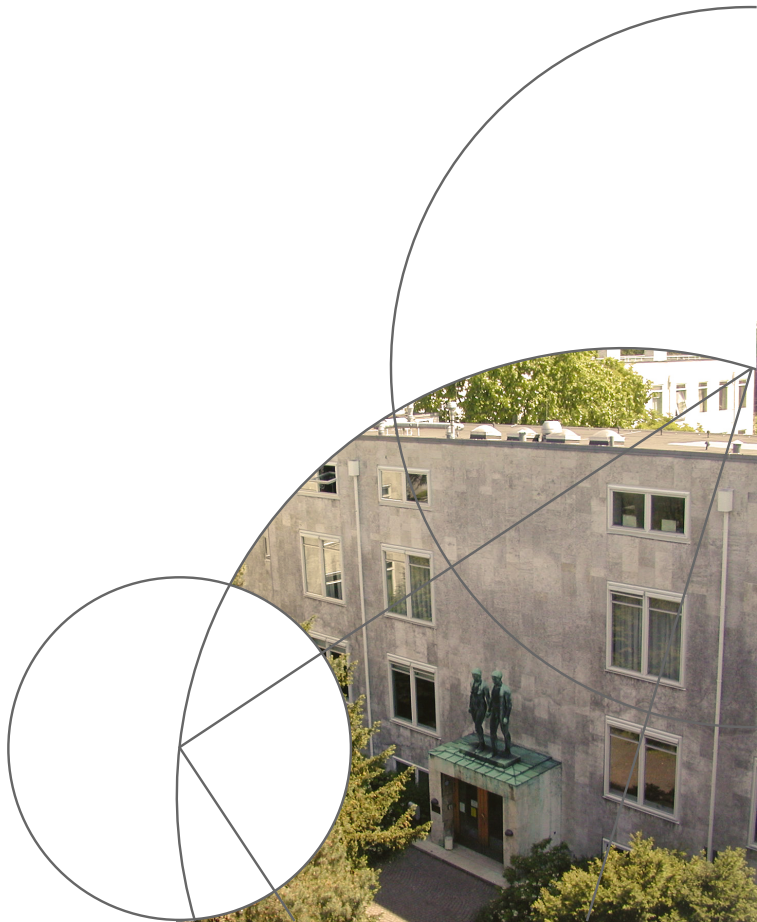
Linear Regression

Machine Learning

Christian Igel
Department of Computer Science



@christian_igel

Outline

Outline

Derivative

Let $f : \mathbb{R} \to \mathbb{R}$ be a real-valued function. If the limit

$$f'(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

exists, the function f is differentiable at the point x and f' is its derivative at x.

Differentiation rules

Let f and g be differentiable functions.

Sum. If $h(x) = f(x) + g(x)$ then h is differentiable and $h'(x) = f'(x) + g'(x)$.

Chain. If $h(x) = f(g(x))$ then h is differentiable and $h'(x) = f'(g(x))g'(x)$.

Product. If $h(x) = f(x)g(x)$ then h is differentiable and $h'(x) = f'(x)g(x) + f(x)g'(x)$.

Power. $f(x) = x^r$ with $r \in \mathbb{R}$ has derivative $f'(x) = rx^{r-1}$.

Exponential. For $f(x) = e^x$ we have $f'(x) = e^x$.

Logarithm. For $f(x) = \ln x$ we have $f'(x) = 1/x$.

Important special cases

Constant. Product rule implies $h'(x) = af'(x)$ for $h(x) = af(x)$ with $a \in \mathbb{R}$

Quotient. Product and power give for

$$h(x) = \frac{f(x)}{g(x)}$$

with non-vanishing g (i.e., g is never zero)

$$h'(x) = -\frac{f'(x)g(x) - f(x)g'(x)}{[g(x)]^2} \quad .$$

Reciprocal. Special case, we have $g'(x) = -1/x^2$ for $g(x) = 1/x$ and, more general, if $h(x) = \frac{1}{f(x)}$ for f non-vanishing, then

$$h'(x) = -\frac{f'(x)}{[f(x)]^2} \quad .$$

Partial derivative

Let $f : \mathbb{R}^d \to \mathbb{R}$ depend on d variables $(x_1, \ldots, x_d)^\mathsf{T} \in \mathbb{R}^d$. The partial derivative of f with respect to x_i at a point $\boldsymbol{x} \in \mathbb{R}^d$ is defined as

$$\frac{\partial}{\partial x_i} f(\boldsymbol{x}) = \lim_{\epsilon \to 0} \frac{f(\boldsymbol{x} + \epsilon \boldsymbol{e}_i) - f(\boldsymbol{x})}{\epsilon}$$

with $\boldsymbol{e}_i$ being the ith unit vector (i.e., a vector where the ith component is one and all other components are zero) given that the limit exists. For $d = 1$ we have

$$\frac{\partial}{\partial x} f(x) = f'(x) \ .$$

Gradient

- *Rate of change* (*directional derivative*) of $f : \mathbb{R}^d \to \mathbb{R}$ at a point $\boldsymbol{x} \in \mathbb{R}^d$ when moving in the direction $\boldsymbol{u} \in \mathbb{R}^d$, $\|\boldsymbol{u}\| = 1$, is defined as:

$$\nabla_{\boldsymbol{u}} f(\boldsymbol{x}) = \lim_{\epsilon \to 0} \frac{f(\boldsymbol{x} + \epsilon \boldsymbol{u}) - f(\boldsymbol{x})}{\epsilon}$$

- The *gradient*

$$\nabla f(\boldsymbol{x}) = \left(\frac{\partial f(\boldsymbol{x})}{\partial x_1}, \frac{\partial f(\boldsymbol{x})}{\partial x_2}, \ldots, \frac{\partial f(\boldsymbol{x})}{\partial x_d} \right)^{\mathsf{T}}$$

  points in the direction $\nabla f(\boldsymbol{x}) / \|\nabla f(\boldsymbol{x})\|$ giving maximum rate of change $\|\nabla f(\boldsymbol{x})\|$.

