10. Least squares data fitting

- data-fitting (model fitting)
- examples
- validation and over-fitting
- feature engineering
- classification

Model fitting

Setup: a scalar y and an n-vector x are related by model, $f : \mathbb{R}^n \to \mathbb{R}$,

$$y \approx f(x)$$

- x is the *independent variable* or *feature vector*
- y is the *outcome* or *response variable*
- we don't know f, which gives the 'true' relationship between x and y

Data: we are given some data (*observations*, *samples*, or *measurements*)

$$x^{(1)}, \ldots, x^{(N)}, \quad y^{(1)}, \ldots, y^{(N)}$$

- $x^{(i)}, y^{(i)}$ is ith *data pair*
- $x_j^{(i)}$ is the jth component of ith data point $x^{(i)}$

Model fitting: choose model $\hat{f}$ that approximates f based on observations

Prediction error

- $\hat{y}^{(i)} = \hat{f}(x^{(i)})$ is (the model's) *prediction* of $y^{(i)}$
- goal: $\hat{y}^{(i)} \approx y^{(i)}$ (model is consistent with observed data)
- $r_i = y^{(i)} - \hat{y}^{(i)}$ is *prediction error* or *residual*

Data fitting problem: choose model to minimize RMS or mean-square error (MSE) prediction error on data set

$$\text{RMS} = \left(\frac{1}{N} \sum_{i=1}^{N} (r^{(i)})^2 \right)^{1/2}$$

or

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (r^{(i)})^2$$

Linear in parameters model

$$\hat{f}(x) = \theta_1 f_1(x) + \cdots + \theta_p f_p(x)$$

- $f_i : \mathbb{R}^n \longrightarrow \mathbb{R}$ are *basis functions* or *feature mappings* that we choose
- θ_i are *model parameters* that we choose

Linear in parameters model fitting: choose θ_i to minimize

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(r^{(i)})^2 = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \hat{f}(x^{(i)})\right)^2$$

- fit linear-in-parameters model to data set $(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})$
- residual for data sample i is

$$r^{(i)} = y^{(i)} - \hat{f}(x^{(i)}) = y^{(i)} - \theta_1 f_1(x^{(i)}) - \cdots - \theta_p f_p(x^{(i)})$$

Least squares data fitting

a least squares problem:

$$\text{minimize} \quad \|A\theta - y^{\mathrm{d}}\|^2$$

with

$$A = \begin{bmatrix} f_1(x^{(1)}) & \cdots & f_p(x^{(1)}) \\ f_1(x^{(2)}) & \cdots & f_p(x^{(2)}) \\ \vdots & & \vdots \\ f_1(x^{(N)}) & \cdots & f_p(x^{(N)}) \end{bmatrix}, \quad \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_p \end{bmatrix}, \quad y^{\mathrm{d}} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix}$$

- $\hat{y} = A\hat{\theta}$ is our prediction of y^{d}, residual $r^{\mathrm{d}} = \hat{y} - y$
- $\|r^{\mathrm{d}}\|^2/N$ is minimum mean-square error (MMSE)
- RMS prediction error $\mathrm{rms}(r^{\mathrm{d}}) = \|r^{\mathrm{d}}\|/\sqrt{N}$; ratio $\frac{\mathrm{rms}(r^{\mathrm{d}})}{\mathrm{rms}(y^{\mathrm{d}})}$ is relative error
- notation remark: here θ is the variable and x refers to given data points

Example: fitting a constant model

model is a constant

$$\hat{f}(x) = \theta_1$$

- simplest possible model: $p = 1$, $f_1(x) = 1$
- $A = \mathbf{1}$, so

$$\hat{\theta}_1 = (\mathbf{1}^T\mathbf{1})^{-1}\mathbf{1}^T y^{\mathrm{d}} = (1/N)\mathbf{1}^T y^{\mathrm{d}} = \mathrm{avg}(y^{\mathrm{d}})$$

