Linear Regression (continued)

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.

Announcements

- Mid-term on Feb 10 (Monday).
  - In-class
  - Closed notes, closed book.
  - Will cover material from lectures and problem sets.
  - All material till and including lecture on Wednesday.
  - Please come to office hours with any questions.
  - Encouraged to go to discussion section on Friday. TAs will discuss some practice problems.

Outline

Linear regression (ordinary least squares)

Setup

- Input: $\boldsymbol{x} \in \mathbb{R}^{\mathsf{D}}$ (covariates, predictors, features, etc)
- Output: $y \in \mathbb{R}$ (responses, targets, outcomes, outputs, etc)
- Hypotheses/Model:

  $h_{\boldsymbol{w},b}$, with $h_{\boldsymbol{w},b}(\boldsymbol{x}) = b + \sum_d w_d x_d = b + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}$

  - $\boldsymbol{w} = [w_1 \; w_2 \; \cdots \; w_{\mathsf{D}}]^{\mathrm{T}}$: weights

    b is called the bias or offset or intercept term.

    $\boldsymbol{\theta} = [b \; w_1 \; w_2 \; \cdots \; w_{\mathsf{D}}]^{\mathrm{T}}$
- Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, \mathsf{N}\}$

How do we learn parameters?

Minimize prediction error on training data

Minimize the sum of squared errors (also called residual sum of squares RSS):

cost function for linear regression.

$$J(\boldsymbol{\theta}) = \sum_n [y_n - h_{\boldsymbol{\theta}}(\boldsymbol{x}_n)]^2$$

How do we minimize the cost function (residual sum of squares)?

Numerical optimization

Gradient descent

Analytical solution

Can compute minimum in closed form for linear regression!

A simple case: $\boldsymbol{x}$ is just one-dimensional (D=1)

Residual sum of squares (RSS)

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

Identify stationary points by taking derivative with respect to parameters and setting to zero

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_0} = 0 \Rightarrow -2 \sum_n [y_n - (\theta_0 + \theta_1 x_n)] = 0$$

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_1} = 0 \Rightarrow -2 \sum_n [y_n - (\theta_0 + \theta_1 x_n)] x_n = 0$$

Simplify these expressions to get "Normal Equations"

$$\sum y_n = N\theta_0 + \theta_1 \sum x_n$$

$$\sum x_n y_n = \theta_0 \sum x_n + \theta_1 \sum x_n^2$$

We have two equations and two unknowns! Do some algebra to get:

$$\theta_1 = \frac{\sum (x_n - \bar{x})(y_n - \bar{y})}{\sum (x_i - \bar{x})^2} \qquad \text{and} \qquad \theta_0 = \bar{y} - \theta_1 \bar{x}$$

where $\bar{x} = \frac{1}{n}\sum_n x_n$ and $\bar{y} = \frac{1}{n}\sum_n y_n$.

Why is minimizing J sensible?

Probabilistic interpretation

- Noisy observation model

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

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

Use properties of Gaussian random variables

$$Z \sim \mathcal{N}(0,1), W = a + bZ \quad \text{then} \quad W \sim \mathcal{N}(a, b^2)$$

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

- Distribution of Y determined by parameters $(\theta_0, \theta_1, \sigma^2)$.
- Likelihood of one training sample (x_n, y_n)

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

Probabilistic interpretation

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

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

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 for a fixed σ

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

- Maximize over σ

Why does the probabilistic interpretation help us?

Gives us a template for modeling

- 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.

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
- Constant variance σ^2

Outline

1. Linear regression
2. Multivariate solution
   - Input is D-dimensional
   - Computational and numerical optimization
3. Nonlinear hypotheses

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

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

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

where we have redefined some variables (by augmenting)

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

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

Design matrix and target vector

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

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

Design matrix and target vector

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

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

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

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

Vector of errors for a given θ

$$\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} - \begin{pmatrix} \boldsymbol{x}_1^T\boldsymbol{\theta} \\ \boldsymbol{x}_2^T\boldsymbol{\theta} \\ \vdots \\ \boldsymbol{x}_N^T\boldsymbol{\theta} \end{pmatrix} = \begin{pmatrix} y_1 - \boldsymbol{x}_1^T\boldsymbol{\theta} \\ y_2 - \boldsymbol{x}_2^T\boldsymbol{\theta} \\ \vdots \\ y_N - \boldsymbol{x}_N^T\boldsymbol{\theta} \end{pmatrix}$$

