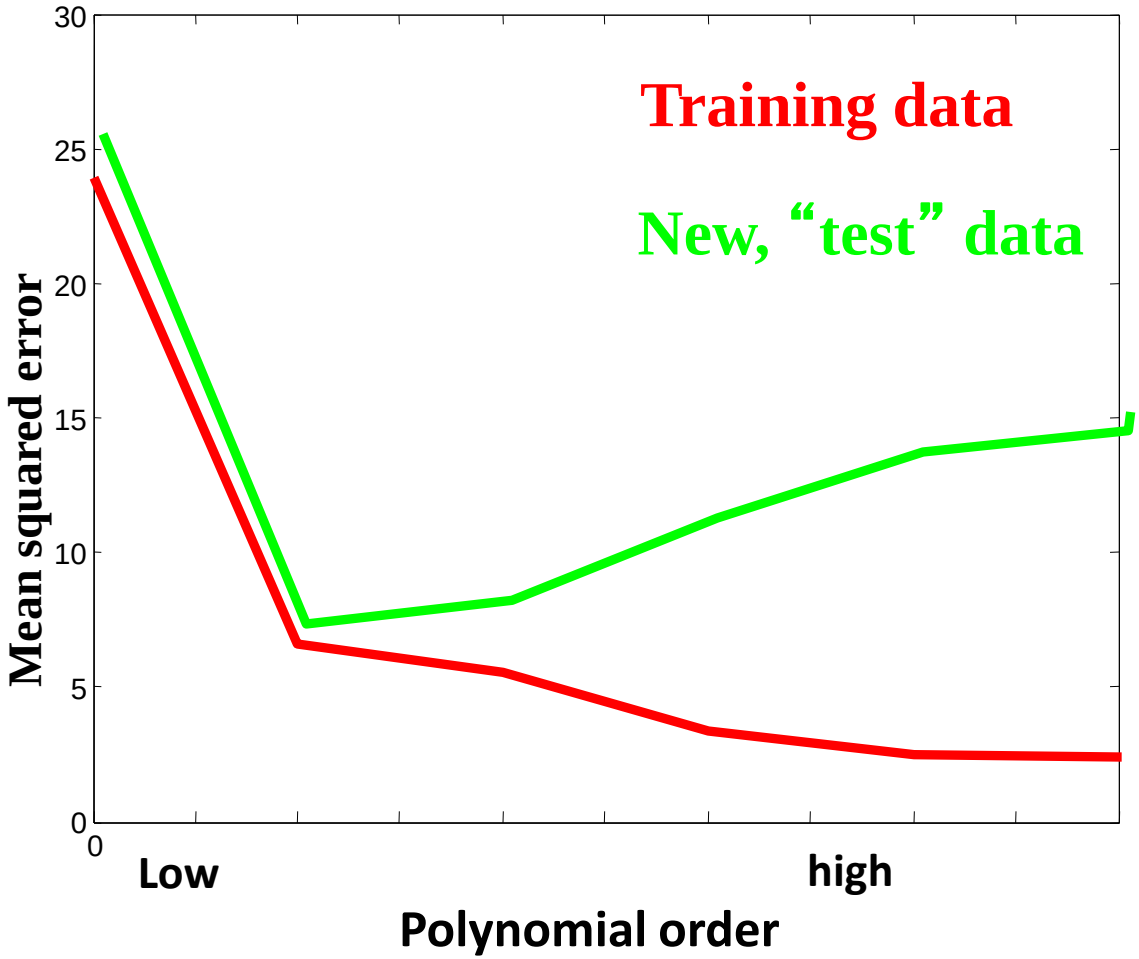
- More **complex models** will always fit the training data better
- **But** they may "**overfit**" the training data, learning complex relationships that are **not really present**





Training versus test error

- Plot MSE as a function of model complexity
  - Polynomial order
- Decreases
  - More complex function fits training data better
- What about new data?
- Low order
  - Error decreases
  - Underfitting
- Higher order
  - Error increases
  - Overfitting