Chain rule

The *chain rule* for computing the derivative of a composition of two functions,

$$\frac{\partial f(g(x))}{\partial x} = f'(g(x))g'(x)$$

with $f'(x) = \frac{\partial f(x)}{\partial x}$ and $g'(x) = \frac{\partial g(x)}{\partial x}$, can be extended to:

$$\frac{\partial f(g_1(x), g_2(x), \ldots, g_d(x))}{\partial x} = \sum_{i=1}^{d} \frac{\partial f(g_1(x), \ldots, g_d(x))}{\partial g_i(x)} \frac{\partial g_i(x)}{\partial x}$$

Outline

The linear regression problem

- The goal is to predict a target variable $\boldsymbol{y} \in \mathbb{R}^m$ based on given data $\boldsymbol{x} \in \mathcal{X} = \mathbb{R}^d$. For simplicity, let $m = 1$.
- We build affine linear predictive models

$$y = f(\boldsymbol{x}) = \sum_{i=1}^{d} w_i x_i + b = \boldsymbol{w}^\mathsf{T}\boldsymbol{x} + b$$

  with $\boldsymbol{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$.
- For convenience, we define $\tilde{\boldsymbol{x}}^\mathsf{T} = (x_1, \ldots, x_d, 1)$ and $\tilde{\boldsymbol{w}}^\mathsf{T} = (w_1, \ldots, w_d, b)$ and consider the equivalent formulation

$$y = f(\tilde{\boldsymbol{x}}) = \sum_{i=1}^{d+1} \tilde{w}_i \tilde{x}_i = \tilde{\boldsymbol{w}}^\mathsf{T}\tilde{\boldsymbol{x}} \quad .$$

Data matrix

Training data $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ is gathered in data matrix (x_{ij} is the jth component of the ith training pattern, i.e., $[\boldsymbol{x}_i]_j$)

$$\tilde{\boldsymbol{X}} = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1d} & 1 \\ x_{21} & x_{22} & \ldots & x_{2d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{nd} & 1 \end{pmatrix}$$

Let's omit the tilde to keep notation uncluttered.

Formalizing the goal

- We want to minimize the empirical risk under the squared loss $\ell(y, \hat{y}) = (y - \hat{y})^2$:

$$\frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i) = \frac{1}{n}\sum_{i=1}^{n} \left(y_i - \boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i\right)^2$$

  This quantity is also known as mean-squared-error (MSE).
- If $\boldsymbol{X}^\mathsf{T}\boldsymbol{X}$ has full rank, optimal parameters are given by

$$\boldsymbol{w}^\star = \underbrace{\left(\boldsymbol{X}^\mathsf{T}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\mathsf{T}}_{\boldsymbol{X}^\dagger}\boldsymbol{y} \ ,$$

  where $\boldsymbol{y} = (y_1, \ldots, y_n)^\mathsf{T}$.

Linear regression algorithm

Algorithm 1: linear regression

Input: $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\} \in (\mathbb{R}^d \times \mathbb{R})^n$

Output: affine linear model

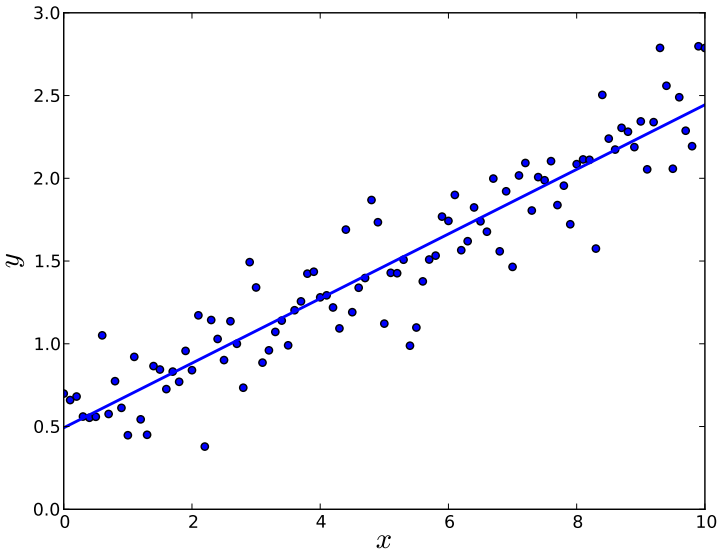
1 $\boldsymbol{y} = (y_1, \ldots, y_n)^\mathsf{T}$

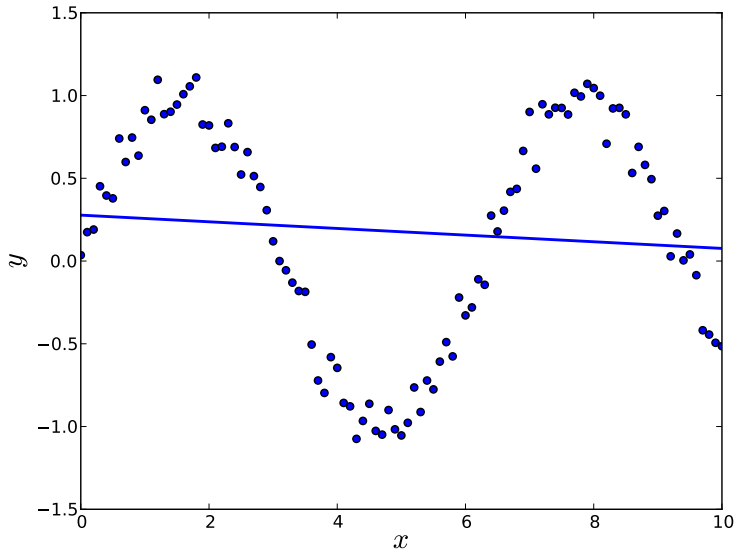
2 $\tilde{\boldsymbol{X}} = \begin{pmatrix} x_{11} & \ldots & x_{1d} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ x_{n1} & \ldots & x_{nd} & 1 \end{pmatrix}$

3 $(w_1, \ldots, w_d, b)^\mathsf{T} = \tilde{\boldsymbol{X}}^\dagger \boldsymbol{y}$

Result: $(\boldsymbol{w}, b)$, defining the model $f(\boldsymbol{x}) = \boldsymbol{w}^\mathsf{T}\boldsymbol{x} + b$

