Linear Regression. Overfitting and regularization

Sriram Sankararaman

The instructor gratefully acknowledges Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Linear regression cost function (residual sum of squares)

- Input: $\boldsymbol{x} \in \mathbb{R}$
- Output: $y \in \mathbb{R}$
- Hypotheses/Model: $h_{\boldsymbol{\theta}}$, with $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \theta_0 + \theta_1 x_1 = \boldsymbol{\theta}^{\top}\boldsymbol{x}$

Residual sum of squares (RSS)

$$\underset{\theta}{\arg\min}\ J(\boldsymbol{\theta}) = \sum_n [y_n - h_{\boldsymbol{\theta}}(\boldsymbol{x}_n)]^2 = \sum_n [y_n - (\theta_0 + \theta_1 x_n)]^2$$

Outline

Why is minimizing J sensible?

$Y_1 = \theta_0 + \theta_1 X_1 + \eta_1 \rightarrow (X_1, Y_1)$

$Y_2 = \theta_0 + \theta_1 X_2 + \eta_2$ $\quad (X_2, Y_2)$

$\vdots$

$Y_N = \theta_0 + \theta_1 X_N + \eta_N$ $\quad (X_N, Y_N)$

Known (assumption!)

$\Rightarrow \ell(\theta_0, \theta_1, \sigma^2) = P(\text{Data} \mid \theta_0, \theta_1, \sigma^2)$

Probabilistic interpretation

- Noisy observation model

$$Y = \theta_0 + \theta_1 X + \eta$$

where $\eta \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian random variable

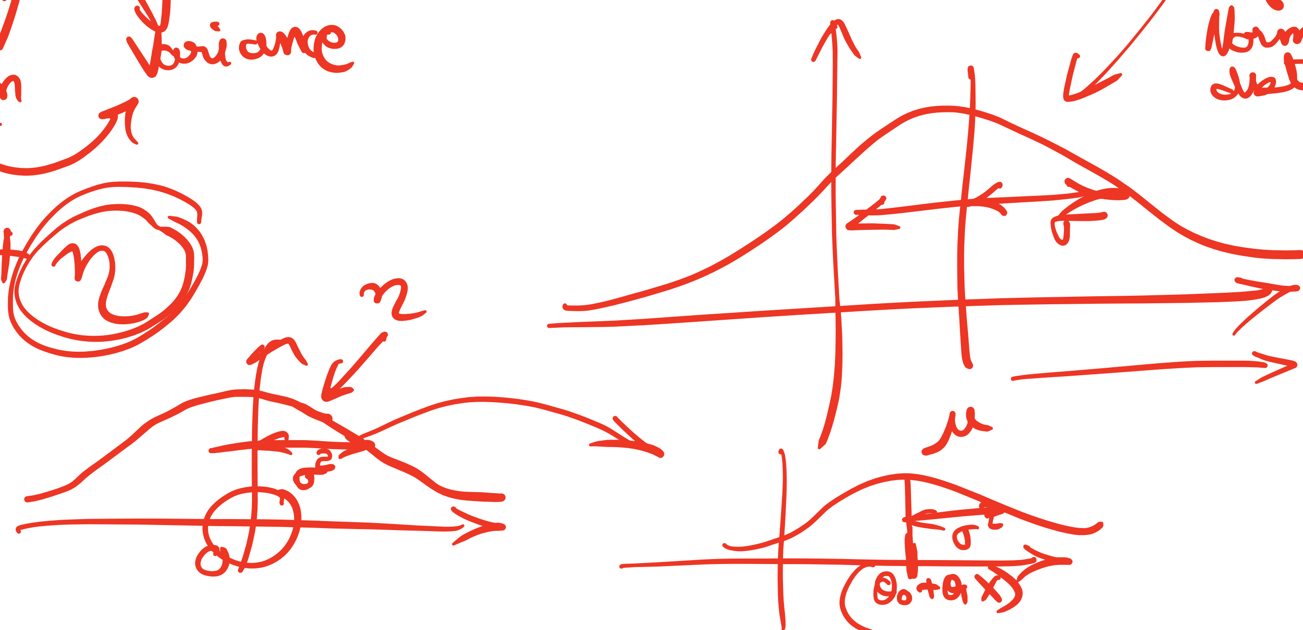
Mean

Variance

PDF of Normal distribution

$Y = \theta_0 + \theta_1 X + \eta$

$Y \sim N(\theta_0 + \theta_1 X, \sigma^2)$



Why is minimizing J sensible?

Probabilistic interpretation

- Noisy observation model

$$Y = \theta_0 + \theta_1 X + \eta$$

$x \sim \mathcal{N}(\mu, \sigma^2)$

$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$

  where $\eta \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian random variable

- Likelihood of one training sample (x_n, y_n)

$$p(y_n | x_n; \boldsymbol{\theta}) = \mathcal{N}(\theta_0 + \theta_1 x_n, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{[y_n - (\theta_0 + \theta_1 x_n)]^2}{2\sigma^2}}$$

$y_n \sim \mathcal{N}(\quad , \quad)$

Probabilistic interpretation (cont'd)

Log-likelihood of the training data $\mathcal{D}$ (assuming i.i.d)

$$\mathcal{LL}(\boldsymbol{\theta}) = \log P(\mathcal{D})$$

Independence

$$= \log \prod_{n=1}^{N} p(y_n|x_n) = \sum_n \log p(y_n|x_n)$$

$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_n-(\theta_0+\theta_1 x_n))^2}{2\sigma^2}\right)$$

$$= \sum_n \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y_n-(\theta_0+\theta_1 x_n))^2\right)\right)$$

$$= \sum_n \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + \log\left(\exp\left(-\frac{1}{2\sigma^2}(y_n-(\theta_0+\theta_1 x_n))^2\right)\right)$$

$$= \sum_n -\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(y_n-(\theta_0+\theta_1 x_n))^2$$

Probabilistic interpretation (cont'd)

Log-likelihood of the training data $\mathcal{D}$ (assuming i.i.d)

$$
\begin{aligned}
\mathcal{LL}(\boldsymbol{\theta}) &= \log P(\mathcal{D}) \\
&= \log \prod_{n=1}^{\mathsf{N}} p(y_n|x_n) = \sum_n \log p(y_n|x_n) \\
&= \sum_n \left\{ -\frac{[y_n - (\theta_0 + \theta_1 x_n)]^2}{2\sigma^2} - \log \sqrt{2\pi}\sigma \right\}
\end{aligned}
$$

Probabilistic interpretation (cont'd)

Log-likelihood of the training data $\mathcal{D}$ (assuming i.i.d)

$$
\begin{aligned}
\mathcal{LL}(\boldsymbol{\theta}) &= \log P(\mathcal{D}) \\
&= \log \prod_{n=1}^{N} p(y_n|x_n) = \sum_n \log p(y_n|x_n) \\
&= \sum_n \left\{ -\frac{[y_n - (\theta_0 + \theta_1 x_n)]^2}{2\sigma^2} - \underline{\underline{\log \sqrt{2\pi}\sigma}} \right\} \\
&= -\frac{1}{2\sigma^2} \underline{\underline{\sum_n [y_n - (\theta_0 + \theta_1 x_n)]^2}} - \underline{\underline{\frac{N}{2}\log \sigma^2}} - N \log \sqrt{2\pi}
\end{aligned}
$$

$\log \sqrt{2\pi\sigma^2}$

$= \frac{1}{2}\log(2\pi\sigma^2)$

$= \left(\frac{1}{2}\log 2\pi + \frac{1}{2}\log\sigma^2\right)$

Probabilistic interpretation (cont'd)

Log-likelihood of the training data $\mathcal{D}$ (assuming i.i.d)

$$\text{argmax}_\theta \; \mathcal{LL}(\boldsymbol{\theta}) = \log P(\mathcal{D})$$

$$= \log \prod_{n=1}^{\mathsf{N}} p(y_n|x_n) = \sum_n \log p(y_n|x_n)$$

$$= \sum_n \left\{ -\frac{[y_n - (\theta_0 + \theta_1 x_n)]^2}{2\sigma^2} - \log \sqrt{2\pi}\sigma \right\}$$

$$= -\frac{1}{2\sigma^2} \sum_n [y_n - (\theta_0 + \theta_1 x_n)]^2 - \frac{\mathsf{N}}{2} \log \sigma^2 - \mathsf{N} \log \sqrt{2\pi}$$

$$= -\frac{1}{2} \left\{ \frac{1}{\sigma^2} \sum_n [y_n - (\theta_0 + \theta_1 x_n)]^2 + \mathsf{N} \log \sigma^2 \right\} + \text{const}$$

$$\text{argmin}_\theta \; J(\theta) = \sum_n (y_n - (\theta_0 + \theta_1 x_n))^2$$

$$\text{argmin} \left\{ \frac{1}{\sigma^2} \sum_n (y_n - (\theta_0 + \theta_1 x_n))^2 + \mathsf{N} \log \sigma^2 \right\}$$

What is the relationship between minimizing J and maximizing the log-likelihood?