- the mean (average) of $y^{(1)}, \ldots, y^{(N)}$ is the least squares fit by a constant
- RMS error is $\mathrm{rms}(y - \mathrm{avg}(y^{\mathrm{d}})\mathbf{1}) = \mathrm{std}(y^{\mathrm{d}})$

Regression

recall the regression model:

$$\hat{y} = \hat{f}(x) = x^T\beta + v$$

- least squares regression: choose the model parameters v, β that minimize

$$\frac{1}{N}\sum_{i=1}^{N}\left(v + (x^{(i)})^T\beta - y^{(i)}\right)^2 = \|A\theta - y^{\mathrm{d}}\|^2$$

with

$$A = \begin{bmatrix} 1 & (x^{(1)})^T \\ \vdots & \vdots \\ 1 & (x^{(N)})^T \end{bmatrix}, \quad \theta = \begin{bmatrix} v \\ \beta \end{bmatrix}, \quad y^{\mathrm{d}} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(N)} \end{bmatrix}$$

- same as data fitting with basis functions

$$f_1(x) = 1, \; f_i(x) = x_{i-1}, \quad i = 2, \ldots, n+1$$

Outline

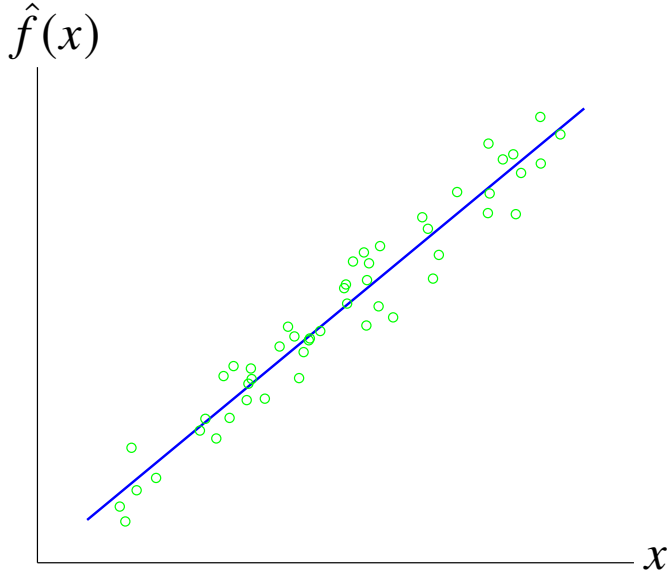
- data-fitting (model fitting)
- **examples**
- validation and over-fitting
- feature engineering
- classification

Fitting univariate functions

- approximate a univariate function $f : \mathbb{R} \to \mathbb{R}$ ($n = 1$)
- we can plot the data $(x^{(i)}, y^{(i)})$ and the model function $\hat{y} = \hat{f}(x)$

Straight-line fit

$$\hat{f}(x) = \theta_1 + \theta_2 x$$



- $p = 2$, $f_1(x) = 1$, $f_2(x) = x$
- matrix A has form

$$A = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ \vdots & \vdots \\ 1 & x^{(N)} \end{bmatrix}$$

Asset α and β in finance

predict the return of an individual asset from the return of the whole market

- x is return of whole market over some period
- y is return of a particular asset over some period
- a widely used model is the straight-line model

$$\hat{y} = \left(r^{\mathrm{rf}} + \alpha\right) + \beta\left(x - \mu^{\mathrm{mkt}}\right)$$

  - r^{rf} is the risk-free interest rate over the period
  - μ^{mkt} is the average market return
- 'α' and 'β ' called *asset*
- other models exists