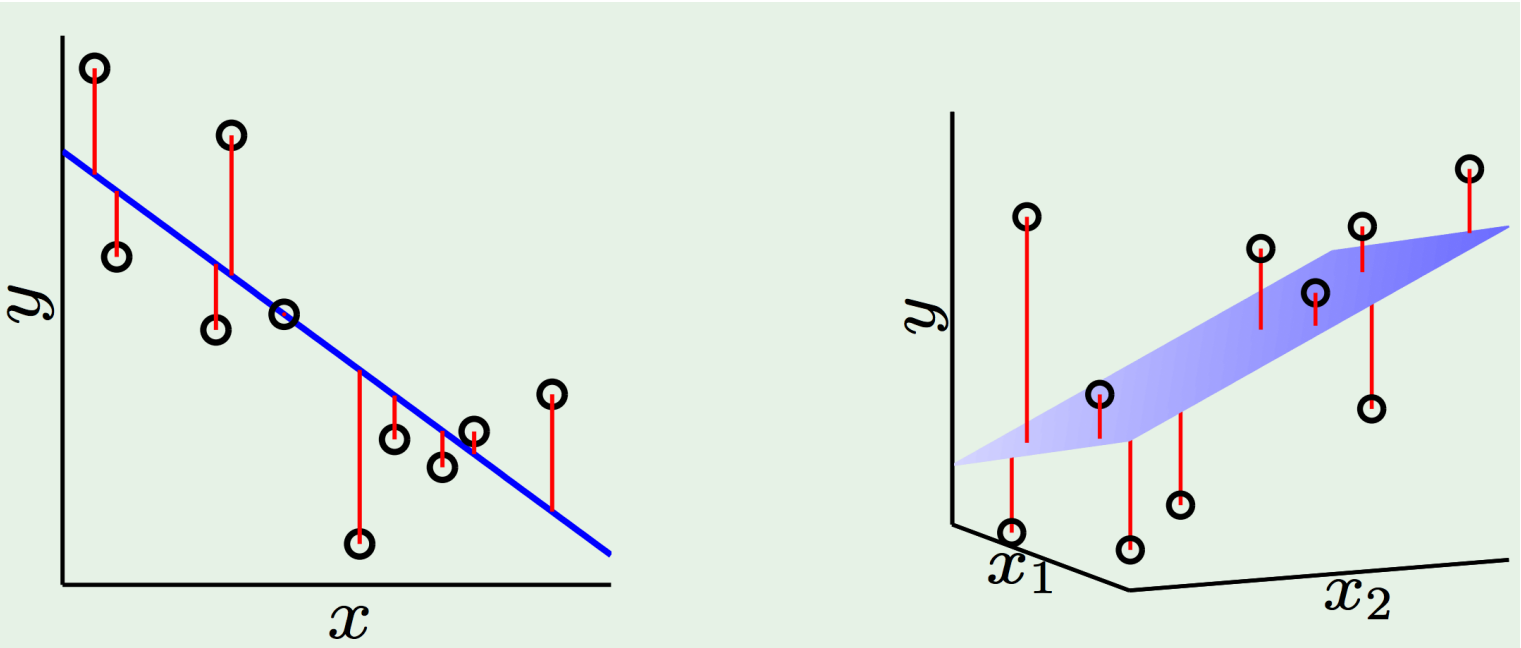
Examples





The right plot is an example of "underfitting".

Visualizing the error



Abu-Mostafa, Magdon-Ismail, and Lin. *Learning from Data*. AMLbook, 2012

Outline

Overdetermined/unique solution case I

Let's assume $\boldsymbol{X}^\mathsf{T}\boldsymbol{X}$ has full rank, implying $n \geq d$. Setting the derivative

$$\nabla_{\boldsymbol{w}} \frac{1}{2} \sum_{i=1}^{n} \left(y_i - \boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i\right)^2 = -\sum_{i=1}^{n} \left(y_i - \boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i\right) \boldsymbol{x}_i$$

of ($n/2$ times) the empirical risk to $\mathbf{0}$ yields

$$\mathbf{0} = \sum_{i=1}^{n} y_i \boldsymbol{x}_i - \sum_{i=1}^{n} \boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i \boldsymbol{x}_i \ ,$$

which implies with $\boldsymbol{y} = (y_1, \ldots, y_n)^\mathsf{T}$

$$\mathbf{0} = \sum_{i=1}^{n} y_i \boldsymbol{x}_i - \sum_{i=1}^{n} \left[\boldsymbol{X}\boldsymbol{w}\right]_i \boldsymbol{x}_i = \boldsymbol{X}^\mathsf{T}\boldsymbol{y} - \boldsymbol{X}^\mathsf{T}\boldsymbol{X}\boldsymbol{w} \ .$$

Overdetermined/unique solution case II

$$\mathbf{0} = \boldsymbol{X}^\mathsf{T}\boldsymbol{y} - \boldsymbol{X}^\mathsf{T}\boldsymbol{X}\boldsymbol{w}$$

implies

$$\left(\boldsymbol{X}^\mathsf{T}\boldsymbol{X}\right)\boldsymbol{w} = \boldsymbol{X}^\mathsf{T}\boldsymbol{y} \ .$$

Assuming $\boldsymbol{X}^\mathsf{T}\boldsymbol{X}$ has full rank, we get the maximum likelihood estimate:

$$\boldsymbol{w}^\star = \underbrace{\left(\boldsymbol{X}^\mathsf{T}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\mathsf{T}}_{\boldsymbol{X}^\dagger}\boldsymbol{y}$$

$\boldsymbol{X}^\dagger$ is called a *Moore-Penrose pseudo-inverse*.

Underdetermined case

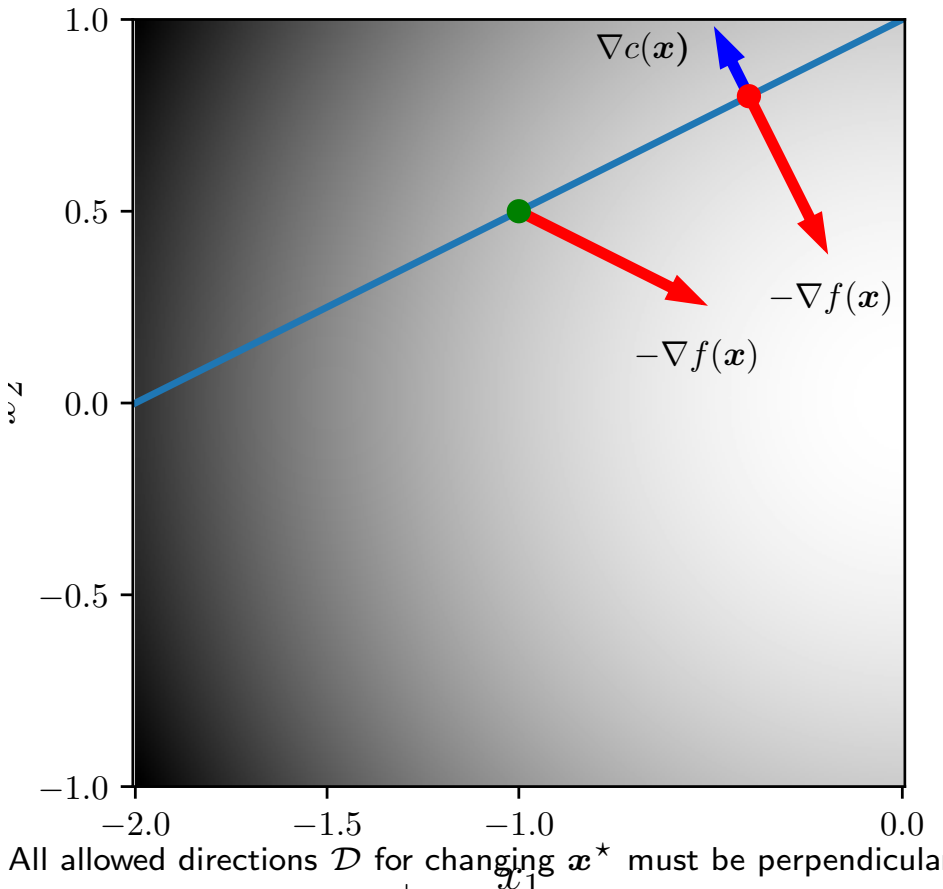
If the system is overdetermined ($n < d$), we take the interpolating solution (function that goes through all points) with the smallest (squared) L_2 norm:

$$\boldsymbol{w}^\star = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \|\boldsymbol{w}\|^2 \quad \text{s.t.} \quad \forall i : \boldsymbol{w}^\mathsf{T}\boldsymbol{x_i} - y_i = 0$$

This is a convex quadratic optimization problem with linear equality constraints.