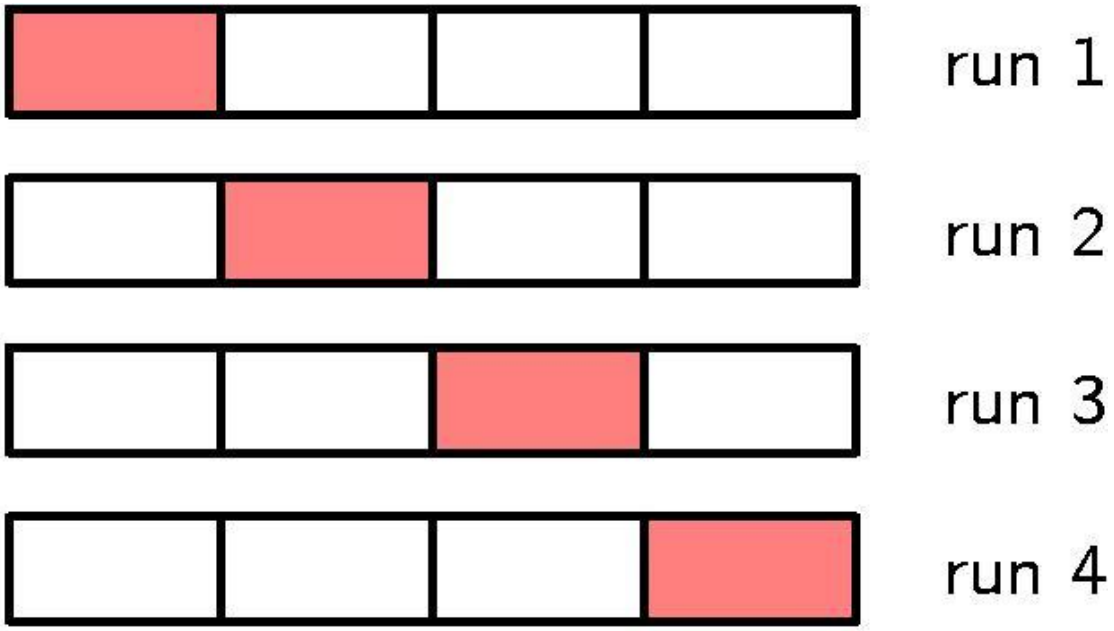


Dealing with Overfitting

- ✓Use more data
- ✓**Use a tuning set**
- ✓**Regularization**
- ✓**Be a Bayesian**

1. Avoiding overfitting: Cross-validation

- *Cross-validation* allows us to estimate the generalization error based on training examples alone

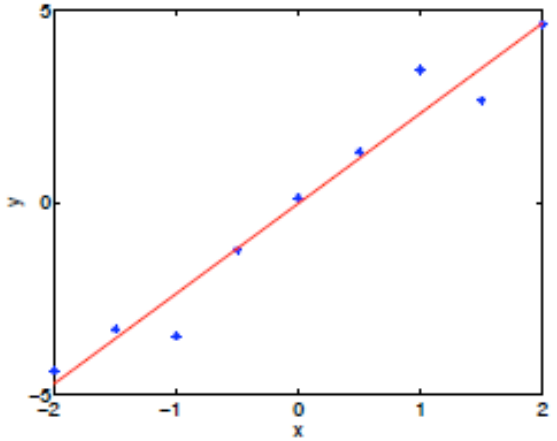


- Leave-one-out cross-validation treats each training example in turn as a test example:

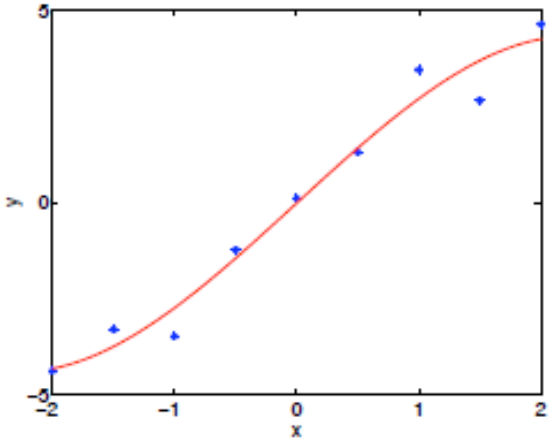
$$CV = \frac{1}{N}\sum_{i=1}^{N}\left(y\left(x_i, \hat{w}^{-i}\right) - t_i\right)^2$$

where $\hat{w}^{-i}$ are the least squares estimates of the parameters without the i^{th} training example.

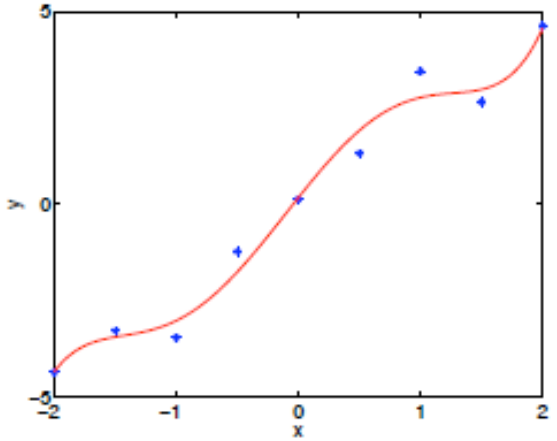
Polynomial Regression example



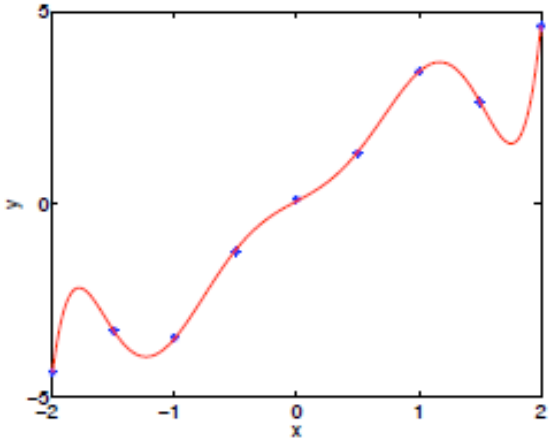
degree = 1, CV = 0.6



degree = 3, CV = 1.5



degree = 5, CV = 6.0



degree = 7, CV = 15.6

2. Regularization

- Overfitting is reduced by imposing a constraint on the overall magnitude of the parameters.