Polynomial fit

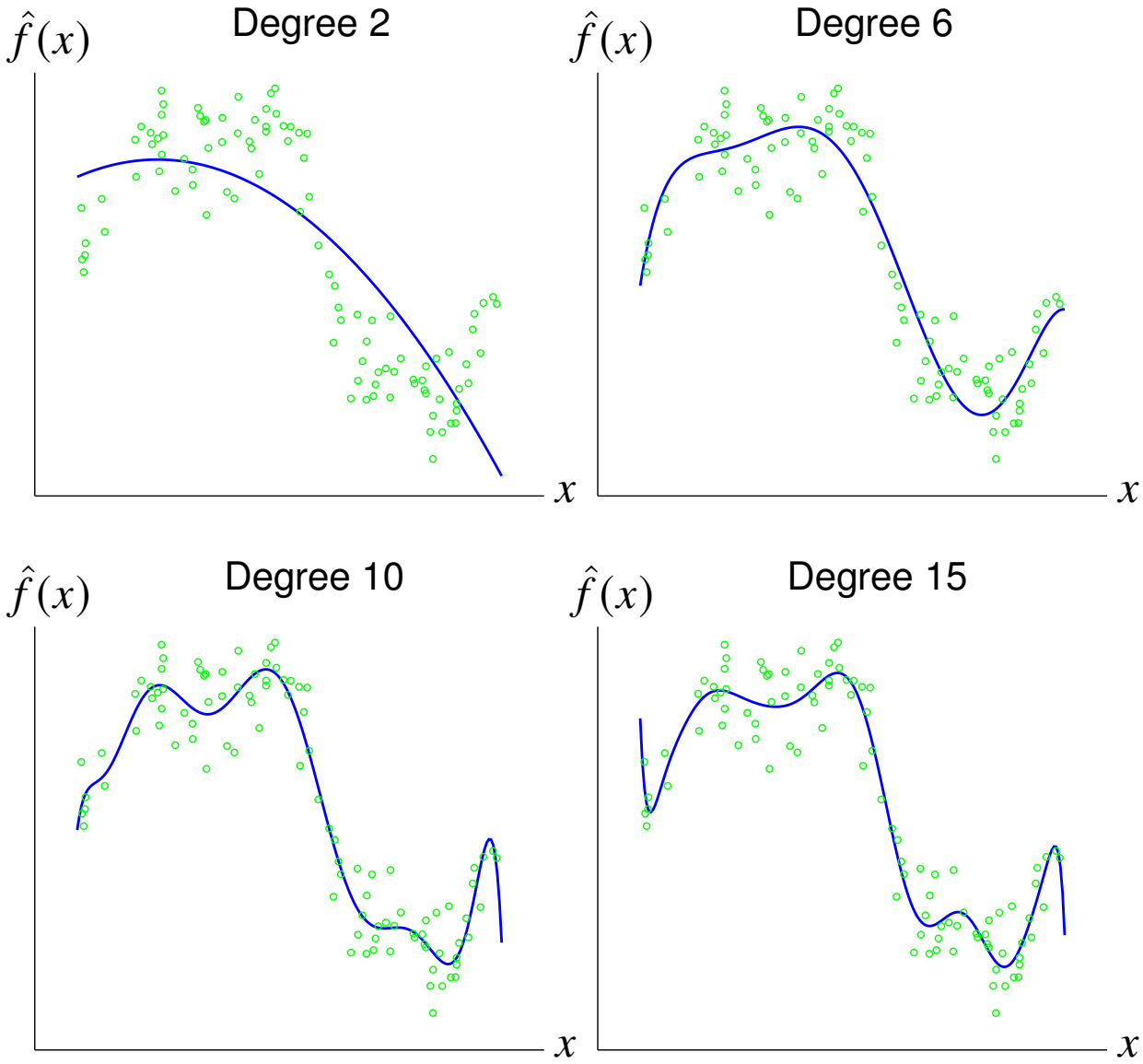
model is a polynomial of degree at most $p-1$:

$$\hat{f}(x) = \theta_1 + \theta_2 x + \cdots + \theta_p x^{p-1}$$

- $f_i(x) = x^{i-1}, \quad i = 1, \ldots, p$; here x^i means scalar x to ith power
- $x^{(i)}$ is ith data point
- A is Vandermonde matrix

$$A = \begin{bmatrix} 1 & x^{(1)} & \cdots & (x^{(1)})^{p-1} \\ 1 & x^{(2)} & \cdots & (x^{(2)})^{p-1} \\ \vdots & \vdots & & \vdots \\ 1 & x^{(N)} & \cdots & (x^{(N)})^{p-1} \end{bmatrix}$$