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

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

$$\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} - \begin{pmatrix} \boldsymbol{x}_1^T\boldsymbol{\theta} \\ \boldsymbol{x}_2^T\boldsymbol{\theta} \\ \vdots \\ \boldsymbol{x}_N^T\boldsymbol{\theta} \end{pmatrix} = \begin{pmatrix} y_1 - \boldsymbol{x}_1^T\boldsymbol{\theta} \\ y_2 - \boldsymbol{x}_2^T\boldsymbol{\theta} \\ \vdots \\ y_N - \boldsymbol{x}_N^T\boldsymbol{\theta} \end{pmatrix}$$

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

$$\begin{aligned} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}||_2^2 & \\ &= (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) \\ &= \begin{pmatrix} y_1 - \boldsymbol{x}_1^T\boldsymbol{\theta} & \cdots & y_N - \boldsymbol{x}_N^T\boldsymbol{\theta} \end{pmatrix} \begin{pmatrix} y_1 - \boldsymbol{x}_1^T\boldsymbol{\theta} \\ \vdots \\ y_N - \boldsymbol{x}_N^T\boldsymbol{\theta} \end{pmatrix} \\ &= \sum_n [y_n - \boldsymbol{x}_n^T\boldsymbol{\theta}]^2 \leftarrow \text{That is } J(\boldsymbol{\theta}) \end{aligned}$$

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

Compact expression

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

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 $\boldsymbol{A}$)

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

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

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

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

This leads to the linear regression solution

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

Mini-Summary

- Linear regression is the linear combination of features $h : \boldsymbol{x} \rightarrow y$, with $h(\boldsymbol{x}) = \theta_0 + \sum_d \theta_d x_d = \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{x}$
- If we minimize residual sum of squares as our learning objective, we get a closed-form solution of parameters
- Probabilistic interpretation: maximum likelihood if assuming the output is Gaussian distributed

Computational complexity

Bottleneck of computing the solution?

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

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

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?

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

Alternative method: an example of using numerical optimization

(Batch) Gradient descent

Algorithm 1 Gradient Descent (J)

1: $t \leftarrow 0$
2: Initialize $\theta^{(0)}$
3: **repeat**
4: $\quad \nabla J(\boldsymbol{\theta}^{(t)}) = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\theta}^{(t)} - \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y} = \sum_n (\boldsymbol{x}_n^{\mathrm{T}}\boldsymbol{\theta}^{(t)} - y_n)\boldsymbol{x}_n$
5: $\quad \boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta \nabla J(\boldsymbol{\theta}^{(t)})$
6: $\quad t \leftarrow t + 1$
7: **until** convergence
8: Return final value of $\boldsymbol{\theta}$

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 $\boldsymbol{\theta}$

Stochastic gradient descent

Update parameters using one example at a time

Algorithm 2 Stochastic Gradient Descent (J)

1: $t \leftarrow 0$
2: Initialize $\boldsymbol{\theta}^{(0)}$
3: **repeat**
4: Randomly choose a training a sample $\boldsymbol{x}_t$
5: Compute its contribution to the gradient $\boldsymbol{g}_t = (\boldsymbol{x}_t^{\mathrm{T}} \boldsymbol{\theta}^{(t)} - y_t)\boldsymbol{x}_t$
6: $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta \boldsymbol{g}_t$
7: $t \leftarrow t + 1$
8: **until** convergence
9: 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.
- Stochastic gradient descent approximates the gradient with a single data point;
- 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 $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ is not invertible

Can you think of any reasons why that could happen?

What if $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ 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?

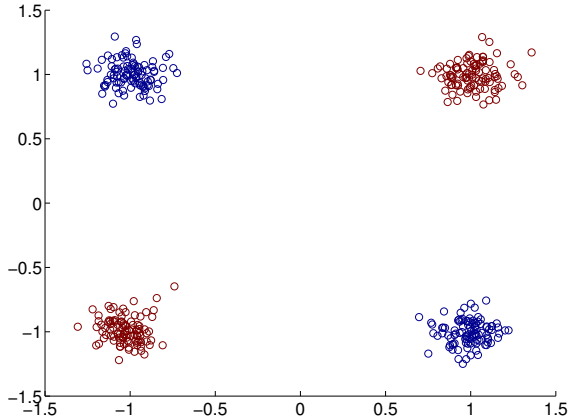
Answer 1: N $<$ D. Intuitively, not enough data to estimate all the parameters.