- Objective function is modified

$$J(w) = E_D(w) + \lambda E_W(w)$$

Data term + Regularization term

λ : regularization parameter

- $\mathbf{E_w(w)}$ set constraints to linear weights

L2-Regularization or Ridge Regression

- The regularization term is quadratic

$$E_W(w) = \frac{1}{2} w^T w = \frac{1}{2} \sum_{j=1}^{M} w_j^2$$

that penalize large weights

- Objective function

$$J(w) = \frac{1}{2} \sum_{i=1}^{N} \left(w^T \phi(x_i) - t_i \right)^2 + \frac{\lambda}{2} w^T w =$$

$$= \frac{1}{2} \left\| \Phi w - T \right\|^2 + \frac{\lambda}{2} w^T w$$

Regularization Derivation

$$\nabla_w J(w) = 0 \Rightarrow \Phi^{\mathrm{T}}\Phi\hat{w} - \Phi^{T}T + \lambda\hat{w} = 0 \Rightarrow$$

$$\hat{w} = \left(\Phi^{\mathrm{T}}\Phi + \lambda I\right)\Phi^{T}T$$

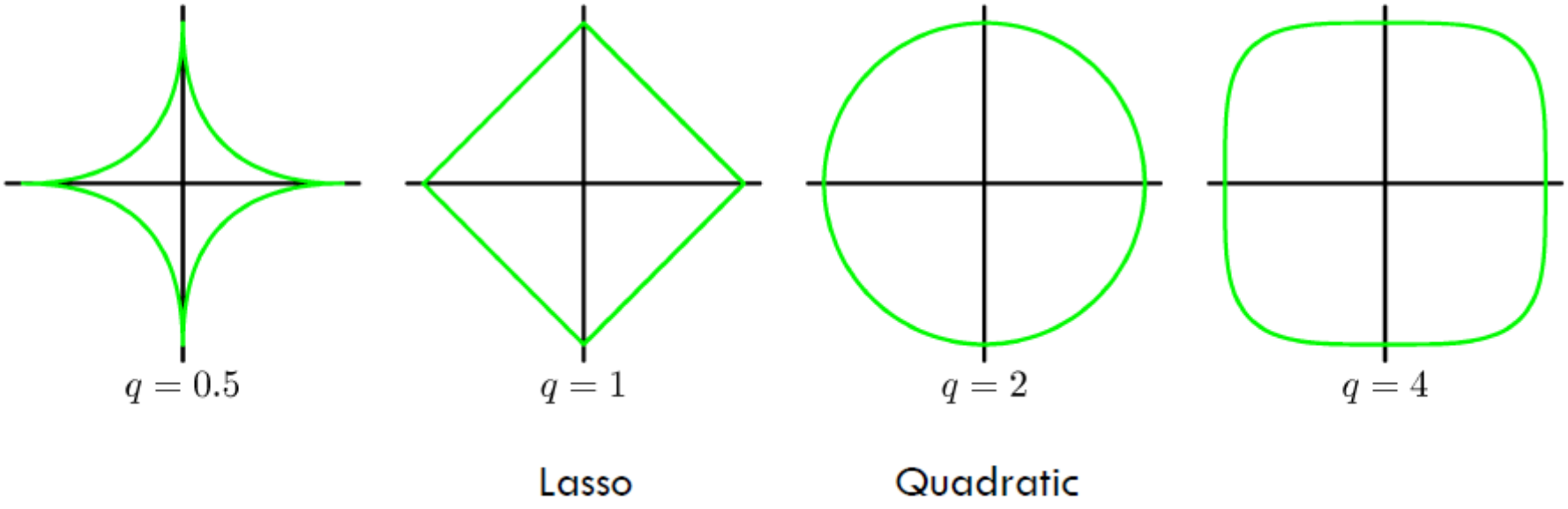
Identity matrix (m x m)

- Ridge regression or weight decay
- Since the squared weights is compatible with the squared error function, we get a nice closed form solution for the optimal weights.

More general regularizers

- Several regularized regression models

$$\frac{1}{2}\sum_{i=1}^{N}\left(w^T\phi(x_i)-t_i\right)^2+\frac{\lambda}{2}\sum_{j=1}^{m}\left|w_j\right|^q$$



L1 Regularization (LASSO)

- The regularization term is

$$E_W(w) = \frac{1}{2}\|w\|_1 = \frac{1}{2}\sum_{j=1}^{M}|w_j|$$

- Objective function

$$J(w) = \frac{1}{2}\|\Phi w - T\|_2^2 + \frac{\lambda}{2}\|w\|_1$$

- Ability to create **sparse models** that are more easily interpreted

Constrained formulation of the L1 Regularization

$$\min_{w} \left\{ \frac{1}{2} \|\Phi w - T\|_2^2 \right\} \qquad s.t. \ \|w\|_1 = 0$$

- *Least Absolute Selection and Shrinkage Operator* (**LASSO**)
- Several optimization techniques for solving the problem