Treating equality constraints (intuition)

Minimize $f : \mathbb{R}^n \to \mathbb{R}$ subject to $c(\boldsymbol{x}) = 0$ (blue line)



Necessary for some scalar λ:

$$\nabla f(\boldsymbol{x}^\star) = \lambda \nabla c(\boldsymbol{x}^\star)$$

Multiple constraints c_i, $i = 1, \ldots, k$:

$$\nabla f(\boldsymbol{x}^\star) = \sum_{i=1}^{k} \lambda_i \nabla c_i(\boldsymbol{x}\star)$$

All allowed directions $\mathcal{D}$ for changing $\boldsymbol{x}^\star$ must be perpendicular to $\nabla f(\boldsymbol{x}^\star)$, otherwise $\boldsymbol{x}^\star$ would not be a minimum: $\nabla f(\boldsymbol{x}^\star) \in \mathcal{D}^\perp$. All allowed directions are along the constraints, i.e., perpendicular to $\mathcal{S} = \text{span}\{\nabla c_1(\boldsymbol{x}^\star), \ldots, \nabla c_k(\boldsymbol{x}^\star)\}$. Thus, $\mathcal{D} = \mathcal{S}^\perp$ and $\mathcal{D}^\perp = \mathcal{S}$. Therefore $\nabla f(\boldsymbol{x}^\star) \in \mathcal{S}$.

Solving constrained convex problem I

We have ($f(\boldsymbol{w}) = \|\boldsymbol{w}\|^2$, $c_i(\boldsymbol{w}) = \boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i - y_i$):

$$\nabla_{\boldsymbol{w}} \|\boldsymbol{w}\|^2 = 2\boldsymbol{w}$$

(i.e., $\nabla f(\boldsymbol{w}) = 2\boldsymbol{w}$) and

$$\nabla_{\boldsymbol{w}} \left(\boldsymbol{w}^\mathsf{T}\boldsymbol{x_i} - y_i\right) = \boldsymbol{x}_i$$

for all i (i.e., $\nabla c_i(\boldsymbol{w}) = \boldsymbol{x}_i$). We have

$$\sum_{i=1}^{n} \lambda_i \boldsymbol{x}_i = \boldsymbol{X}^\mathsf{T}\boldsymbol{\lambda}$$

for $\boldsymbol{\lambda} \in \mathbb{R}^d$. Thus, we get the necessary condition:

$$2\boldsymbol{w} - \boldsymbol{X}^\mathsf{T}\boldsymbol{\lambda} = 0$$

Solving constrained convex problem II

Constraints require ($\boldsymbol{y} = (y_1, \ldots, y_d)$):

$$\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y} = \boldsymbol{0}$$

From $2\boldsymbol{w} - \boldsymbol{X}^\mathsf{T}\boldsymbol{\lambda} = \boldsymbol{0}$ we get

$$\boldsymbol{w} = \frac{1}{2}\boldsymbol{X}^\mathsf{T}\boldsymbol{\lambda} \ .$$

Plugging this into the above equation and solving for $\boldsymbol{\lambda}$ gives

$$\frac{1}{2}\boldsymbol{X}\boldsymbol{X}^\mathsf{T}\boldsymbol{\lambda} - \boldsymbol{y} = \boldsymbol{0} \quad \Rightarrow \quad \boldsymbol{\lambda} = 2\left(\boldsymbol{X}\boldsymbol{X}^\mathsf{T}\right)^{-1}\boldsymbol{y}$$

resulting in:

$$\boldsymbol{w}^\star = \underbrace{\boldsymbol{X}^\mathsf{T}\left(\boldsymbol{X}\boldsymbol{X}^\mathsf{T}\right)^{-1}}_{\boldsymbol{X}^\dagger}\boldsymbol{y}$$

Moore-Penrose pseudoinverse

A Moore-Penrose pseudoinverse $\boldsymbol{X}^\dagger \in \mathbb{R}^{d \times n}$ of a matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ has the properties

$$\boldsymbol{X}\boldsymbol{X}^\dagger\boldsymbol{X} = \boldsymbol{X}$$
$$\boldsymbol{X}^\dagger\boldsymbol{X}\boldsymbol{X}^\dagger = \boldsymbol{X}^\dagger$$
$$\left(\boldsymbol{X}\boldsymbol{X}^\dagger\right)^\mathsf{T} = \boldsymbol{X}\boldsymbol{X}^\dagger$$
$$\left(\boldsymbol{X}^\dagger\boldsymbol{X}\right)^\mathsf{T} = \boldsymbol{X}^\dagger\boldsymbol{X}$$

We can compute the pseudoinverse via matrix decompositions, for example via singular value decomposition (SVD).

Singular value decomposition (SVD)

Any real $n \times d$ matrix $\boldsymbol{X}$ can be decomposed as

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Gamma}\boldsymbol{V}^\mathsf{T}$$

with for $r = \min\{n, d\}$

- $\boldsymbol{U} \in \mathbb{R}^{n \times r}$ semi-orthogonal,
- $\boldsymbol{\Gamma} = \operatorname{diag}(\gamma_1, \ldots, \gamma_r) \in \mathbb{R}^{r \times r}$ with $\gamma_1 \geq \gamma_2 \geq \cdots \geq 0$,
- $\boldsymbol{V} \in \mathbb{R}^{d \times r}$ semi-orthogonal.

A $n \times d$ matrix is *semi-orthogonal*, if for $n < d$ the rows are orthonormal and for $d < n$ the columns are orthonormal.