Maximum likelihood estimation

Estimating σ, θ_0 and θ_1 can be done in two steps

- Maximize over θ_0 and θ_1

$$\max \; \log P(\mathcal{D}) \Leftrightarrow \min \sum_n [y_n - (\theta_0 + \theta_1 x_n)]^2 \leftarrow \text{That is } J(\boldsymbol{\theta})!$$

Maximum likelihood estimation

Estimating σ, θ_0 and θ_1 can be done in two steps

- Maximize over θ_0 and θ_1

$$\max \ \log P(\mathcal{D}) \Leftrightarrow \min \sum_n [y_n - (\theta_0 + \theta_1 x_n)]^2 \leftarrow \text{That is } J(\boldsymbol{\theta})!$$

- Maximize over $s = \sigma^2$ (we could estimate σ directly)

$$\frac{\partial \log P(\mathcal{D})}{\partial s} = \cancel{-\frac{1}{2}} \left\{ -\frac{1}{s^2} \sum_n [y_n - (\theta_0 + \theta_1 x_n)]^2 + \mathrm{N}\frac{1}{s} \right\} = 0$$

$$\cancel{s^2} \times \frac{1}{\cancel{s^2}} \sum_n (y_n - (\theta_0 + \theta_1 x_n))^2 = \frac{N}{\cancel{s}} \times s^{\cancel{2}}$$

Maximum likelihood estimation

Y, X

$\theta_0 + \theta_1 X_*$

Estimating σ, θ_0 and θ_1 can be done in two steps

- Maximize over θ_0 and θ_1

$$\max \; \log P(\mathcal{D}) \Leftrightarrow \min \sum_n [y_n - (\theta_0 + \theta_1 x_n)]^2 \leftarrow \text{That is } J(\boldsymbol{\theta})!$$

- Maximize over $s = \sigma^2$ (we could estimate σ directly)

$$\frac{\partial \log P(\mathcal{D})}{\partial s} = -\frac{1}{2}\left\{ -\frac{1}{s^2} \sum_n [y_n - (\theta_0 + \theta_1 x_n)]^2 + \mathrm{N}\frac{1}{s} \right\} = 0$$

$\theta_0 + \theta_1 X + \eta$

$$\rightarrow \sigma^{*2} = s^* = \frac{1}{\mathrm{N}} \sum_n [y_n - (\theta_0 + \theta_1 x_n)]^2$$

Why does the probabilistic interpretation help us?

Gives us a template for modeling

$P(x;\theta)$

- Probabilistic model $P(x;\theta)$
  - "Story" for generating the data
- Given data x and probabilistic model $P(x;\theta)$, we can find a "good" value for θ.
  - Maximize the likelihood
- We have seen this for logistic regression, linear regression (minimizing the RSS) but this principle is very general.
- Makes clear what the assumptions are.

$y_n = \theta_0 + \theta_1 x_n + \eta$

$\eta \sim N(0, \sigma^2)$

$\sum_n (y_n - (\theta_0 + \theta_1 x_n))^2$

Independence

y_{n+1}

Why does the probabilistic interpretation help us?

Assumptions underlying linear regression

y_n are

- Independent
- Normally distrubted
- Mean is a linear function of x_n $\longrightarrow$ $\theta_0 + \theta_1 x_n$
- Constant variance σ^2

Outline

1. Probabilistic interpretation
2. Multivariate solution
   - D-dimensional inputs
   - Computational and numerical optimization
3. Nonlinear hypotheses
4. Basic ideas to overcome overfitting

Linear regression when $\boldsymbol{x}$ is D-dimensional

$J(\boldsymbol{\theta}))$ in matrix form

$$\underset{\theta}{\text{argmin}}\; J(\boldsymbol{\theta}) = \sum_n [y_n - (\theta_0 + \sum_d \theta_d x_{nd})]^2 = \sum_n [y_n - \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{x}_n]^2$$

where we have redefined some variables (by augmenting)

$$\boldsymbol{x} \leftarrow [1 \; x_1 \; x_2 \; \ldots \; x_{\mathsf{D}}]^{\mathrm{T}}, \quad \boldsymbol{\theta} \leftarrow [\theta_0 \; \theta_1 \; \theta_2 \; \ldots \; \theta_{\mathsf{D}}]^{\mathrm{T}}$$

$J(\boldsymbol{\theta})$ in new notations

$x_1 = \begin{pmatrix} x_{1,0} \\ x_{1,1} \\ \vdots \\ x_{1,D} \end{pmatrix}$

Design matrix and target vector

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1^{\mathrm{T}} \\ \boldsymbol{x}_2^{\mathrm{T}} \\ \vdots \\ \boldsymbol{x}_{\mathsf{N}}^{\mathrm{T}} \end{pmatrix} \in \mathbb{R}^{\mathsf{N}\times(D+1)}, \quad \underbrace{\boldsymbol{y}}_{(N\times 1)} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{\mathsf{N}} \end{pmatrix}$$

Vector of predictions for a given $\boldsymbol{\theta}$

$$\boldsymbol{X}\boldsymbol{\theta} = \begin{pmatrix} \boldsymbol{x}_1^{\mathrm{T}} \\ \boldsymbol{x}_2^{\mathrm{T}} \\ \vdots \\ \boldsymbol{x}_{\mathsf{N}}^{\mathrm{T}} \end{pmatrix} \boldsymbol{\theta} \quad = \begin{pmatrix} \boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{\theta} \\ \boldsymbol{x}_2^{\mathrm{T}}\boldsymbol{\theta} \\ \vdots \\ \boldsymbol{x}_{\mathsf{N}}^{\mathrm{T}}\boldsymbol{\theta} \end{pmatrix}$$

$\begin{pmatrix} x_1^T\theta \\ x_2^T\theta \\ x_N^T\theta \end{pmatrix} = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{pmatrix}\theta = X\theta$

$J(\boldsymbol{\theta})$ in new notations

Vector of errors for a given θ

$$\boldsymbol{y} - \boldsymbol{X\theta} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{\mathsf{N}} \end{pmatrix} - \begin{pmatrix} \boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{\theta} \\ \boldsymbol{x}_2^{\mathrm{T}}\boldsymbol{\theta} \\ \vdots \\ \boldsymbol{x}_{\mathsf{N}}^{\mathrm{T}}\boldsymbol{\theta} \end{pmatrix} = \begin{pmatrix} y_1 - \boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{\theta} \\ y_2 - \boldsymbol{x}_2^{\mathrm{T}}\boldsymbol{\theta} \\ \vdots \\ y_{\mathsf{N}} - \boldsymbol{x}_{\mathsf{N}}^{\mathrm{T}}\boldsymbol{\theta} \end{pmatrix}$$

$(N\times 1)$

error on datapt 1

on datapt 2

Expression for $J(\boldsymbol{\theta})$

$$||\boldsymbol{y} - \boldsymbol{X\theta}||_2^2$$

$$= (\boldsymbol{y} - \boldsymbol{X\theta})^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{X\theta})$$

$$= \begin{pmatrix} y_1 - \boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{\theta} & \cdots & y_{\mathsf{N}} - \boldsymbol{x}_{\mathsf{N}}^{\mathrm{T}}\boldsymbol{\theta} \end{pmatrix} \begin{pmatrix} y_1 - \boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{\theta} \\ \vdots \\ y_{\mathsf{N}} - \boldsymbol{x}_{\mathsf{N}}^{\mathrm{T}}\boldsymbol{\theta} \end{pmatrix}$$

$$= \sum_n [y_n - \boldsymbol{x}_n^{\mathrm{T}}\boldsymbol{\theta}]^2 \leftarrow \text{That is } J(\boldsymbol{\theta})$$

$$||x||_2^2 = (x^{\mathrm{T}}x) = \sum_{d=1}^{D} x_d^2$$

$$\nabla J(\theta) = 0$$

$J(\theta)$ in new notations

$f(x) = b^T x = b_1 x_1 + b_2 x_2 + \dots + b_D x_D$

$x \in \mathbb{R}^D$

$\nabla f(x) = b$

Compact expression

$\nabla_\theta J(\theta)$

$$\begin{aligned}
J(\boldsymbol{\theta}) &= \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}\|_2^2 \\
&= (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) \\
&= (\boldsymbol{y}^{\mathrm{T}} - \{\boldsymbol{X}\boldsymbol{\theta}\}^{\mathrm{T}})(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) \\
&= (\boldsymbol{y}^{\mathrm{T}} - \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}})(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) \\
&= (\boldsymbol{y}^{\mathrm{T}})(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) - \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) \\
&= \boldsymbol{y}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{y}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y} + \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\theta} \\
&= \text{constant} - 2\boldsymbol{y}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\theta}
\end{aligned}$$