An Optimization Scheme: Sequentially added sign constraints

$$\min_{w} \left\{ \frac{1}{2} \|Xw - T\|_2^2 \right\} \qquad s.t. \ \|w\|_1 \leq L$$

$$\begin{aligned}
&+w_1 + w_2 + w_3 \leq t \\
&+w_1 + w_2 - w_3 \leq t \\
&+w_1 - w_2 + w_3 \leq t \\
&+w_1 - w_2 - w_3 \leq t \\
&-w_1 + w_2 + w_3 \leq t \\
&-w_1 + w_2 - w_3 \leq t \\
&-w_1 - w_2 + w_3 \leq t \\
&-w_1 - w_2 - w_3 \leq t
\end{aligned}$$

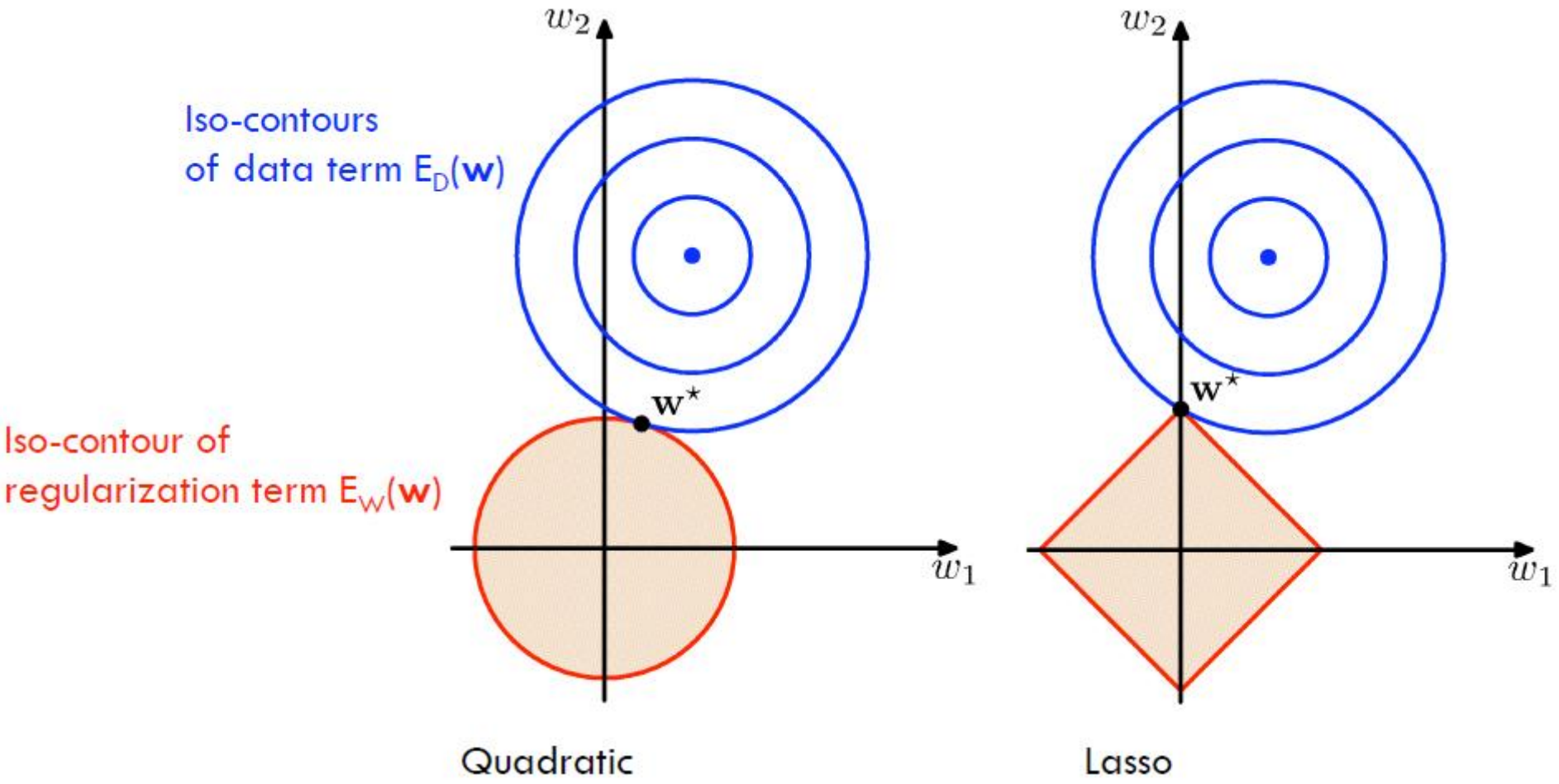
- Examine all possible combinations of the sign of the elements of w.

Algorithm 1 Tibshirani's Method

1: $w = LS(X, y)$
2: $constraints = \{null\}$
3: **while** $||w||_1 \leq L$ **do**
4: add $sign(w)$ to $constraints$
5: $w = min_w ||Xw - y||_2^2$ subject to $constraints$
6: **end while**

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of Royal. Statist. Soc B., Vol. 58, No. 1, pages 267-288).