Example: $N = 100$ data points

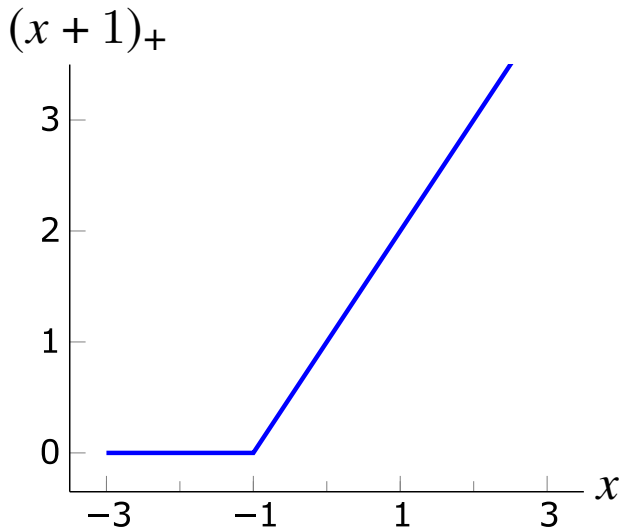


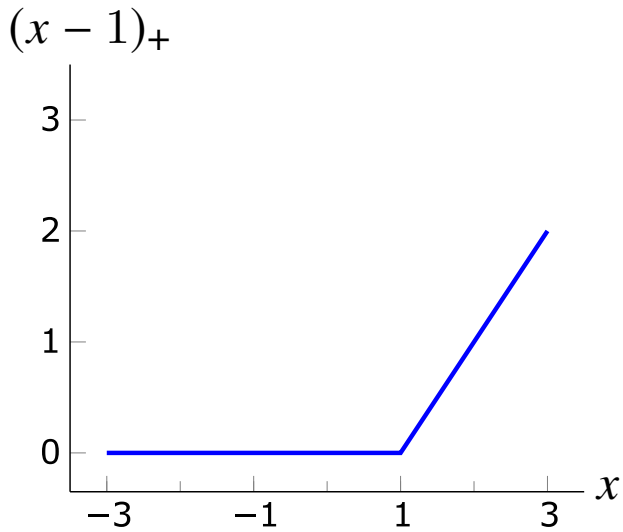
Piecewise-linear fit

- define *knot points* $a_1 < a_2 < \cdots < a_k$ on the real axis
- *piecewise-linear function* is continuous, and affine on each interval $[a_j, a_{j+1}]$
- piecewise-linear function with knot points $a_1, \ldots, a_k$ can be written as

$$\hat{f}(x) = \theta_1 + \theta_2 x + \theta_3 (x - a_1)_+ + \cdots + \theta_{2+k} (x - a_k)_+$$