SVD and pseudoinverse

$n \geq d$: $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Gamma}\boldsymbol{V}^\mathsf{T}$ with $\boldsymbol{U} \in \mathbb{R}^{n \times d}$ and $\boldsymbol{\Gamma}, \boldsymbol{V} \in \mathbb{R}^{d \times d}$

$$
\begin{aligned}
\boldsymbol{w}^\star &= \left(\boldsymbol{X}^\mathsf{T}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{y} \\
&= \left(\boldsymbol{V}\boldsymbol{\Gamma}^\mathsf{T}\boldsymbol{U}^\mathsf{T}\boldsymbol{U}\boldsymbol{\Gamma}\boldsymbol{V}^\mathsf{T}\right)^{-1}\boldsymbol{V}\boldsymbol{\Gamma}^\mathsf{T}\boldsymbol{U}^\mathsf{T}\boldsymbol{y} \\
&= \left(\boldsymbol{V}\boldsymbol{\Gamma}^\mathsf{T}\boldsymbol{\Gamma}\boldsymbol{V}^\mathsf{T}\right)^{-1}\boldsymbol{V}\boldsymbol{\Gamma}^\mathsf{T}\boldsymbol{U}^\mathsf{T}\boldsymbol{y} \\
&= \left(\boldsymbol{\Gamma}\boldsymbol{V}^\mathsf{T}\right)^{-1}\left(\boldsymbol{V}\boldsymbol{\Gamma}^\mathsf{T}\right)^{-1}\boldsymbol{V}\boldsymbol{\Gamma}^\mathsf{T}\boldsymbol{U}^\mathsf{T}\boldsymbol{y} && \boldsymbol{B}^{-1}\boldsymbol{A}^{-1} = (\boldsymbol{A}\boldsymbol{B})^{-1} \\
&= \left(\boldsymbol{\Gamma}\boldsymbol{V}^\mathsf{T}\right)^{-1}\boldsymbol{U}^\mathsf{T}\boldsymbol{y} \\
&= \underbrace{\boldsymbol{V}\boldsymbol{\Gamma}^{-1}\boldsymbol{U}^\mathsf{T}}_{\boldsymbol{X}^\dagger}\boldsymbol{y} && \boldsymbol{V} \text{ is orthogonal}
\end{aligned}
$$

SVD and pseudoinverse

$n < d$: $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Gamma}\boldsymbol{V}^\mathsf{T}$ with $\boldsymbol{U}, \boldsymbol{\Gamma} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{V} \in \mathbb{R}^{d \times n}$

$$
\begin{aligned}
\boldsymbol{w}^\star &= \boldsymbol{X}^\mathsf{T}\left(\boldsymbol{X}\boldsymbol{X}^\mathsf{T}\right)^{-1}\boldsymbol{y} \\
&= \boldsymbol{V}\boldsymbol{\Gamma}^\mathsf{T}\boldsymbol{U}^\mathsf{T}\left(\boldsymbol{U}\boldsymbol{\Gamma}\boldsymbol{V}^\mathsf{T}\boldsymbol{V}\boldsymbol{\Gamma}^\mathsf{T}\boldsymbol{U}^\mathsf{T}\right)^{-1}\boldsymbol{y} \\
&= \boldsymbol{V}\boldsymbol{\Gamma}^\mathsf{T}\boldsymbol{U}^\mathsf{T}\left(\boldsymbol{U}\boldsymbol{\Gamma}\boldsymbol{\Gamma}^\mathsf{T}\boldsymbol{U}^\mathsf{T}\right)^{-1}\boldsymbol{y} \\
&= \boldsymbol{V}\boldsymbol{\Gamma}^\mathsf{T}\boldsymbol{U}^\mathsf{T}\left(\boldsymbol{\Gamma}^\mathsf{T}\boldsymbol{U}^\mathsf{T}\right)^{-1}\left(\boldsymbol{U}\boldsymbol{\Gamma}\right)^{-1}\boldsymbol{y} \\
&= \boldsymbol{V}\left(\boldsymbol{U}\boldsymbol{\Gamma}\right)^{-1}\boldsymbol{y} \\
&= \underbrace{\boldsymbol{V}\boldsymbol{\Gamma}^{-1}\boldsymbol{U}^\mathsf{T}}_{\boldsymbol{X}^\dagger}\boldsymbol{y}
\end{aligned}
$$

Why sum-of-squares?

- Sum-of-squares error has the nice properties of
  - being differentiable and
  - penalizing large deviations from a target value more than proportionally stronger than small deviations.
- There are theoretical reasons ...

Likelihood function

- Likelihood (function) of the parameters $\boldsymbol{w}$ given training data S is the probability of observing S when the data is generated by h with parameters $\boldsymbol{w}$.
- Likelihood for i.i.d. S:

$$\prod_{i=1}^{n} P(Y = y_i \,|\, X = \boldsymbol{x}_i; h) \text{ or short } \prod_{i=1}^{n} P(y_i \,|\, \boldsymbol{x}_i)$$

- Negative log(arithmic) likelihood:

$$-\log \prod_{i=1}^{n} P(y_i \,|\, \boldsymbol{x}_i) = -\sum_{i=1}^{n} \log P(y_i \,|\, \boldsymbol{x}_i)$$

- Learning principle: Negative log-likelihood minimization

Sum-of-squares and Gaussian noise I

Consider a deterministic real-valued target functions perturbed by zero-mean additive Gaussian noise with variance σ^2, then we have

$$p(y \mid \boldsymbol{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-f(\boldsymbol{x}))^2}{2\sigma^2}} \quad .$$

Given $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ the likelihood of the model $h(x) = \boldsymbol{w}^\mathsf{T}\boldsymbol{x}$ is

$$p((y_1, \ldots, y_n)^\mathsf{T} \mid (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n), \boldsymbol{w}, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i)^2}{2\sigma^2}} \quad .$$

Sum-of-squares and Gaussian noise II

Let the logarithm turn products into sums:

$$\begin{aligned}\ln p(\boldsymbol{y} \mid \boldsymbol{w}) &= \ln \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i)^2}{2\sigma^2}} \\ &= \sum_{i=1}^{n} \ln \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i)^2}{2\sigma^2}} \\ &= \underbrace{-\frac{n}{2}\ln\sigma^2 - \frac{n}{2}\ln(2\pi)}_{\text{independent of } \boldsymbol{w}} - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i - \boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i\right)^2\end{aligned}$$

Thus, maximizing the likelihood corresponds to minimizing MSE under the assumption of Gaussian noise.