L2 vs. L1 regularization



L2 vs. L1 regularization

L2 regularized regression	L1 regularized regression
Quadratic regularization has the advantage that the **solution is in closed form** (not appeared in non-quadratic regularizers).	Lasso represents a **convex optimization problem** solved by quadratic programming or other convex optimization methods
L2 regularization **shrinks coefficients towards (but not to) zero**, and towards each other.	L1 regularization **shrinks coefficients to zero** at different rates; different values of λ give models with different subsets of features.

Model Likelihood and Empirical Risk

- Two related but distinct ways to look at a model.
  - **Empirical Risk**: "How much **error** does the model have on the training data?"
  - **Model Likelihood**: "What is the likelihood that a model **generated** the observed data?"

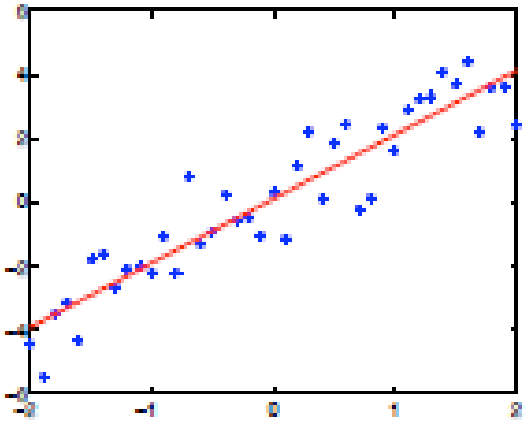
Statistical view of linear regression

- Assumption: (**Generative model**)

Observed output = constructive function + stochastic noise

$$t = y(x, w) + \varepsilon$$

- Whatever we cannot capture with our chosen family of functions will be interpreted as noise



Statistical view of linear regression

$$t = y(x, w) + \varepsilon$$

- If we consider (white) **Gaussian noise**

$$\varepsilon \sim N\left(0, \beta^{-1}\right) \text{ where } \beta = \frac{1}{\sigma^2}$$

precision or inverse variance

then we introduce the stochastic model:

$$t \mid x \sim N\left(y(x, w), \beta^{-1}\right)$$

Model Likelihood

- Training set: $D = \{(x_1, t_1), \ldots, (x_N, t_N)\}$
- Generative model: $\theta = \{w, \beta\}$

$$p(t_i \mid x_i, \theta) = N(t_i \mid y(x_i, w), \beta^{-1}) = \frac{\sqrt{\beta}}{\sqrt{2\pi}} \exp\left\{-\frac{\beta}{2}(y(x_i, w) - t_i)^2\right\}$$

- Likelihood function:

$$p(D \mid \theta) = \prod_{i=1}^{N} p(t_i \mid x_i, \theta) = \prod_{i=1}^{N} N(t_i \mid w^T \phi(x_i), \beta^{-1})$$

assuming Independently Identically distributed (***iid***) data

- **Log-likelihood**:

$$L(\theta) = \ln p(D \mid \theta) = \sum_{i=1}^{N} \ln p(t_i \mid x_i, \theta)$$

$$L(\theta) = -\frac{N}{2}\ln 2\pi + \frac{N}{2}\ln \beta - \frac{\beta}{2}\sum_{i=1}^{N}\left(y(x_i, w) - t_i\right)^2$$

- assuming a linear regression model of m basis functions

$$L(\theta = \{w, \beta\}) = -\frac{N}{2}\ln 2\pi + \frac{N}{2}\ln \beta - \frac{\beta}{2}\sum_{i=1}^{N}\left(w^T \phi(x_i) - t_i\right)^2$$

- **Maximum Likelihood (ML) estimation**

$$L(\theta=\{w,\theta\})=-\frac{N}{2}\ln 2\pi+\frac{N}{2}\ln\beta-\frac{\beta}{2}\sum_{i=1}^{N}\left(w^T\phi(x_i)-t_i\right)^2$$

- Partial derivative of linear weights (***w***):

$$\frac{\partial L(\theta)}{\partial w}=-\frac{\beta}{2}\sum_{i=1}^{N}\left(\hat{w}^T\phi(x_i)-t_i\right)\phi^T(x_i)=0$$

$$\hat{w}^T\sum_{i=1}^{N}\phi(x_i)\phi^T(x_i)-\sum_{i=1}^{N}t_i\phi^T(x_i)=0\Rightarrow\Phi^T\Phi\hat{w}=\Phi^T T$$

$$\hat{w}_{ML}=\left(\Phi^T\Phi\right)^{-1}\Phi^T T=\hat{w}_{LS}$$

- **Maximum Likelihood (ML) estimation**

$$L(\theta=\{w,\theta\})=-\frac{N}{2}\ln 2\pi+\frac{N}{2}\ln\beta-\frac{\beta}{2}\sum_{i=1}^{N}\left(w^T\phi(x_i)-t_i\right)^2$$

- Partial derivative of inverse variance (***β***):

$$\frac{\partial L(\theta)}{\partial\beta}=-\frac{N}{2\hat{\beta}}-\frac{1}{2}\sum_{i=1}^{N}\left(\hat{w}^T\phi(x_i)-t_i\right)^2=0$$

$$\frac{1}{\hat{\beta}_{ML}}=\frac{1}{N}\sum_{i=1}^{N}\left(\hat{w}_{ML}{}^T\phi(x_i)-t_i\right)^2=\frac{1}{N}\left\|\Phi\hat{w}_{ML}-T\right\|_2^2$$

Mean prediction squared error

MAP estimation of Linear Regression

- MAP derivation of linear regression assumes a **prior distribution** over linear weights w:

$$p(w|\alpha) = N(w|0, \alpha^{-1}I) = \left(\frac{\alpha}{2\pi}\right)^{M/2} \exp\left\{-\frac{\alpha}{2} w^T w\right\}$$

α: is a **hyperparameter** over **w** and is the *precision* or *inverse variance* of the distribution.