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_D} \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_D \end{pmatrix} = b$$

$y^T X\theta = (X\theta)^T y$

$-2y^T X\theta$

$\nabla(-2(X^T y)^T \theta) = -2X^T y$

Solution in matrix form

$$\nabla(\theta^T X^T X \theta) = 2(X^T X)\theta$$

Compact expression

$$J(\boldsymbol{\theta}) = ||\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}||_2^2 = \left\{\underline{\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\theta}} - 2\left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}\right)^{\mathrm{T}}\boldsymbol{\theta}\right\} + \text{constant}$$

Gradients of Linear and Quadratic Functions

- $\nabla \boldsymbol{b}^\top \boldsymbol{x} = \boldsymbol{b}$
- $\nabla \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} = 2\boldsymbol{A}\boldsymbol{x}$ (symmetric $\boldsymbol{A}$)

$$\nabla f(x) = 2Ax$$

$$f(x) = x^T A x$$

$x \in \mathbb{R}^D$

$D \times D$

$$\frac{d}{dx}(Ax^2) = 2Ax$$

Solution in matrix form

Compact expression

$$J(\boldsymbol{\theta}) = ||\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}||_2^2 = \left\{\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\theta} - 2\left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}\right)^{\mathrm{T}}\boldsymbol{\theta}\right\} + \text{constant}$$

Gradients of Linear and Quadratic Functions

- $\nabla \boldsymbol{b}^{\top}\boldsymbol{x} = \boldsymbol{b}$
- $\nabla \boldsymbol{x}^{\top}\boldsymbol{A}\boldsymbol{x} = 2\boldsymbol{A}\boldsymbol{x}$ (symmetric $\boldsymbol{A}$)

Normal equations

$$\nabla J(\boldsymbol{\theta}) = \cancel{2}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\theta} - \cancel{2}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y} = 0$$

$$(X^T X)\theta = X^T y$$

$$A x = b$$

Solution in matrix form

$(X^T X)^{-1} \stackrel{?}{=} X^{-1} X^{-T}$

Compact expression

X $N \times (D+1)$ $\quad$ $X^T X$ $(D+1) \times (D+1)$

$$J(\boldsymbol{\theta}) = ||\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}||_2^2 = \left\{\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\theta} - 2\left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}\right)^{\mathrm{T}}\boldsymbol{\theta}\right\} + \text{constant}$$

Gradients of Linear and Quadratic Functions

- $\nabla \boldsymbol{b}^\top \boldsymbol{x} = \boldsymbol{b}$
- $\nabla \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} = 2\boldsymbol{A}\boldsymbol{x}$ (symmetric $\boldsymbol{A}$)

Normal equations

$(X^T X)^{-1}$ $\quad$ $(X^T X)^{-1}$

$$\nabla J(\boldsymbol{\theta}) = 2\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\theta} - 2\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y} = 0$$

$(X^T X)\theta = X^T y$

This leads to the linear regression solution

θ $\quad$ $I\theta$

$$\widehat{\boldsymbol{\theta}} = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

$= (X^T X)^{-1} X^T y$

Computational complexity

Bottleneck of computing the solution?

X y

$N \times D$ $N \times 1$

$$\widehat{\boldsymbol{\theta}} = \underline{\left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right)}^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

$O(D^2N) \Rightarrow X^T X$ $X^T y$

$(X^T X)$ $(D \times D)$

$\underline{(X^T X)^{-1}}$ $O(D^3)$

$(D \times 1)$ $O(ND)$

Computational complexity

Bottleneck of computing the solution?

$$\widehat{\boldsymbol{\theta}} = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

Matrix multiply of $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} \in \mathbb{R}^{(\mathsf{D}+1)\times(\mathsf{D}+1)}$
Inverting the matrix $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$

How many operations do we need?

Computational complexity

Intercept $\rightarrow \theta_0 + \theta_1 x_{n1} + \theta_2 x_{n2} + \ldots + \theta_D x_{nD}$

Bottleneck of computing the solution?

$$\widehat{\boldsymbol{\theta}} = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

$x_n = (1, x_{n1}, x_{n2}, \ldots, x_{nD})$

$\theta = (\theta_0, \theta_1, \ldots, \theta_D)$

$\theta^T x_n$

Matrix multiply of $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} \in \mathbb{R}^{(\mathsf{D}+1)\times(\mathsf{D}+1)}$

Inverting the matrix $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$

How many operations do we need?

- $O(\mathsf{ND}^2)$ for matrix multiplication
- $O(\mathsf{D}^3)$ for matrix inversion
- Impractical for very large D or N

$A \; B = I$

DxD DxD

$= \begin{pmatrix} 1 & & 0 \\ 0 & 1 & \\ & & 1 \end{pmatrix}$

$(D \times D) \times D$

$A b_1 = I_1$

$A b_2 = I_2 \quad \ldots \quad A b_D = I_D$

Alternative method: an example of using numerical optimization

(Batch) Gradient descent

Algorithm 1 Gradient Descent (J)

$t \leftarrow 0$

Initialize $\theta^{(0)}$

repeat

$\nabla J(\boldsymbol{\theta}^{(t)}) = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\theta}^{(t)} - \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y} = \sum_n (\boldsymbol{x}_n^{\mathrm{T}}\boldsymbol{\theta}^{(t)} - y_n)\boldsymbol{x}_n$ $\Rightarrow O(ND)$

$\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta \nabla J(\boldsymbol{\theta}^{(t)})$

$t \leftarrow t + 1$

until convergence

Return final value of $\boldsymbol{\theta}$

$O(D)$

$O(D)$

What is the complexity of each iteration?

Why would this work?

If gradient descent converges, it will converge to the same solution as using matrix inversion.

This is because $J(\boldsymbol{\theta})$ is a convex function in its parameters $\boldsymbol{\theta}$

Hessian matrix of $J(\theta)$ is positive-semi-definite

Stochastic gradient descent

Update parameters using one example at a time $O(ND)$

Algorithm 2 Stochastic Gradient Descent (J)

$t \leftarrow 0$
Initialize $\boldsymbol{\theta}^{(0)}$
repeat
  Randomly choose a training a sample $\boldsymbol{x}_t$
  Compute its contribution to the gradient $\boldsymbol{g}_t = (\boldsymbol{x}_t^{\mathrm{T}}\boldsymbol{\theta}^{(t)} - y_t)\boldsymbol{x}_t$
  $O(D)$
  $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta \boldsymbol{g}_t$
  $t \leftarrow t+1$
until convergence
Return final value of $\boldsymbol{\theta}$

How does the complexity per iteration of stochastic gradient descent (SGD) compare with gradient descent (GD)?

How does the complexity per iteration of stochastic gradient descent (SGD) compare with gradient descent (GD)?

- $O(\text{ND})$ for GD versus $O(\text{D})$ for SGD

Mini-summary

- Batch gradient descent computes the exact gradient. N
- Stochastic gradient descent approximates the gradient with a single data point; 1
- Mini-batch variant: trade-off between accuracy of estimating gradient and computational cost
- Similar ideas extend to other ML optimization problems.
  - For large-scale problems, stochastic gradient descent often works well.

What if $X^{\mathrm{T}}X$ is not invertible

$$\theta = (X^T X)^{-1} X^T y$$

Can you think of any reasons why that could happen?

$$(X^T X)\,\underline{\theta} = X^T y$$

1 , 1

θ 100, -100

10, -10

What if $X^{\mathrm{T}}X$ is not invertible

Can you think of any reasons why that could happen?

Answer 1: N < D. Intuitively, not enough data to estimate all the parameters.

What if $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ is not invertible

Can you think of any reasons why that could happen?