where $u_+ = \max\{u, 0\}$



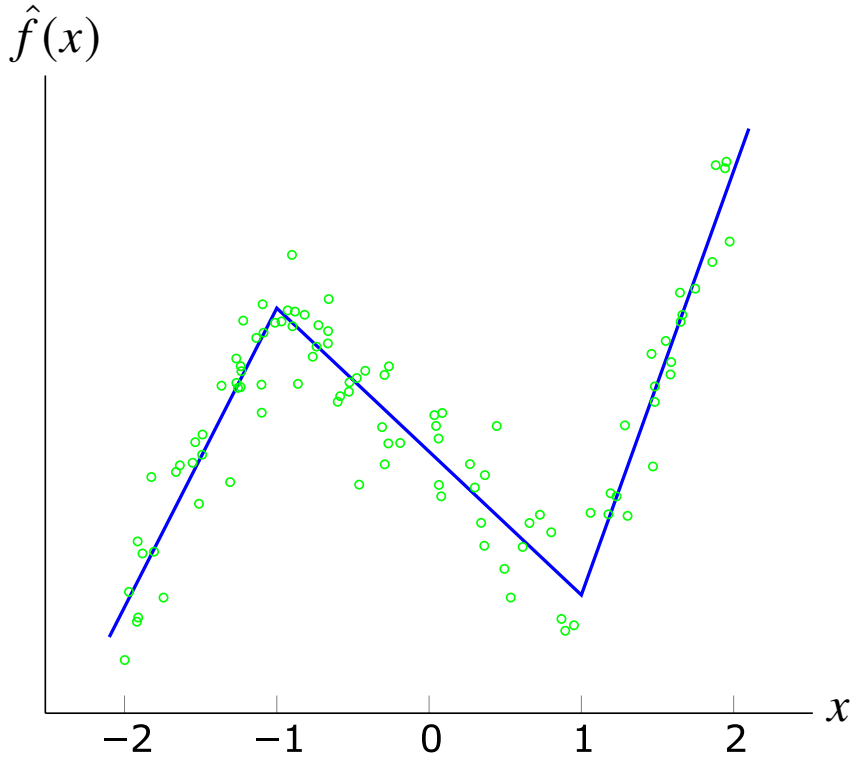


Piecewise-linear fitting

piecewise-linear model is linear in the parameters θ, with basis functions

$$f_1(x) = 1, \quad f_2(x) = x, \quad f_3(x) = (x - a_1)_+, \quad \ldots, \quad f_{k+2}(x) = (x - a_k)_+$$

Example: fit piecewise-affine function with knots $a_1 = -1, a_2 = 1$ to 100 points



Time series trend

- N data samples from time series: $y^{(1)}, \ldots, y^{(N)}$
- straight-line fit

$$\hat{y}^{(i)} = \theta_1 + \theta_2 i$$

  is called the *trend line* (θ_2 is trend coefficient)

- least squares fitting of trend line: minimize $\|A\theta - y^{\mathrm{d}}\|^2$ with

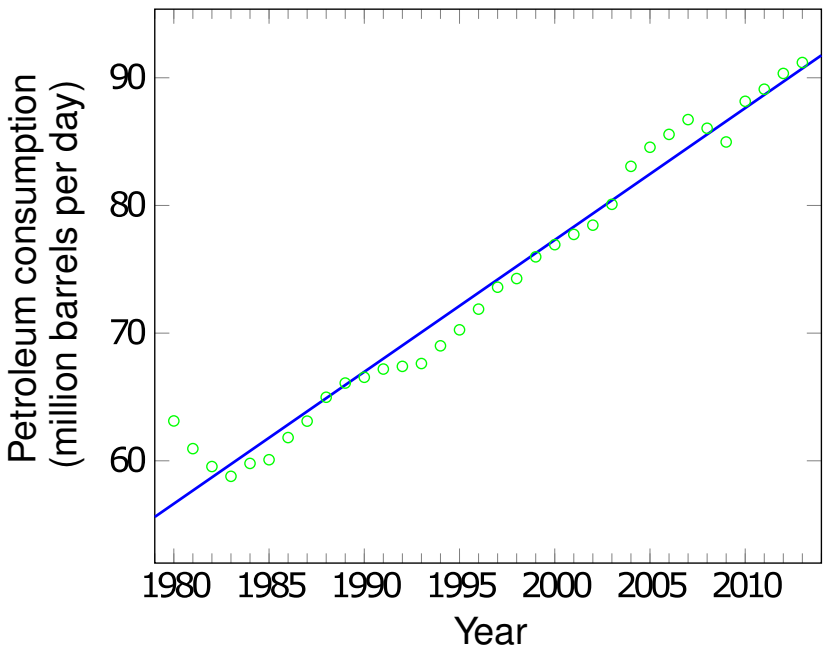
$$A = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ \vdots & \vdots \\ 1 & N \end{bmatrix}, \quad y^{\mathrm{d}} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix}$$

- $y^{\mathrm{d}} - \hat{y}^{\mathrm{d}} = (y^{(1)} - \hat{y}^{(1)}, \ldots, y^{(N)} - \hat{y}^{(N)})$ is the *de-trended time series*