Why Gaussian noise? Central limit theorem, maximum entropy principle

Outline

Non-linear transformations

Consider a non-linear transformation

$$\Phi : \mathcal{X} \to \mathcal{Z} \ .$$

Then we can consider hypotheses

$$h(\boldsymbol{x}) = \tilde{h}(\Phi(\boldsymbol{x})) = \tilde{\boldsymbol{w}}^{\mathsf{T}}\Phi(\boldsymbol{x})$$

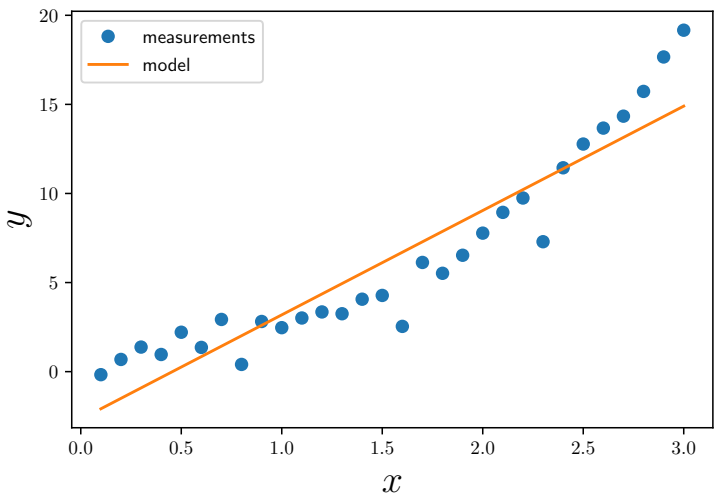
and do linear regression with the data matrix:

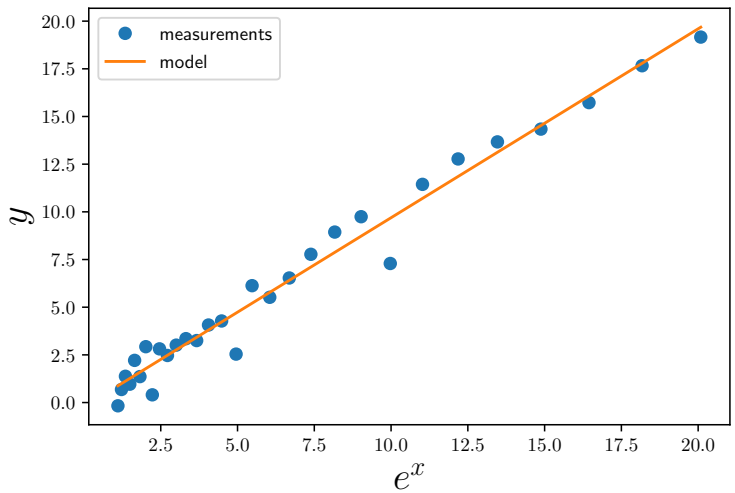
$$\begin{pmatrix} [\Phi(\boldsymbol{x}_1)]_1 & [\Phi(\boldsymbol{x}_1)]_2 & \dots & 1 \\ [\Phi(\boldsymbol{x}_2)]_1 & [\Phi(\boldsymbol{x}_2)]_2 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ [\Phi(\boldsymbol{x}_n)]_1 & [\Phi(\boldsymbol{x}_2)]_2 & \dots & 1 \end{pmatrix}$$

Example I

$$\Phi : \mathbb{R} \to \mathbb{R}$$

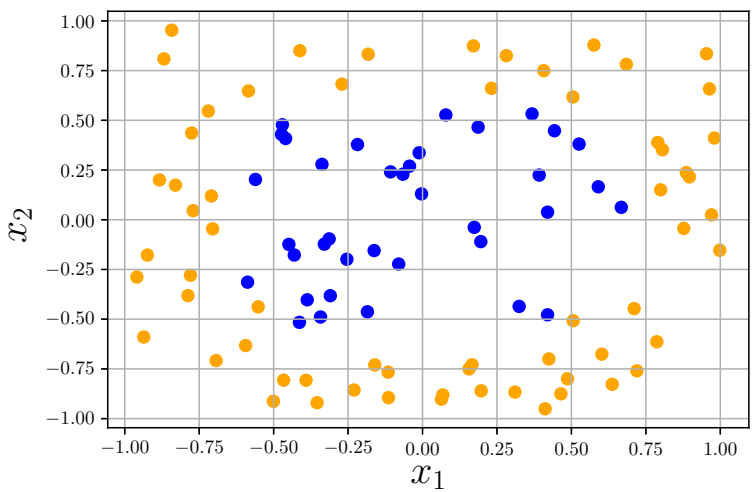
$$x \mapsto \exp x$$

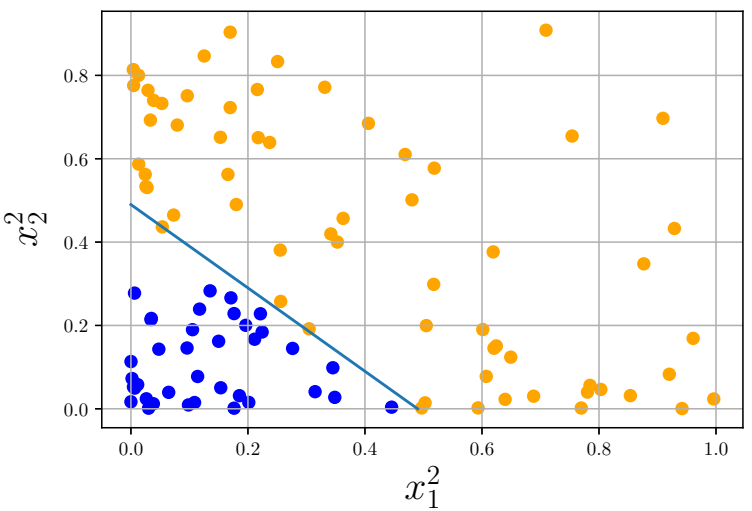




Example II

$$\Phi : \mathbb{R}^2 \to \mathbb{R}^3$$
$$(x_1, x_2) \mapsto (x_1^2, x_2^2)$$





$(x_1, x_2)^\mathsf{T}$ $(x_1^2, x_2^2)^\mathsf{T}$

Outline

Small sample problem

Consider the case $n \leq d$. Then linear regression always finds a model with zero empirical risk by solving the system of linear equations:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w}$$

Even if the y_i are independent of the $\boldsymbol{x}_i$!

Overfitting

Overfitting occurs if the model faithfully reflects idiosyncrasies of the training data rather than the underlying process that has generated the data. This may happen if the model is too complex/flexible in relation to the amount of available information.

Formal concepts of model complexity and flexibility exist.

Outline

Empirical risk minimization

So far, we performed *empirical risk minimization*:

$$\text{argmin}_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, h(\boldsymbol{x}_i))$$

In the linear regression case, $\ell(y, y') = (y - y')^2$ and $\mathcal{H} = \{h(\boldsymbol{x}) = \boldsymbol{w}^\mathsf{T}\boldsymbol{x} \,|\, \boldsymbol{w} \in \mathbb{R}^d\}$.