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

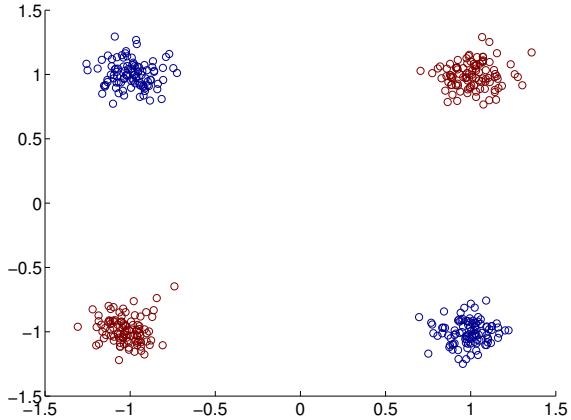
What if data is not linearly separable or fits to a line

Example of nonlinear classification

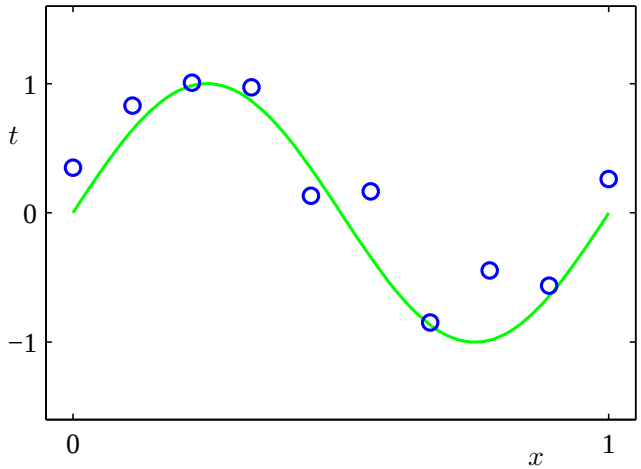


What if data is not linearly separable or fits to a line

Example of nonlinear classification



Example of nonlinear regression



Nonlinear basis for classification

Transform the input/feature

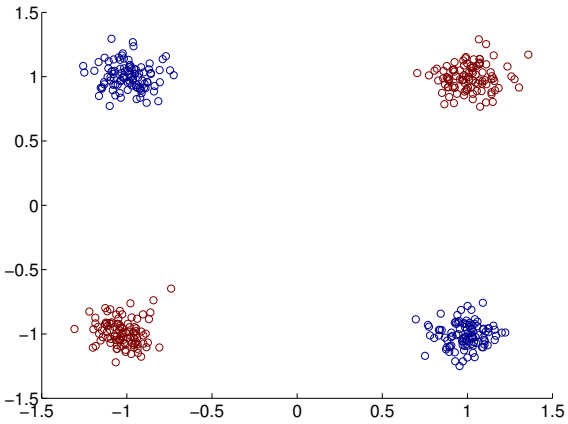
$$\phi(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^2 \rightarrow z = x_1 \cdot x_2$$

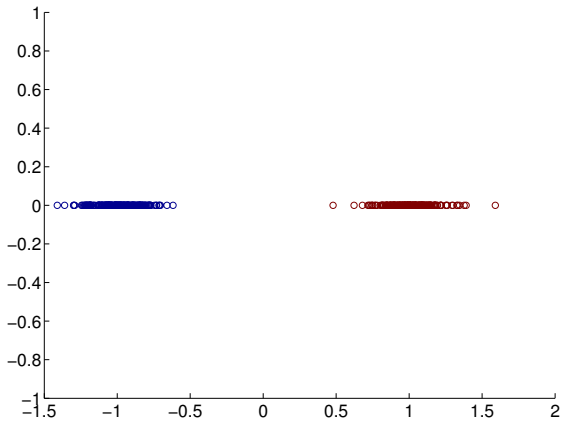
Nonlinear basis for classification

Transform the input/feature

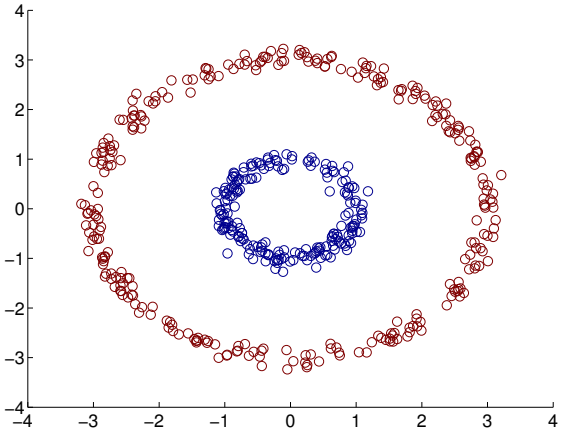
$$\phi(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^2 \rightarrow z = x_1 \cdot x_2$$

Transformed training data: linearly separable!