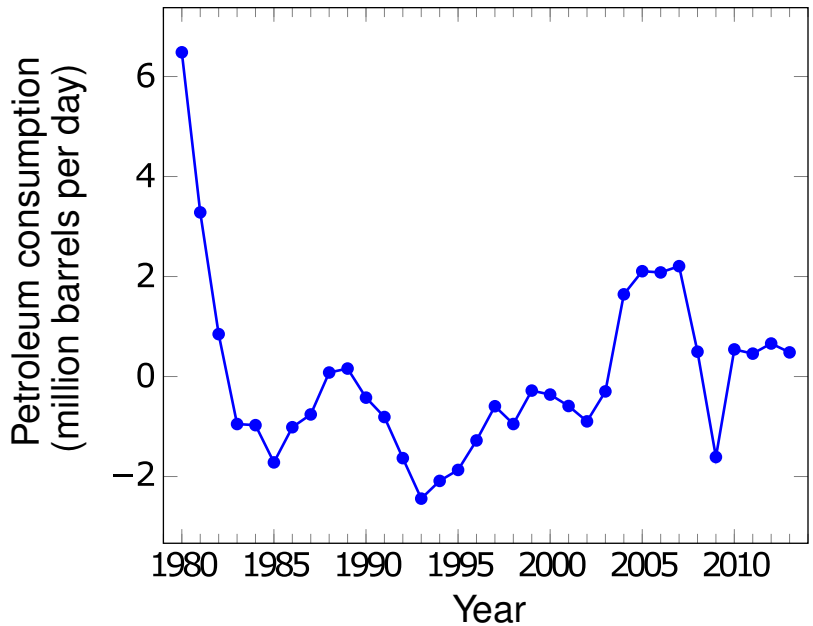
Example: world petroleum consumption

time series of world petroleum consumption (million barrels/day) versus year

Consumption



De-trended consumption



- left figure shows data samples and trend line
- right figure shows de-trended time series

Trend and seasonal component

- model time series as a linear trend plus a periodic component with period P:

$$\hat{y}^{\mathrm{d}} = \hat{y}^{\mathrm{lin}} + \hat{y}^{\mathrm{seas}}$$

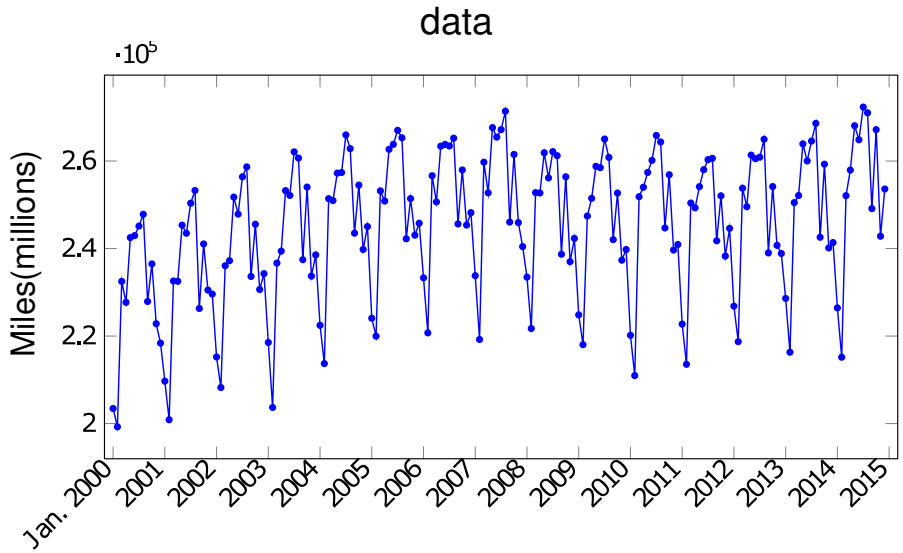
  with $\hat{y}^{\mathrm{lin}} = \theta_1(1, 2, \ldots, N)$ and