- Then, the **posterior distribution** is obtained as

$$p(w|X, T, \alpha, \beta) \propto p(T|X, w, \beta) p(w|\alpha)$$

Optimize the Bayesian posterior

$$p(w \mid X, T, \alpha, \beta) \propto p(T \mid X, w, \beta) p(w \mid \alpha)$$

- Set the **MAP log-likelihood function**:

$$L_{MAP}(\theta) = \ln p(T \mid X, w, \beta) + \ln p(w \mid \alpha)$$

likelihood + prior

Optimize the Bayesian posterior

$$p(w \mid X, T, \alpha, \beta) \propto p(T \mid X, w, \beta) p(w \mid \alpha)$$

- Set the **MAP log-likelihood** function:

$$L_{MAP}(\theta) = \ln p(T \mid X, w, \beta) + \ln p(w \mid \alpha)$$

likelihood

$$L_{ML}(\theta) = \ln p(T \mid X, w, \beta) = \sum_{i=1}^{N} \ln p(t_i \mid x_i, w, \beta) =$$

$$= -\frac{N}{2} \ln 2\pi + \frac{N}{2} \ln \beta - \frac{\beta}{2} \sum_{i=1}^{N} \left(w^T \phi(x_i) - t_i \right)^2$$

Optimize the Bayesian posterior

$$p(w \mid X, T, \alpha, \beta) \propto p(T \mid X, w, \beta) p(w \mid \alpha)$$

- Set the **MAP log-likelihood** function:

$$L_{MAP}(\theta) = \ln p(T \mid X, w, \beta) + \ln p(w \mid \alpha)$$

prior

$$\ln p(w \mid \alpha) = \ln N(w \mid 0, \alpha^{-1} I) =$$

$$= -\frac{M}{2} \ln 2\pi + \frac{M}{2} \ln \alpha - \frac{\alpha}{2} w^T w$$

Optimize the Bayesian posterior

- Ignoring terms that do not depend on w :

$$L_{MAP}(w) = \ln p(T \mid X, w, \beta) + \ln p(w \mid \alpha) =$$
$$\approx -\frac{\beta}{2}\sum_{i=1}^{N}\left(w^T \phi(x_i) - t_i\right)^2 - \frac{\alpha}{2} w^T w$$

- MAP estimation of linear weights:

$$\lambda = \frac{\alpha}{\beta}$$

$$\frac{\partial L_{MAP}(w)}{\partial w} = 0 \Rightarrow \hat{w}_{MAP} = \left(\Phi^T \Phi + \lambda I\right)\Phi^T T$$

"Thus regularized (ridge) L2 regression reflects a 0-mean isotropic Gaussian prior on the weights"

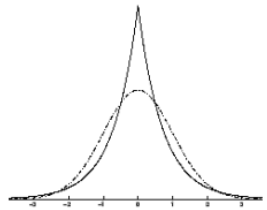
A summary of the Linear Regression techniques

Deterministic approaches	Statistical approaches
Least squares $\frac{1}{2}\sum_{i=1}^{N}\left(w^T\phi(x_i)-t_i\right)^2$	**Maximum Likelihood (ML)** $\varepsilon \sim N\left(0,\beta^{-1}\right)$ $p(t \mid w,\beta)=N\left(w^T\phi(x),\beta^{-1}\right)$