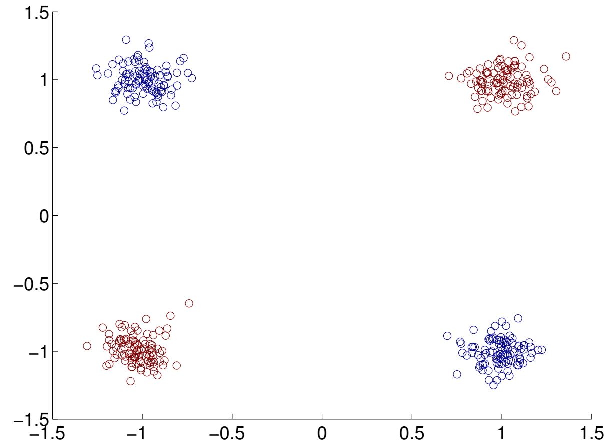
Answer 1: $\mathrm{N} < \mathrm{D}$. Intuitively, not enough data to estimate all the parameters.

Answer 2: $\boldsymbol{X}$ columns are not linearly independent. Intuitively, there are two features that are perfectly correlated. In this case, solution is not unique.

Outline

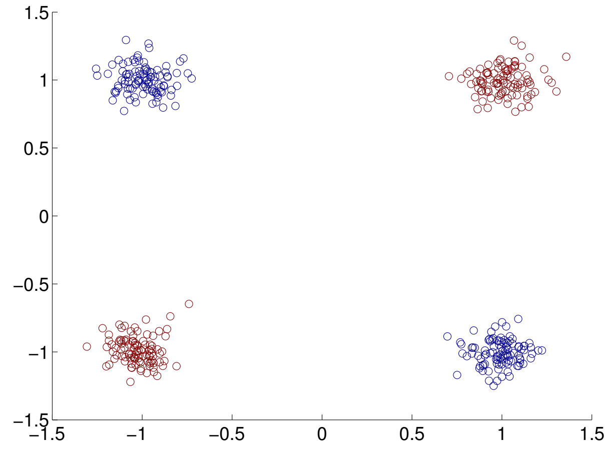
What if data is not linearly separable or fits to a line

Example of nonlinear classification

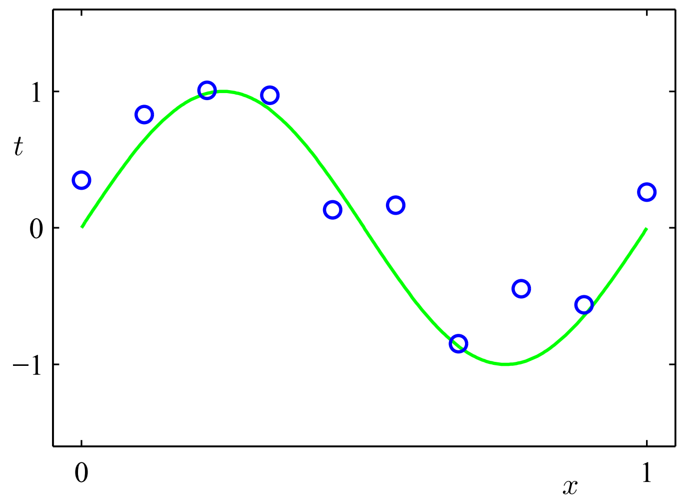


What if data is not linearly separable or fits to a line

Example of nonlinear classification



Example of nonlinear regression



Nonlinear basis for classification

Transform the input/feature

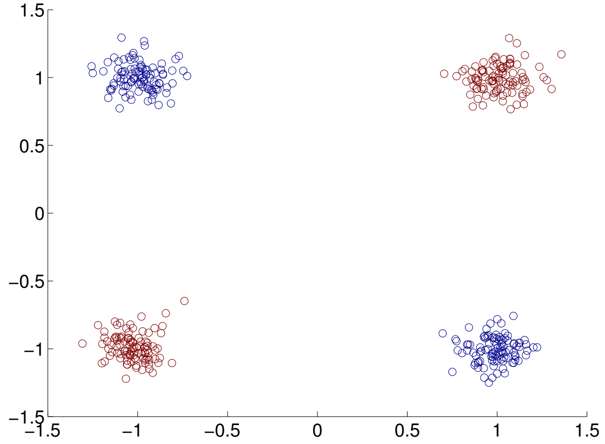
$$\phi(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^2 \rightarrow z = x_1 \cdot x_2$$

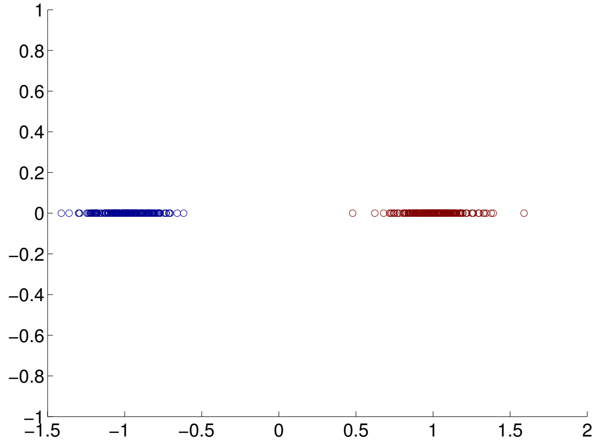
Nonlinear basis for classification

Transform the input/feature

$$\phi(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^2 \rightarrow z = x_1 \cdot x_2$$

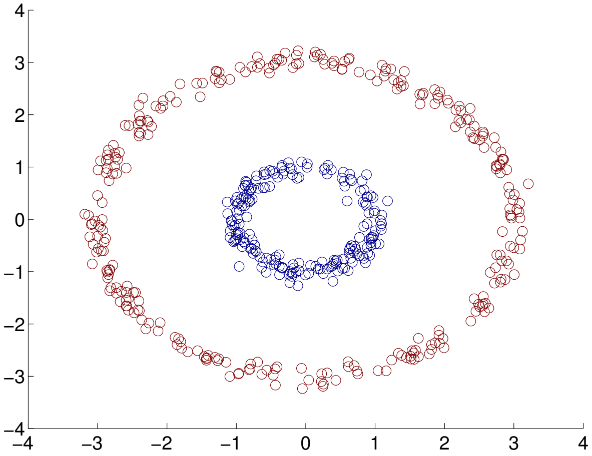
Transformed training data: linearly separable!



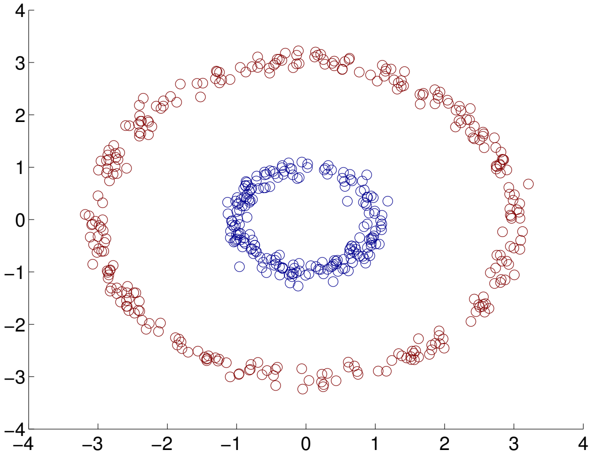


Another example

How to transform the input/feature?