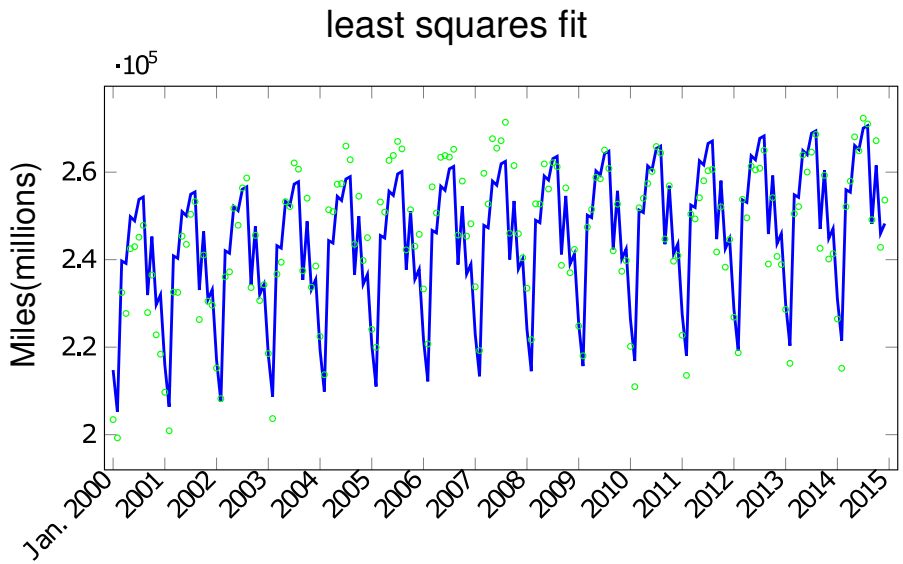
$$\hat{y}^{\mathrm{seas}} = (\theta_2, \theta_3, \ldots, \theta_{P+1}, \theta_2, \theta_3, \ldots, \theta_{P+1}, \ldots, \theta_2, \theta_3, \ldots, \theta_{P+1})$$

- the mean of $\hat{y}^{\mathrm{seas}}$ serves as a constant offset
- residual $y^{\mathrm{d}} - \hat{y}^{\mathrm{d}}$ is the de-trended, seasonally adjusted time series
- least squares formulation: minimize $\|A\theta - y^{\mathrm{d}}\|^2$ with

$$A_{1:N,1} = \begin{bmatrix} 1 \\ 2 \\ \vdots \\ N \end{bmatrix}, \quad A_{1:N,2:P+1} = \begin{bmatrix} I_P \\ I_P \\ \vdots \\ I_P \end{bmatrix}, \quad y^{\mathrm{d}} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix}$$

Example: vehicle miles traveled in the US per month





Auto-regressive time series model

auto-regressive model (AR model) for the time series, $z_1, z_2, \ldots$, has the form

$$\hat{z}_{t+1} = \theta_1 z_t + \theta_2 z_{t-1} + \cdots + \theta_M z_{t-M+1}, \quad t = M, M+1, \ldots$$

- $\hat{z}_{t+1}$ is a prediction of z_{t+1}, made at time t
- prediction $\hat{z}_{t+1}$ is a linear function of previous M values $z_t, \ldots, z_{t-M+1}$
- M is the *memory* of the model