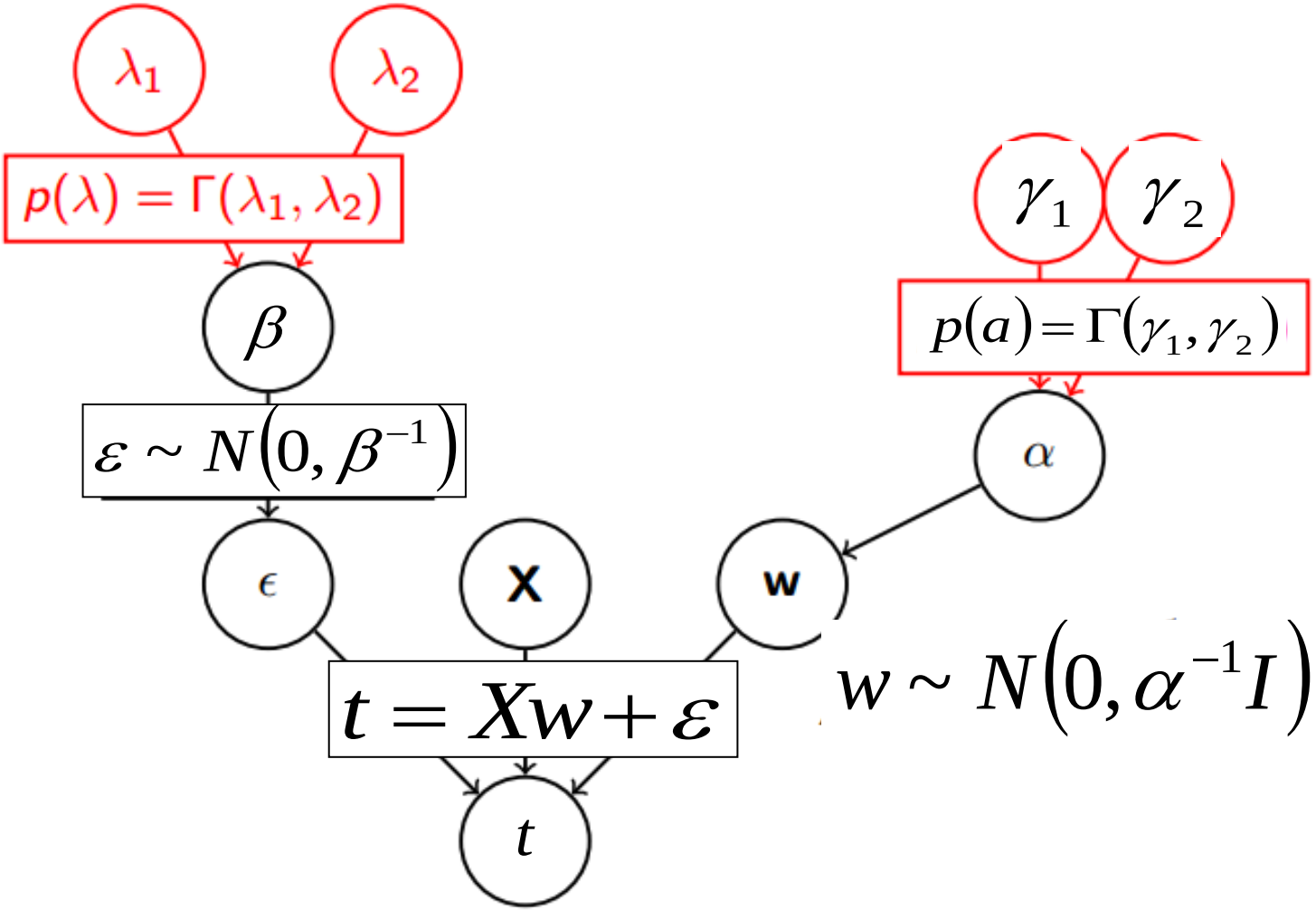
A summary of the Linear Regression techniques

Deterministic approaches	Statistical approaches
Least squares $\frac{1}{2}\sum_{i=1}^{N}\left(w^T\phi(x_i)-t_i\right)^2$	**Maximum Likelihood (ML)** $\varepsilon \sim N\left(0,\beta^{-1}\right)$ $p\left(t \mid w,\beta\right)=N\left(w^T\phi(x),\beta^{-1}\right)$
L2 regularized (ridge) $\frac{1}{2}\sum_{i=1}^{N}\left(w^T\phi(x_i)-t_i\right)^2+\frac{\lambda}{2}\sum_{j=1}^{m}\left(w_j\right)^2$	**MAP with Gaussian prior** $p\left(w \mid \alpha\right)=N\left(w \mid 0,\alpha^{-1}I\right)$ $\ln p\left(T \mid X,w,\beta\right)+\ln p\left(w \mid \alpha\right)$

A summary of the Linear Regression techniques

Deterministic approaches	Statistical approaches
Least squares $\frac{1}{2}\sum_{i=1}^{N}\left(w^T\phi(x_i)-t_i\right)^2$	**Maximum Likelihood (ML)** $\varepsilon \sim N\left(0,\beta^{-1}\right)$ $p(t\mid w,\beta)=N\left(w^T\phi(x),\beta^{-1}\right)$
L2 regularized (ridge) $\frac{1}{2}\sum_{i=1}^{N}\left(w^T\phi(x_i)-t_i\right)^2+\frac{\lambda}{2}\sum_{j=1}^{m}\left(w_j\right)^2$	**MAP with Gaussian prior** $p(w\mid\alpha)=N\left(w\mid 0,\alpha^{-1}I\right)$ $\ln p(T\mid X,w,\beta)+\ln p(w\mid\alpha)$
L1 regularized (lasso) $\frac{1}{2}\sum_{i=1}^{N}\left(w^T\phi(x_i)-t_i\right)^2+\frac{\lambda}{2}\sum_{j=1}^{m}\lvert w_j\rvert$	**MAP with Laplacian prior** $w\sim Ce^{-\lambda\lvert w\rvert_1}$ 

Hierarchical Bayesian model



Alternative Regression models

- **Elastic Net** – Complex Prior for the weights w

$$w \sim Ce^{-\lambda_1 |w|_1 + \lambda_2 |w|_2}$$

Penalization by weighted **L1 and L2** norms

$$J(w) = \frac{1}{2}\|\Phi w - T\|_2^2 + \frac{\lambda_1}{2}\|w\|_1 + \frac{\lambda_2}{2}\|w\|_2^2$$