Another example

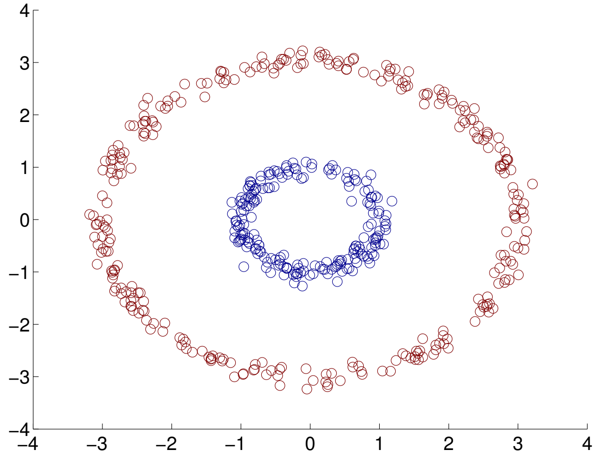


How to transform the input/feature?

$$\boldsymbol{\phi}(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^2 \rightarrow \boldsymbol{z} = \begin{bmatrix} x_1^2 \\ x_1 \cdot x_2 \\ x_2^2 \end{bmatrix} \in \mathbb{R}^3$$

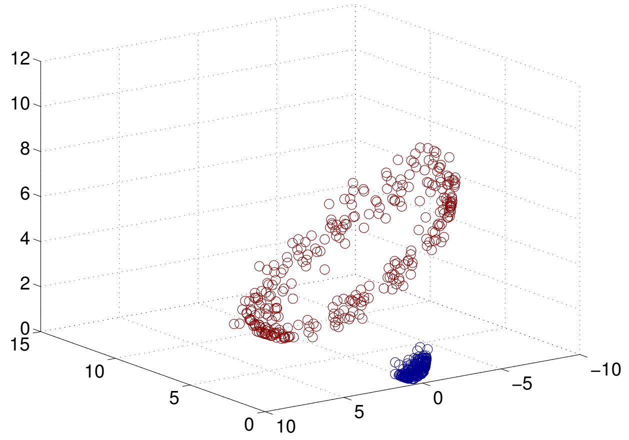
Another example

How to transform the input/feature?



$$\boldsymbol{\phi}(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^2 \rightarrow \boldsymbol{z} = \begin{bmatrix} x_1^2 \\ x_1 \cdot x_2 \\ x_2^2 \end{bmatrix} \in \mathbb{R}^3$$

Transformed training data: linearly separable



General nonlinear basis functions

We can use a nonlinear mapping

$$\boldsymbol{\phi}(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^D \rightarrow \boldsymbol{z} \in \mathbb{R}^M$$

where M is the dimensionality of the new feature/input $\boldsymbol{z}$ (or $\boldsymbol{\phi}(\boldsymbol{x})$). Note that M could be either greater than D or less than or the same.

General nonlinear basis functions

We can use a nonlinear mapping

$$\boldsymbol{\phi}(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^D \rightarrow \boldsymbol{z} \in \mathbb{R}^M$$

where M is the dimensionality of the new feature/input $\boldsymbol{z}$ (or $\boldsymbol{\phi}(\boldsymbol{x})$). Note that M could be either greater than D or less than or the same.

With the new features, we can apply our learning techniques to minimize our errors on the transformed training data

- linear methods: prediction is based on $\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})$
- other methods: nearest neighbors, decision trees, etc

Regression with nonlinear basis

Residual sum squares

$$\sum_n [\boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) - y_n]^2$$

where $\boldsymbol{\theta} \in \mathbb{R}^M$, the same dimensionality as the transformed features $\boldsymbol{\phi}(\boldsymbol{x})$.

Regression with nonlinear basis

Residual sum squares

$$\sum_n [\boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) - y_n]^2$$

where $\boldsymbol{\theta} \in \mathbb{R}^M$, the same dimensionality as the transformed features $\boldsymbol{\phi}(\boldsymbol{x})$.

The linear regression solution can be formulated with the new design matrix

$$\boldsymbol{\Phi} = \begin{pmatrix} \boldsymbol{\phi}(\boldsymbol{x}_1)^{\mathrm{T}} \\ \boldsymbol{\phi}(\boldsymbol{x}_2)^{\mathrm{T}} \\ \vdots \\ \boldsymbol{\phi}(\boldsymbol{x}_N)^{\mathrm{T}} \end{pmatrix} \in \mathbb{R}^{N \times M}, \quad \widehat{\boldsymbol{\theta}} = \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{y}$$

Example with regression

Polynomial basis functions

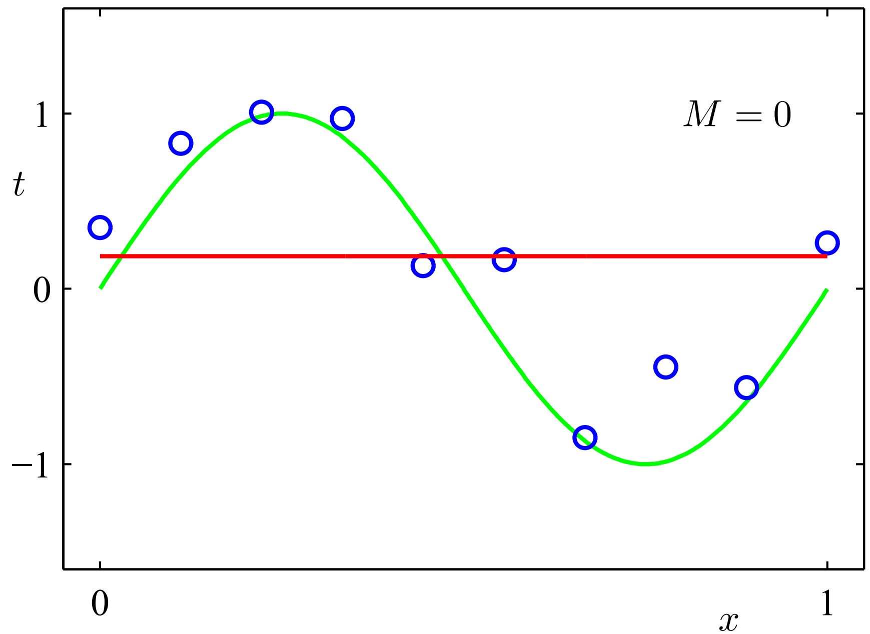
$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^M \end{bmatrix} \Rightarrow h(x) = \theta_0 + \sum_{m=1}^{M} \theta_m x^m$$

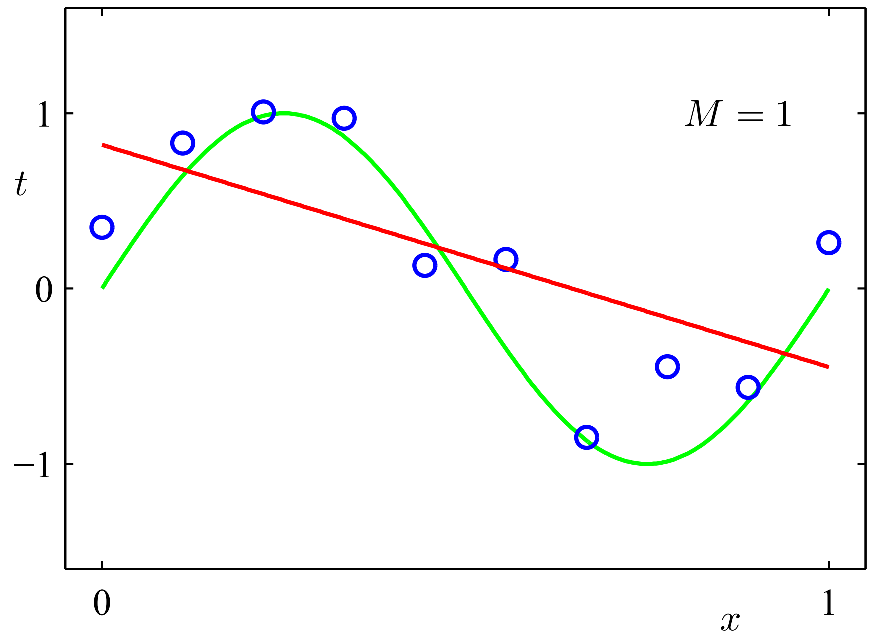
Example with regression

Polynomial basis functions

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^M \end{bmatrix} \Rightarrow h(x) = \theta_0 + \sum_{m=1}^{M} \theta_m x^m$$

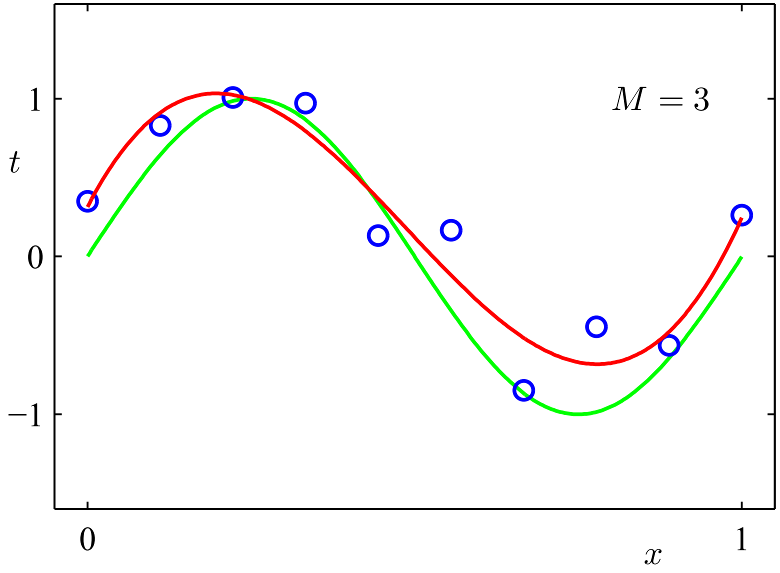
Fitting samples from a sine function: *underrfitting* as $h(x)$ is too simple