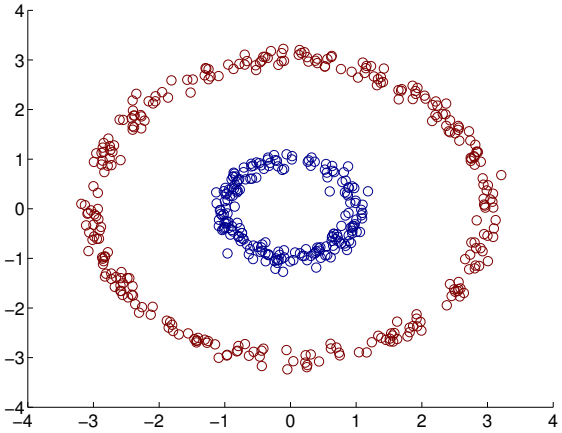


Another example



How to transform the input/feature?

Another example

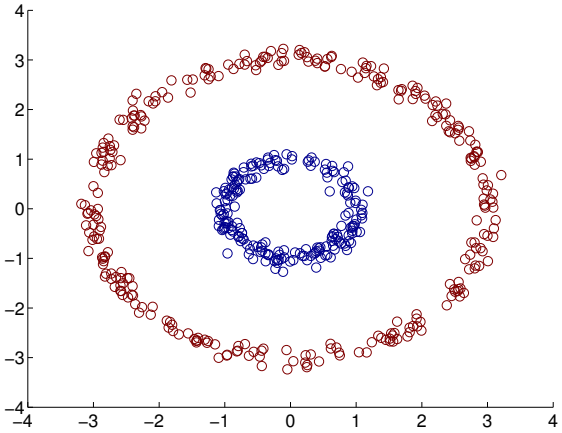


How to transform the input/feature?

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

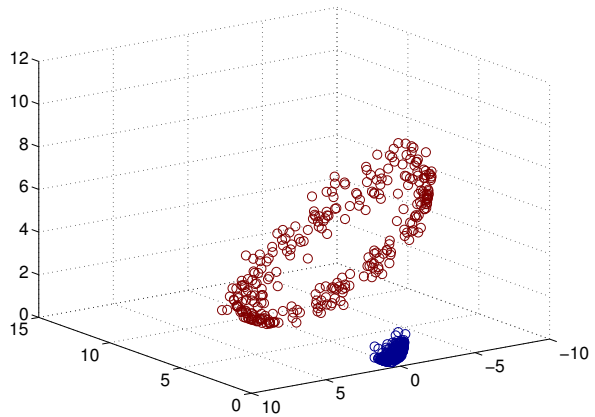
Another example

How to transform the input/feature?



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

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

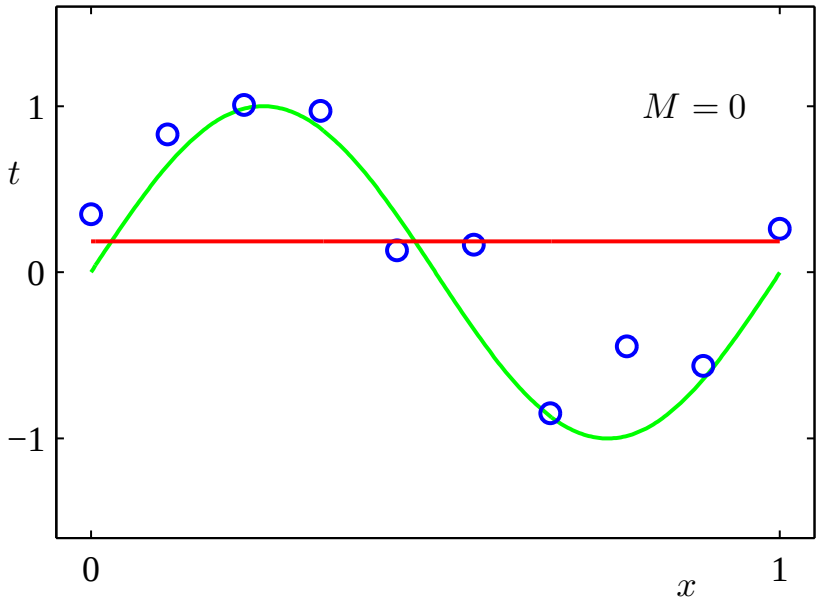
$$\boldsymbol{\phi}(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^M \end{bmatrix} \Rightarrow h(x) = \theta_0 + \sum_{m=1}^{M} \theta_m x^m$$