Adding a transformation $\Phi : \mathcal{X} \to \mathcal{Z}$ gives the hypotheses space

$$\mathcal{H} = \left\{ h(\boldsymbol{x}) = \boldsymbol{w}^\mathsf{T}\Phi(\boldsymbol{x}) \,\middle|\, \boldsymbol{w} \in \mathbb{R}^{\tilde{d}} \right\}$$

where $\tilde{d}$ is the feature space dimensionality.

Regularization & model selection

Let $\mathcal{X} = \mathbb{R}$ and

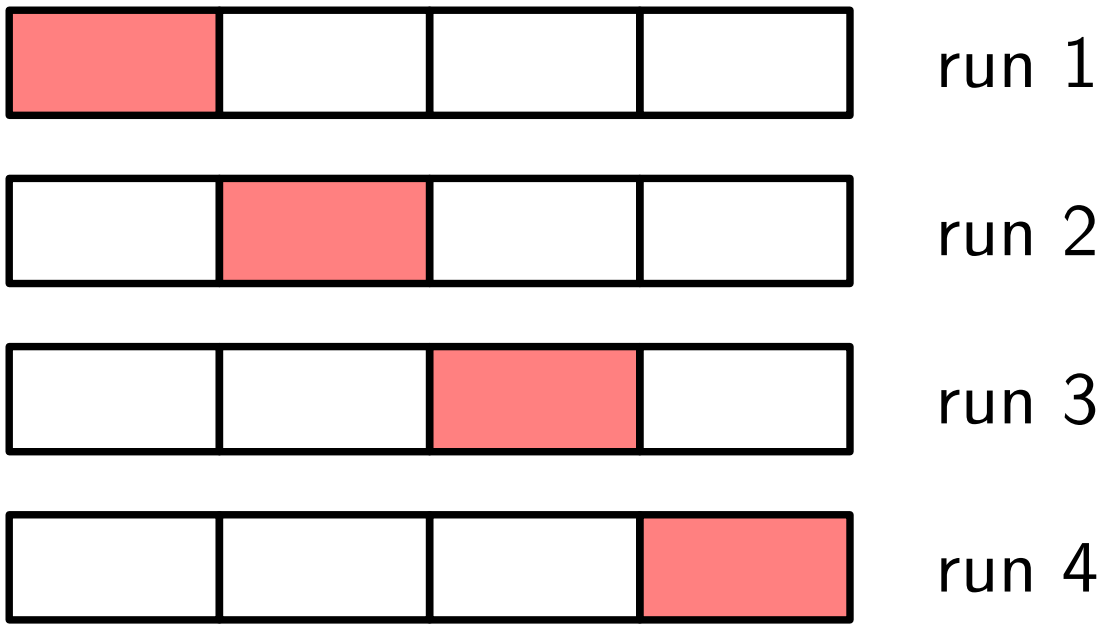
$$\Phi_k : x \mapsto (1, x, x^2, \ldots, x^k)^\mathsf{T} \ .$$

When k is too small, we may underfit. If k and thus $\tilde{d}$ is too large, we may overfit.

How to select the right k? This is often referred to a *model selection* problem.

$\rightarrow$ One way is using (cross-)validation.

Excursus: Cross-validation



C. M. Bishop. *Pattern Recognition and Machine Learning*, Springer, 2006

- K-fold cross-validation (K-CV): Split training data into K (almost) even parts
- Train K times on $K-1$ parts and test on the held-out part
- Take hyperparameters maximizing mean test performance for building the model using all data

Always remember!

Perhaps most important rule

As a general rule, you must not use the "test data" (i.e., data you use for estimating the final performance of a model) in the model building process at all (neither for training, data normalization, nor hyperparameter selection), because otherwise you may get a biased estimate of the generalization performance of the model.

Two-norm regularization

Let us restrict the hypothesis space to

$$\mathcal{H}_C = \left\{ h(\boldsymbol{x}) = \boldsymbol{w}^\mathsf{T} \Phi(\boldsymbol{x}) \,\middle|\, \boldsymbol{w} \in \mathbb{R}^{\tilde{d}} \wedge \|\boldsymbol{w}\| \leq C \right\}$$

with $C > 0$.

For simplicity, assume $\mathcal{X} = \mathbb{R}$, $\tilde{d} = 1$, and Φ to be Lipschitz with constant α (i.e., $|\Phi(x_1) - \Phi(x_2)| \leq \alpha |x_1 - x_2|$ for $x_1, x_2 \in \mathbb{R}$).

Then the functions in $\mathcal{H}_C$ are Lipschitz with constant $C \cdot \alpha$.

That is, C limits how fast the functions can change – it constrains a notion of complexity of the hypotheses class.

Equivalence

There exists a λ dependent on C such that

$$\text{argmin}_{\boldsymbol{w}\in\mathbb{R}^{\tilde{d}}} \frac{1}{n}\sum_{i=1}^{n}(y_i - \boldsymbol{w}^\mathsf{T}\Phi(\boldsymbol{x}))^2 \quad \text{s.t.} \quad \|\boldsymbol{w}\| \leq C$$

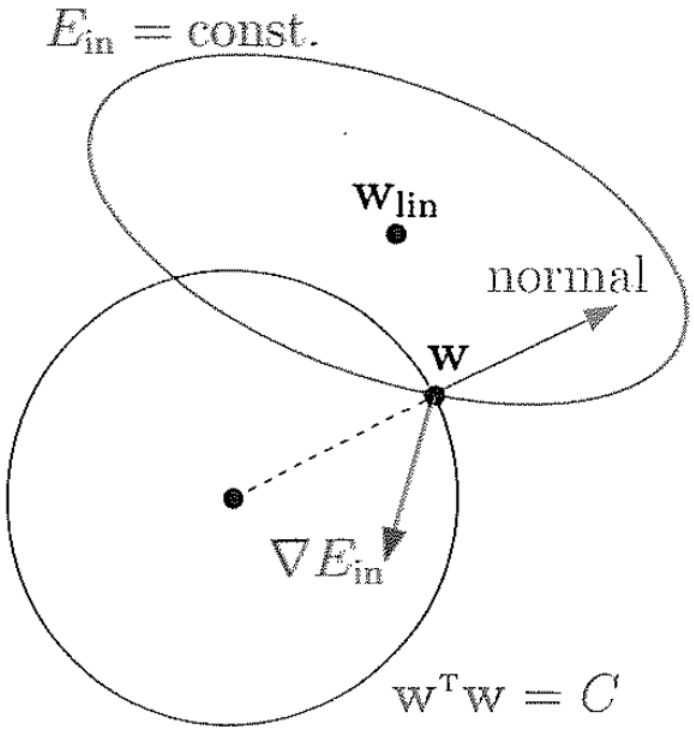
is equivalent to:

$$\text{argmin}_{\boldsymbol{w}\in\mathbb{R}^{\tilde{d}}} \left[\frac{1}{n}\sum_{i=1}^{n}(y_i - \boldsymbol{w}^\mathsf{T}\Phi(\boldsymbol{x}))^2 + \lambda\boldsymbol{w}^\mathsf{T}\boldsymbol{w}\right]$$

The *regularizarion parameter* λ can be chosen by (cross-)validation.