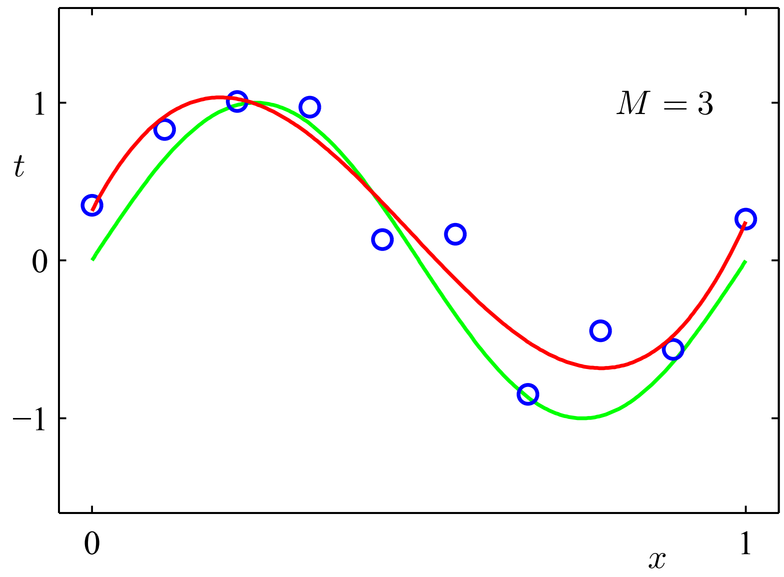
Adding high-order terms

M=3

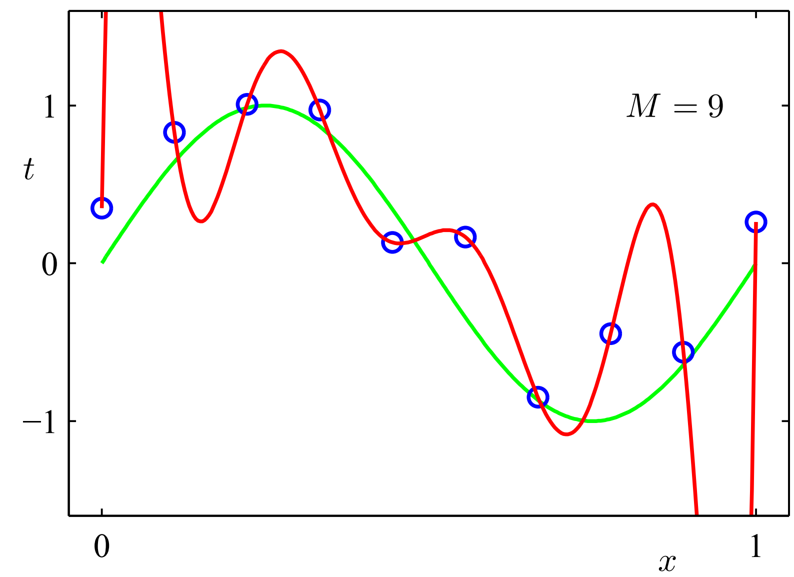


Adding high-order terms

M=3



M=9: *overfitting*



More complex features lead to better results on the training data, but potentially worse results on new data, e.g., test data!

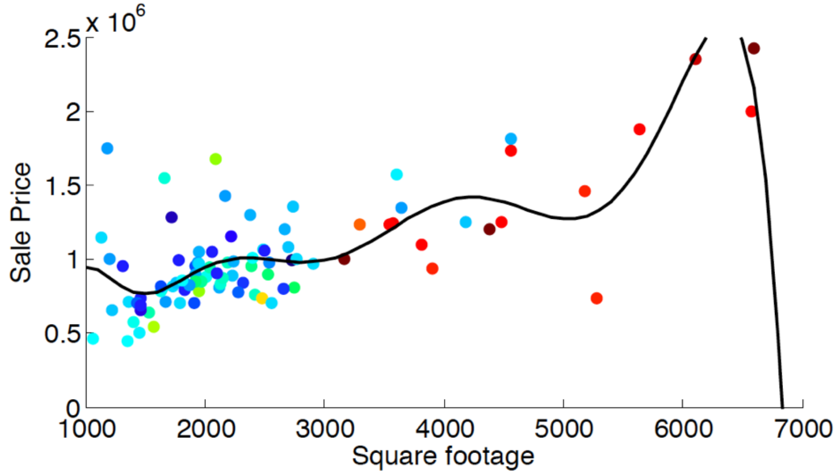
Overfitting

Parameters for higher-order polynomials are very large

	$M=0$	$M=1$	$M=3$	$M=9$
θ_0	0.19	0.82	0.31	0.35
θ_1		-1.27	7.99	232.37
θ_2			-25.43	-5321.83
θ_3			17.37	48568.31
θ_4				-231639.30
θ_5				640042.26
θ_6				-1061800.52
θ_7				1042400.18
θ_8				-557682.99
θ_9				125201.43

Overfitting can be quite disastrous

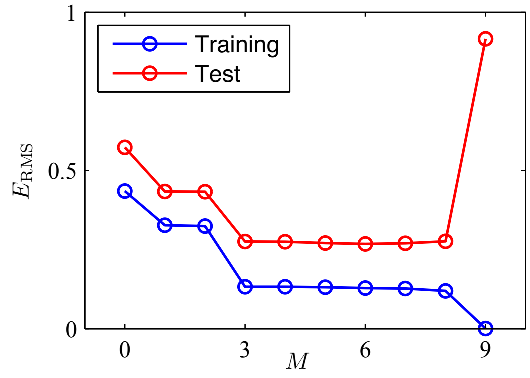
Fitting the housing price data with $M = 3$



Note that the price would goes to zero (or negative) if you buy bigger ones!
This is called poor generalization/overfitting.

Detecting overfitting

Plot model complexity versus objective function



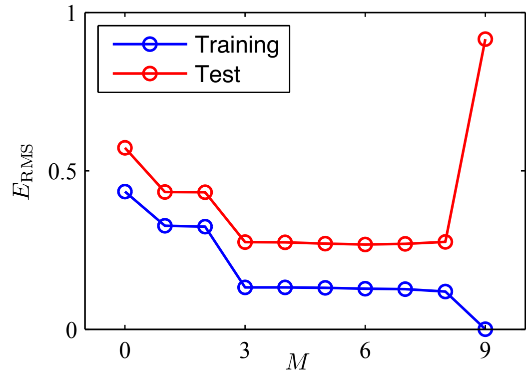
As a model increases in complexity, performance on training data keeps improving while performance on test data may first improve but eventually deteriorate.

- Horizontal axis: *measure of model complexity*; in this example complexity defined by order of the polynomial basis functions.

Detecting overfitting

Plot model complexity versus objective function

As a model increases in complexity, performance on training data keeps improving while performance on test data may first improve but eventually deteriorate.



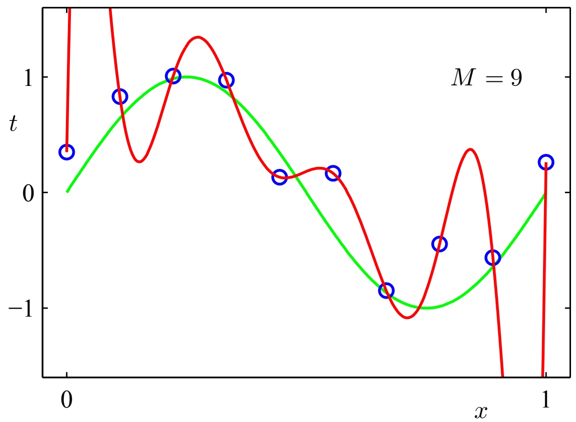
- Horizontal axis: *measure of model complexity*; in this example complexity defined by order of the polynomial basis functions.
- Vertical axis:
  1. For regression, residual sum of squares or residual mean squared (squared root of RSS)
  2. For classification, classification error rate.