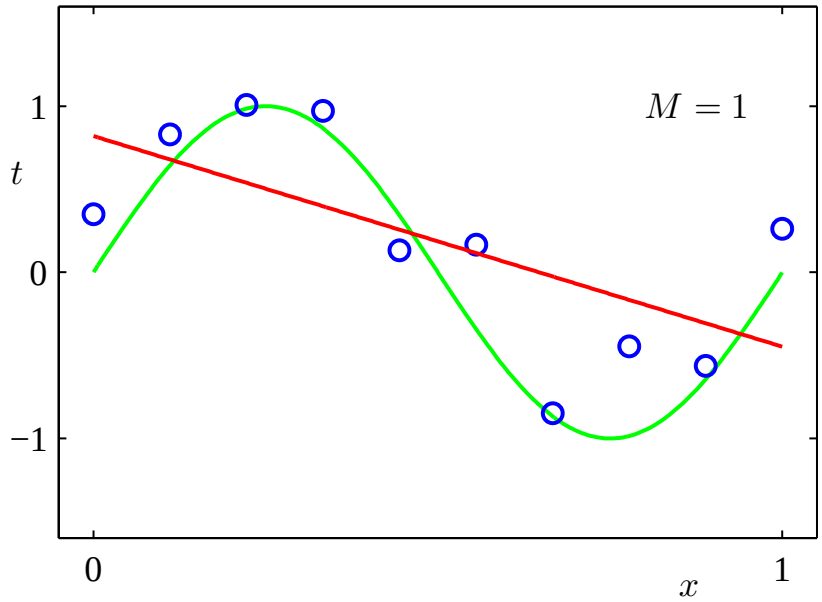
Example with regression

Polynomial basis functions

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

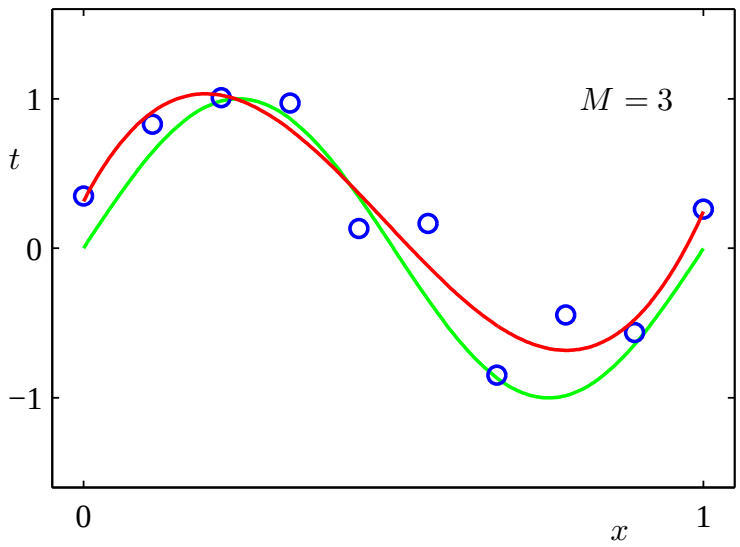
Fitting samples from a sine function: *underrfitting* as $h(x)$ is too simple