Linear regression with two-norm regularization is often called *ridge regression*.

Equivalence proof sketch



E_{in} is the empirical risk minimized by $\mathbf{w}_{\text{lin}} = \boldsymbol{w}^\star$.

$$\frac{\partial}{\partial \boldsymbol{w}} \left[\frac{1}{n} \sum_{i=1}^{n} (y_i - f(\boldsymbol{x}))^2 + \lambda \boldsymbol{w}^{\mathsf{T}} \boldsymbol{w} \right] = 0$$

$$\Rightarrow \frac{\partial}{\partial \boldsymbol{w}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(\boldsymbol{x}))^2 = -\lambda \boldsymbol{w}^{\mathsf{T}}$$

$$\Rightarrow \nabla E_{\text{in}} = -\lambda \boldsymbol{w}^\star$$

Abu-Mostafa, Magdon-Ismail, and Lin. *Learning from Data*. AMLbook, 2012

Solving ridge regression

We had

$$\frac{\partial}{\partial \boldsymbol{w}} \frac{1}{2} \sum_{i=1}^{n} \left(y_i - \boldsymbol{w}^\mathsf{T} \boldsymbol{x}_i \right)^2 = \boldsymbol{y}^\mathsf{T} \boldsymbol{X} - \boldsymbol{w}^\mathsf{T} \boldsymbol{X}^\mathsf{T} \boldsymbol{X}$$

and thus

$$\frac{\partial}{\partial \boldsymbol{w}} \left[\frac{1}{2} \sum_{i=1}^{n} \left(y_i - \boldsymbol{w}^\mathsf{T} \boldsymbol{x}_i \right)^2 + \lambda \boldsymbol{w}^\mathsf{T} \boldsymbol{w} \right] = \boldsymbol{y}^\mathsf{T} \boldsymbol{X} - \boldsymbol{w}^\mathsf{T} \boldsymbol{X}^\mathsf{T} \boldsymbol{X} + \lambda \boldsymbol{w}^\mathsf{T}$$

which is solved by:

$$\boldsymbol{w}^\star = \left(\boldsymbol{X}^\mathsf{T} \boldsymbol{X} + \lambda \boldsymbol{I} \right)^{-1} \boldsymbol{X}^\mathsf{T} \boldsymbol{y}$$

What happens to $\boldsymbol{w}^\star$ for $\lambda \to \infty$?

One-norm regularization

The one-norm can be used instead of the two-norm:

$$\text{argmin}_{\boldsymbol{w}\in\mathbb{R}^{\tilde{d}}} \left[\frac{1}{n} \sum_{i=1}^{n} (y_i - \boldsymbol{w}^\mathsf{T} \Phi(\boldsymbol{x}))^2 + \lambda \underbrace{|\boldsymbol{w}|}_{\sum_{j=1}^{\tilde{d}} |w_j|} \right]$$

Linear regression with one-norm regularization is referred to as *LASSO* (least absolute shrinkage and selection operator).

Sparsity

One-norm regularization leads to sparser models (i.e., models with several coefficients $w_i = 0$).

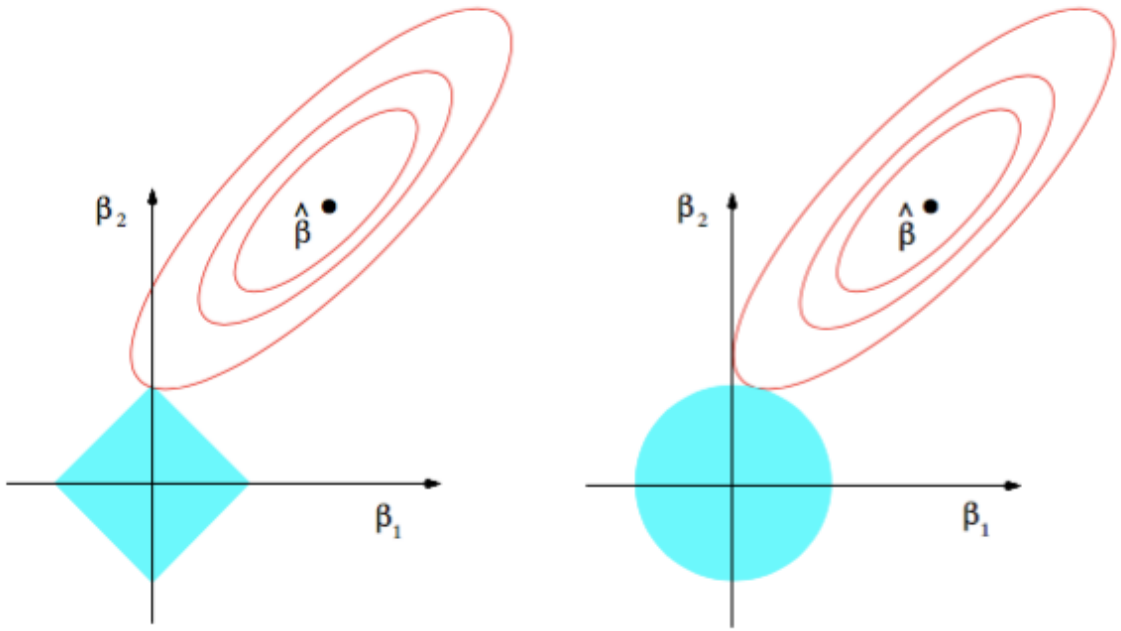


FIGURE 3.11. *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions* $|\beta_1| + |\beta_2| \leq t$ *and* $\beta_1^2 + \beta_2^2 \leq t^2$*, respectively, while the red ellipses are the contours of the least squares error function.*

Hastie, Tibshirani, & Friedman. *The Elements of Statistical Learning*. Springer, 2009

Outline

Summary

- Regression means building predictive models for real-valued target patterns.
- Linear regression is a baseline method for regression tasks.
- Too complex/flexible models can lead to overfitting.
- However, the affine linear models built by standard linear regression are not very flexible and rather tend to "underfit" the data. Therefore non-linear methods are needed.
- Non-linear transformations combined with linear regression lead to non-linear models.
- Regularization can prevent overfitting.
- 1-norm regularization leads to sparsity.