Outline

1. Probabilistic interpretation
2. Multivariate solution
3. Nonlinear hypotheses
4. Basic ideas to overcome overfitting
   - Use more training data
   - Regularization methods
   - Regularized classification

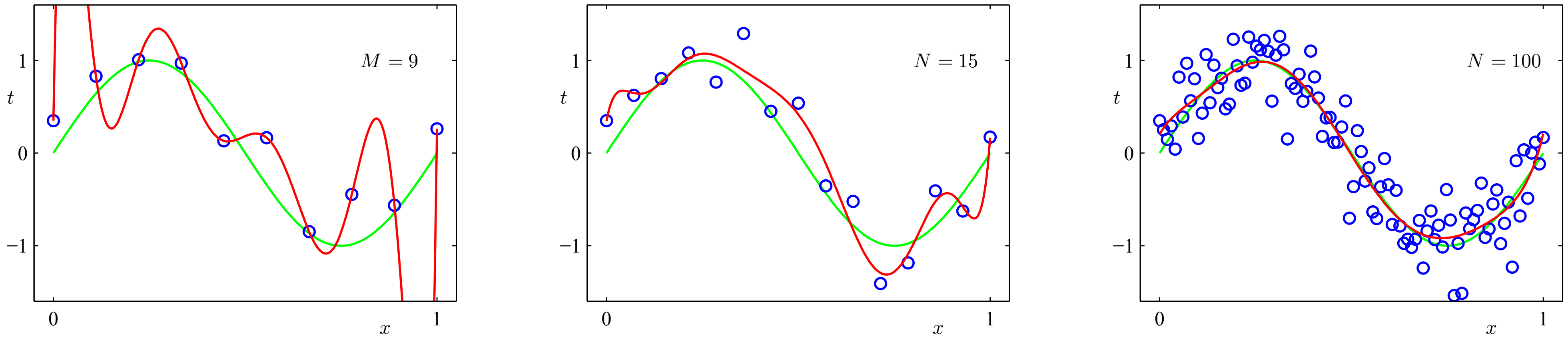
Use more training data to prevent over fitting

The more, the merrier



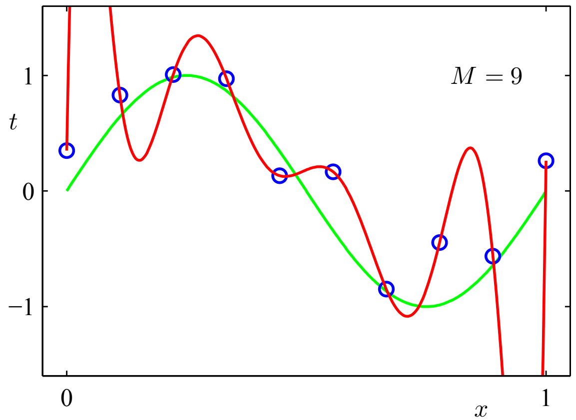
Use more training data to prevent over fitting

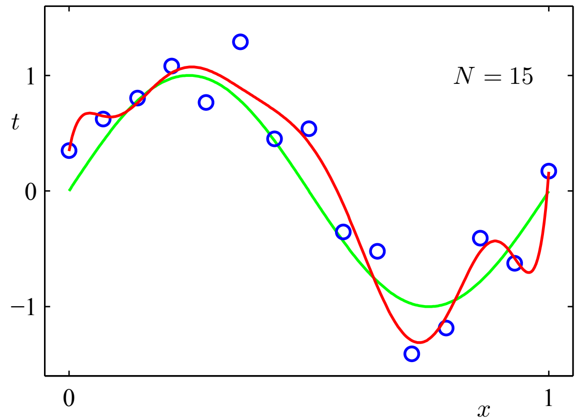
The more, the merrier

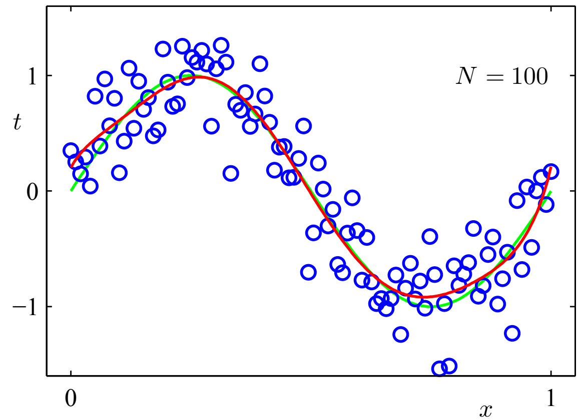


Use more training data to prevent over fitting

The more, the merrier







What if we do not have a lot of data?

Regularization methods

Intuition: For a linear model for regression

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$$

we can try to identify 'simpler' models. But what does it mean for a model to *be simple*?

Regularization methods

Intuition: For a linear model for regression

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$$

we can try to identify 'simpler' models. But what does it mean for a model to *be simple*?

Assumption (inductive bias)
A simpler model is one where most of the weights are zero.

A simpler model is one with smaller weights.

Why are smaller weights associated with simpler models?

Simpler functions are smoother, *i.e.*, nearby values of $\boldsymbol{x}$ have similar outputs $\hat{y}$.

Two values $\boldsymbol{x}$ and $\boldsymbol{x}'$ that differ in the first component by a small value ϵ.

Their predictions $\hat{y}$ and $\hat{y'}$ differ by ϵw_1.

Smaller w_1 (closer to zero), more similar are the predictions.

Regularized linear regression

A new cost function or error function to minimize

$$J(\boldsymbol{w}, b) = \sum_n (y_n - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n - b)^2 + \lambda \|\boldsymbol{w}\|_2^2$$

where $\lambda > 0$. This extra term $\|\boldsymbol{w}\|_2^2$ is called regularization/regularizer and controls the model complexity.

Regularized linear regression

A new cost function or error function to minimize

$$J(\boldsymbol{w}, b) = \sum_n (y_n - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n - b)^2 + \lambda \|\boldsymbol{w}\|_2^2$$

where $\lambda > 0$. This extra term $\|\boldsymbol{w}\|_2^2$ is called regularization/regularizer and controls the model complexity.

Intuitions

Regularized linear regression

A new cost function or error function to minimize

$$J(\boldsymbol{w}, b) = \sum_n (y_n - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n - b)^2 + \lambda \|\boldsymbol{w}\|_2^2$$

where $\lambda > 0$. This extra term $\|\boldsymbol{w}\|_2^2$ is called regularization/regularizer and controls the model complexity.

Intuitions

- If $\lambda \to +\infty$, then

$$\widehat{\boldsymbol{w}} \to \boldsymbol{0}$$

Regularized linear regression

A new cost function or error function to minimize

$$J(\boldsymbol{w}, b) = \sum_n (y_n - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n - b)^2 + \lambda \|\boldsymbol{w}\|_2^2$$

where $\lambda > 0$. This extra term $\|\boldsymbol{w}\|_2^2$ is called regularization/regularizer and controls the model complexity.

Intuitions

- If $\lambda \to +\infty$, then

$$\widehat{\boldsymbol{w}} \to \mathbf{0}$$

- If $\lambda \to 0$, then we trust our data more. Numerically,

$$\widehat{\boldsymbol{w}} \to \arg\min \sum_n (\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n + b - y_n)^2$$

Closed-form solution

For regularized linear regression: the solution changes very little (in form) from the OLS (Ordinary Least Squares) solution

$$\arg\min \sum_n (y_n - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n - b)^2 + \lambda\|\boldsymbol{w}\|_2^2 \Rightarrow \widehat{\boldsymbol{w}} = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

and reduces to the OLS solution when $\lambda = 0$, as expected.

Closed-form solution

For regularized linear regression: the solution changes very little (in form) from the OLS (Ordinary Least Squares) solution

$$\arg\min \sum_n (y_n - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n - b)^2 + \lambda \|\boldsymbol{w}\|_2^2 \Rightarrow \widehat{\boldsymbol{w}} = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

and reduces to the OLS solution when $\lambda = 0$, as expected.

If we have to use numerical procedure, the gradient would change nominally too,

$$\nabla J(\boldsymbol{w}) = 2(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y} + \lambda \boldsymbol{w})$$

As long as $\lambda \geq 0$, the optimization is convex.