- **Weighted Least Squares**: Assign for each case (x_i, t_i) a **weight $v_i \geq 0$** (the higher the more important the case)

$$\frac{1}{2}\sum_{i=1}^{N} v_i \left(w^T \phi(x_i) - t_i\right)^2 = \left\| V^{\frac{1}{2}} (\Phi w - T) \right\|^2$$

Then: $$\hat{w}_{WLS} = \left(\Phi^T V \Phi\right)^{-1} \Phi^T V T$$

$$V = \begin{pmatrix} v_1 & & & \\ & v_2 & & 0 \\ & & \ddots & \\ 0 & & & v_N \end{pmatrix}$$

Bayesian Linear Regression

- The previous MLE or MAP derivation of linear regression uses **point estimates** for the weight vector, **w**.

- Bayesian modeling estimates the **posterior distribution** of weights after receiving all observations.

- This allows us to find the distribution of the target of the new coming input, and thus make **prediction**.

- **Generative model:** $t = w^T \phi(x) + \varepsilon$

$$p(t \mid x, w, \beta) = N\left(t \mid w^T \phi(x), \beta^{-1}\right)$$

- Let N observations: $D = \{(x_1, t_1), \ldots, (x_N, t_N)\}$

- Then, the **join distribution** of N observations $T=\{t_1, \ldots, t_N\}$ is:

$$p(T \mid X, w) = \prod_{i=1}^{N} p(t_i \mid x_i, w) = N(T \mid \Phi w, \beta^{-1} I)$$

- We treat that **linear weights w as Gaussian random variables** with mean $\boldsymbol{m_0}$ and covariance matrix $\boldsymbol{S_0}$

$$p(w) = N(m_0, S_0)$$

- Posterior distribution: product of two Gaussians

$$p(w|T,X) \propto p(T|X,w)p(w) = N(T|\Phi w, \beta^{-1}I)N(w|m_0, S_0)$$

If we have a marginal Gaussian distribution for x and a conditional Gaussian distribution for y given x in the form

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}) \quad \text{(B.42)}$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}) \quad \text{(B.43)}$$

then the marginal distribution of y, and the conditional distribution of x given y, are given by

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}}) \quad \text{(B.44)}$$

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma}\{\mathbf{A}^{\mathrm{T}}\mathbf{L}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Lambda}\boldsymbol{\mu}\}, \boldsymbol{\Sigma}) \quad \text{(B.45)}$$

where

$$\boldsymbol{\Sigma} = (\boldsymbol{\Lambda} + \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1}. \quad \text{(B.46)}$$

C. M. Bishop, "*Pattern Recognition and Machine Learning*", page 689

- Substituting we take the general form of the posterior distribution:

$$p(w|T,X) = N(T|\Phi w, \beta^{-1}I)N(w|m_0, S_0) = N(w|m_N, S_N)$$

where:

$$S_N = \left(\beta\Phi^T\Phi + S_0^{-1}\right)^{-1} \qquad m_N = S_N\left(\beta\Phi^T T + S_0^{-1}m_0\right)$$

- **If** $m_0 = 0$ **and** $S_0 = \alpha^{-1} I$ **then**

$$S_N = \left(\beta\Phi^T\Phi + aI\right)^{-1} \qquad \lambda = \frac{\alpha}{\beta}$$

$$m_N = \left(\beta\Phi^T\Phi + \alpha I\right)^{-1}\beta\Phi^T T = \left(\Phi^T\Phi + \lambda I\right)^{-1}\Phi^T T$$

Predictive distribution

- Assuming a new input x^* we are looking for making a prediction of its target t^* . This is equivalent on estimating the posterior distribution:

$$p\left(t^* \mid x^*, T\right)$$

- Marginalizing we take:

$$p\left(t^* \mid x^*, T\right) = \int p\left(t^* \mid x^*, w\right) p\left(w \mid T\right) dw$$

where

$$p\left(w \mid T\right) = N\left(w \mid m_N, S_N\right)$$

$$m_N = \left(\beta \Phi^T \Phi + \alpha I\right)^{-1} \beta \Phi^T T$$

$$S_N = \left(\beta \Phi^T \Phi + aI\right)^{-1}$$

$$p\left(t^* \mid x^*, w\right) = N\left(t^* \mid w^T \phi\left(x^*\right), \beta^{-1}\right)$$

Predictive distribution (cont.)

$$p(t^* \mid x^*, T) = \int p(t^* \mid x^*, w)\, p(w \mid T)\, dw$$

$$p(w \mid T) = N(w \mid m_N, S_N) \qquad S_N = (\beta \Phi^T \Phi + aI)^{-1}$$

$$p(t^* \mid x^*, w) = N(t^* \mid w^T \phi(x^*), \beta^{-1})$$

- According to Gaussian properties (***B.44***) we receive:

$$p(t^* \mid x^*, T) = N(t^* \mid m_N^T \phi(x^*), \sigma_N^2(x^*))$$

where:

$$\sigma_N^2(x^*) = \beta^{-1} + \phi^T(x^*) S_N \phi(x^*)$$