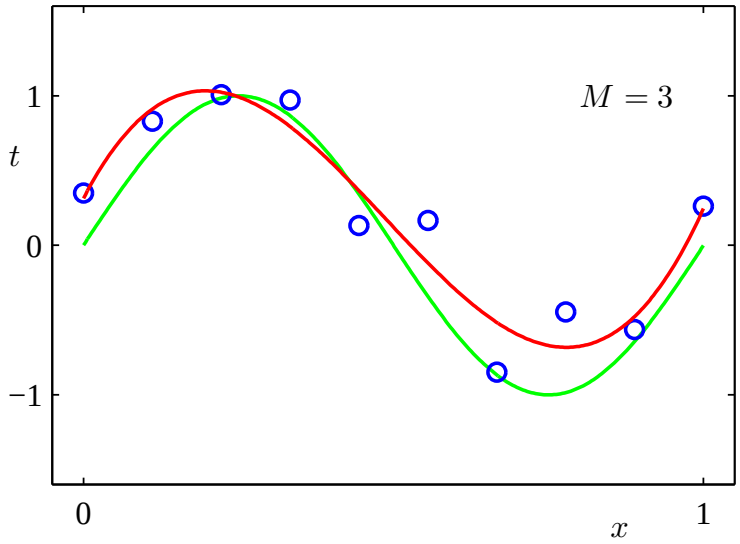
Adding high-order terms

M=3

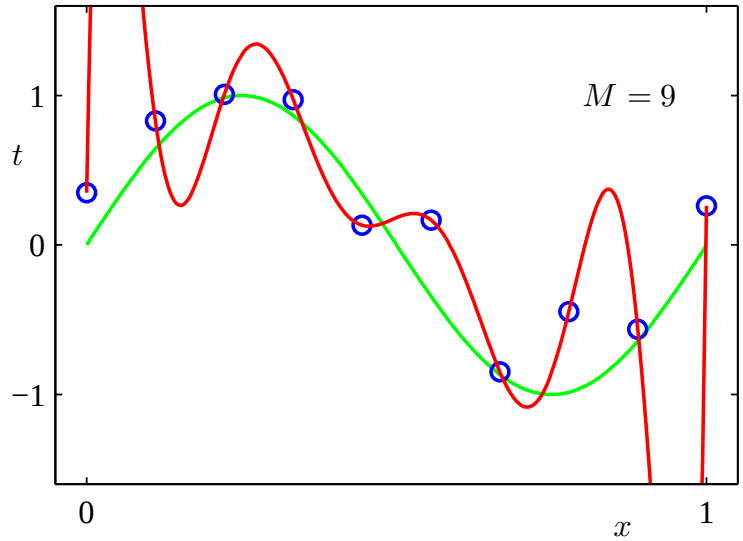


Adding high-order terms

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!

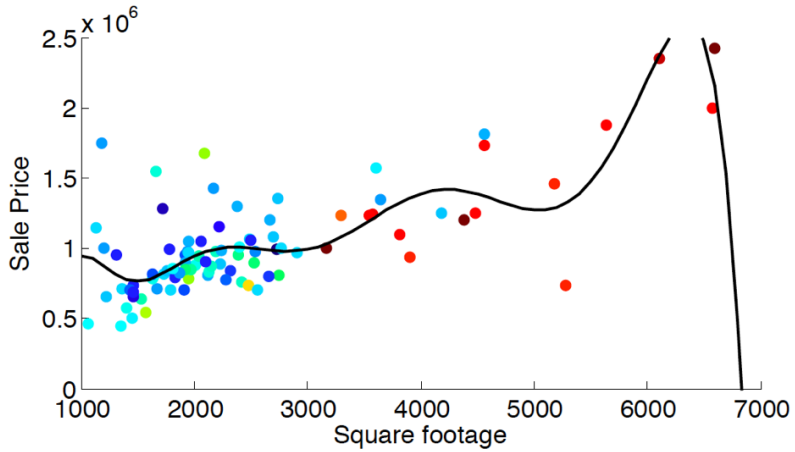
Overfitting

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 $M = 3$

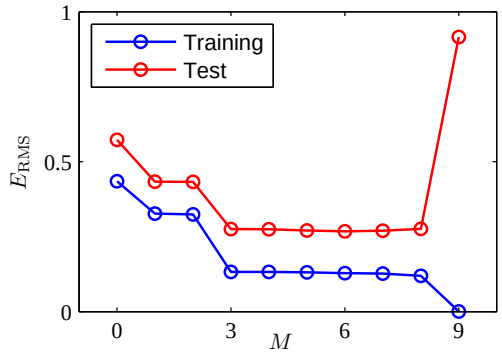


Note that the price would goes to zero (or negative) if you buy bigger ones!
This is called poor generalization/overfitting.

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.

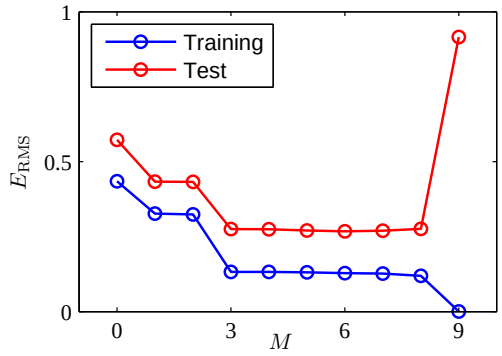


- Horizontal axis: *measure of model complexity*; in this example complexity defined by order of the polynomial basis functions.

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.

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.