Example: fitting data with polynomials

Our regression model

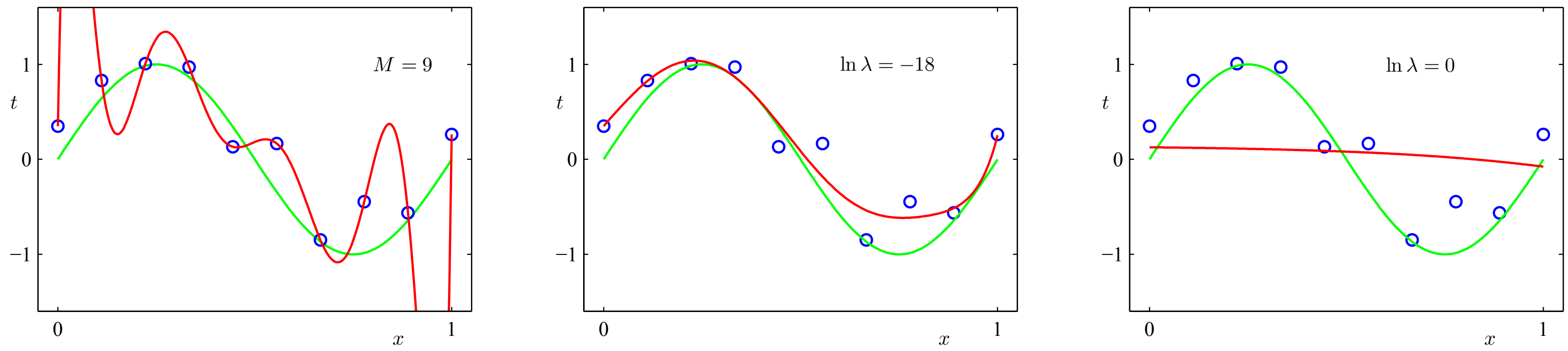
$$y = \sum_{m=1}^{M} w_m x^m$$

Regularization would discourage large parameter values as we saw with the OLS solution, thus potentially preventing overfitting.

	$M = 0$	$M = 1$	$M = 3$	$M = 9$
w_0	0.19	0.82	0.31	0.35
w_1		-1.27	7.99	232.37
w_2			-25.43	-5321.83
w_3			17.37	48568.31
w_4				-231639.30
w_5				640042.26
w_6				-1061800.52
w_7				1042400.18
w_8				-557682.99
w_9				125201.43

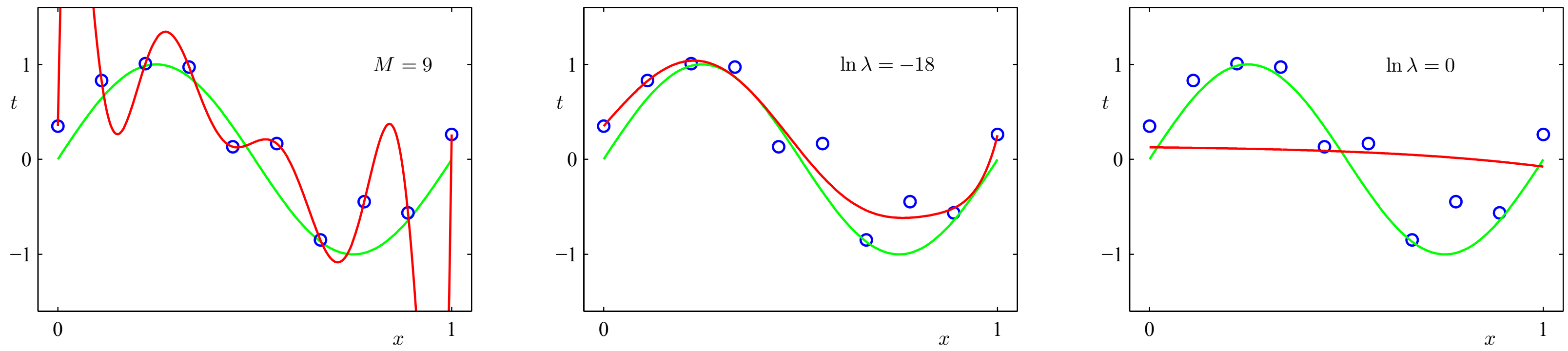
Overfitting in terms of λ

Overfitting is reduced from complex model to simpler one with the help of increasing regularizer

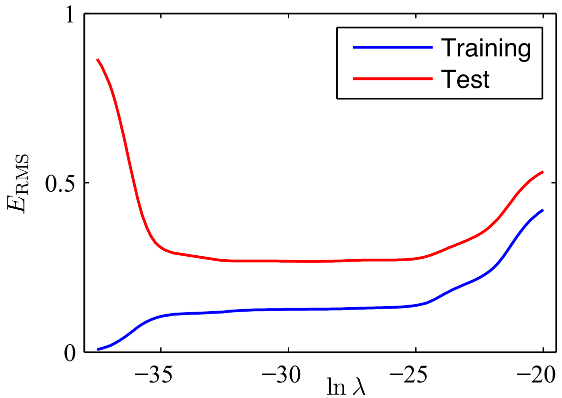


Overfitting in terms of λ

Overfitting is reduced from complex model to simpler one with the help of increasing regularizer



λ vs. residual error shows the difference of the model performance on training and testing dataset



The effect of λ

Large λ attenuates parameters towards 0

	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
w_0	0.35	0.35	0.13
w_1	232.37	4.74	-0.05
w_2	-5321.83	-0.77	-0.06
w_3	48568.31	-31.97	-0.06
w_4	-231639.30	-3.89	-0.03
w_5	640042.26	55.28	-0.02
w_6	-1061800.52	41.32	-0.01
w_7	1042400.18	-45.95	-0.00
w_8	-557682.99	-91.53	0.00
w_9	125201.43	72.68	0.01

l_2 regularized logistic regression

Adding regularizer to the cost function for logistic regression

$$J(\boldsymbol{w}, b) = -\sum_n \{y_n \log h_{\boldsymbol{w},b}(\boldsymbol{x}_n) + (1 - y_n) \log[1 - h_{\boldsymbol{w},b}(\boldsymbol{x}_n)]\} + \underbrace{\lambda \|\boldsymbol{w}\|_2^2}_{\text{regularization}}$$

$$h_{\boldsymbol{w},b}(\boldsymbol{x}) = \sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b)$$

l_2 regularized logistic regression

Adding regularizer to the cost function for logistic regression

$$J(\boldsymbol{w}, b) = -\sum_n \{y_n \log h_{\boldsymbol{w},b}(\boldsymbol{x}_n) + (1 - y_n) \log[1 - h_{\boldsymbol{w},b}(\boldsymbol{x}_n)]\} + \underbrace{\lambda \|\boldsymbol{w}\|_2^2}_{\text{regularization}}$$

$$h_{\boldsymbol{w},b}(\boldsymbol{x}) = \sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b)$$

Numerical optimization

- Objective functions remains convex as long as $\lambda \geq 0$.
- Gradients and Hessians are changed marginally and can be easily derived.

How to choose the right amount of regularization?

Can we tune λ on the training dataset?

How to choose the right amount of regularization?

Can we tune λ on the training dataset?
No: as this will set λ to zero, i.e., without regularization, defeating our intention to use it to control model complexity and to gain better generalization.

λ **is thus a hyperparmeter**. To tune it,

- We can use a development/validation dataset independent of training and testing dataset.

The procedure is similar to choosing K in the nearest neighbor classifiers.

How to choose the right amount of regularization?

Can we tune λ on the training dataset?

No: as this will set λ to zero, i.e., without regularization, defeating our intention to use it to control model complexity and to gain better generalization.

λ **is thus a hyperparmeter**. To tune it,

- We can use a development/validation dataset independent of training and testing dataset.

The procedure is similar to choosing K in the nearest neighbor classifiers.

For different λ, we get $\widehat{\boldsymbol{w}}$ and evaluate the model on the development/validation dataset.

We then plot the curve λ versus prediction error (accuracy, classification error) and find the place that the performance on the validation is the best.

Summary

- Hypotheses/models: linear functions of features.
- Objective: choose a function (*i.e.*, parameters for the function) that minimize a cost function.
  - Cost function measures loss or error of the predictions made by a model/hypothesis on training set.
  - Cost function depends on the learning problem.
- Probabilistic interpretation
  - Minimizing cost function equivalent to maximizing likelihood.
  - Allows us to understand assumptions and to generalize our models.
- How do we minimize the cost function ?
  - Numerical methods: gradient and stochastic gradient descent.
  - Sometimes analytical solutions are available.
  - Computational considerations influence the choice.
- Can allow non-linear models/hypotheses by transforming features using non-linear functions.

Summary

- Regularization: prefering a simpler model by penalizing large weights or many non-zero weights.
- Helpful to reduce overfitting and to prevent numerical instability.
- Can be added to any linear model.
- Only a minor modification to the unregularized algorithms.
- New hyperparameter λ needs to be chosen using cross-validation.