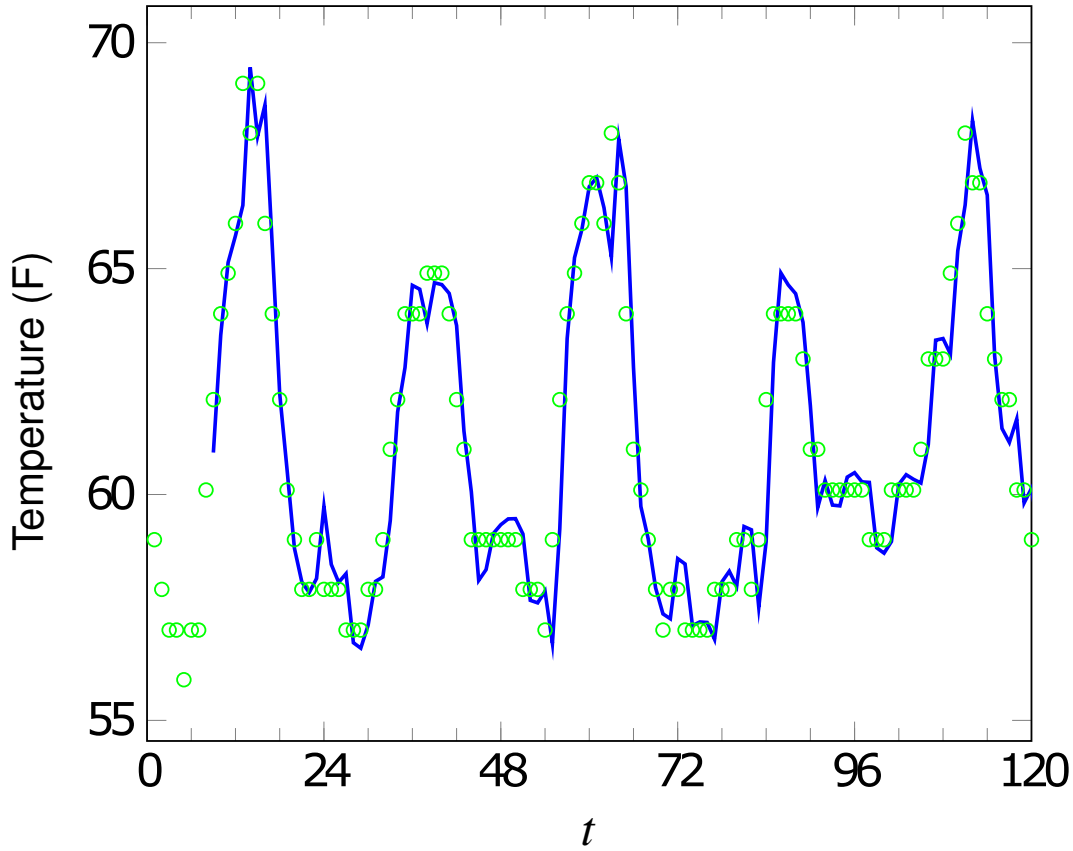
Least squares fitting of AR model: given observed data $z_1, \ldots, z_T$, minimize

$$(z_{M+1} - \hat{z}_{M+1})^2 + (z_{M+2} - \hat{z}_{M+2})^2 + \cdots + (z_T - \hat{z}_T)^2$$

this is a least squares problem: minimize $\|A\theta - y^{\mathrm{d}}\|^2$ with

$$A = \begin{bmatrix} z_M & z_{M-1} & \cdots & z_1 \\ z_{M+1} & z_M & \cdots & z_2 \\ \vdots & \vdots & & \vdots \\ z_{T-1} & z_{T-2} & \cdots & z_{T-M} \end{bmatrix}, \quad \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_M \end{bmatrix}, \quad y^{\mathrm{d}} = \begin{bmatrix} z_{M+1} \\ z_{M+2} \\ \vdots \\ z_T \end{bmatrix}$$

Example: hourly temperature at LAX in May 2016, length 744



- predictor $\hat{z}_{t+1} = z_t$ has RMS error $1.16^\circ\mathrm{F}$
- predictor $\hat{z}_{t+1} = z_{t-23}$ has RMS error $1.73^\circ\mathrm{F}$
- AR model with $M = 8$ gives RMS error $0.98^\circ\mathrm{F}$
- solid line shows one-hour ahead predictions from AR model, first 5 days

Outline

- data-fitting (model fitting)
- examples
- **validation and over-fitting**
- feature engineering
- classification

Generalization and validation

Generalization

- goal of model fitting is typically to achieve a good fit on *new* unseen data
- model with good predictions on new, unseen data has *generalization ability*

(Out-of-sample) validation: to assess generalization ability,

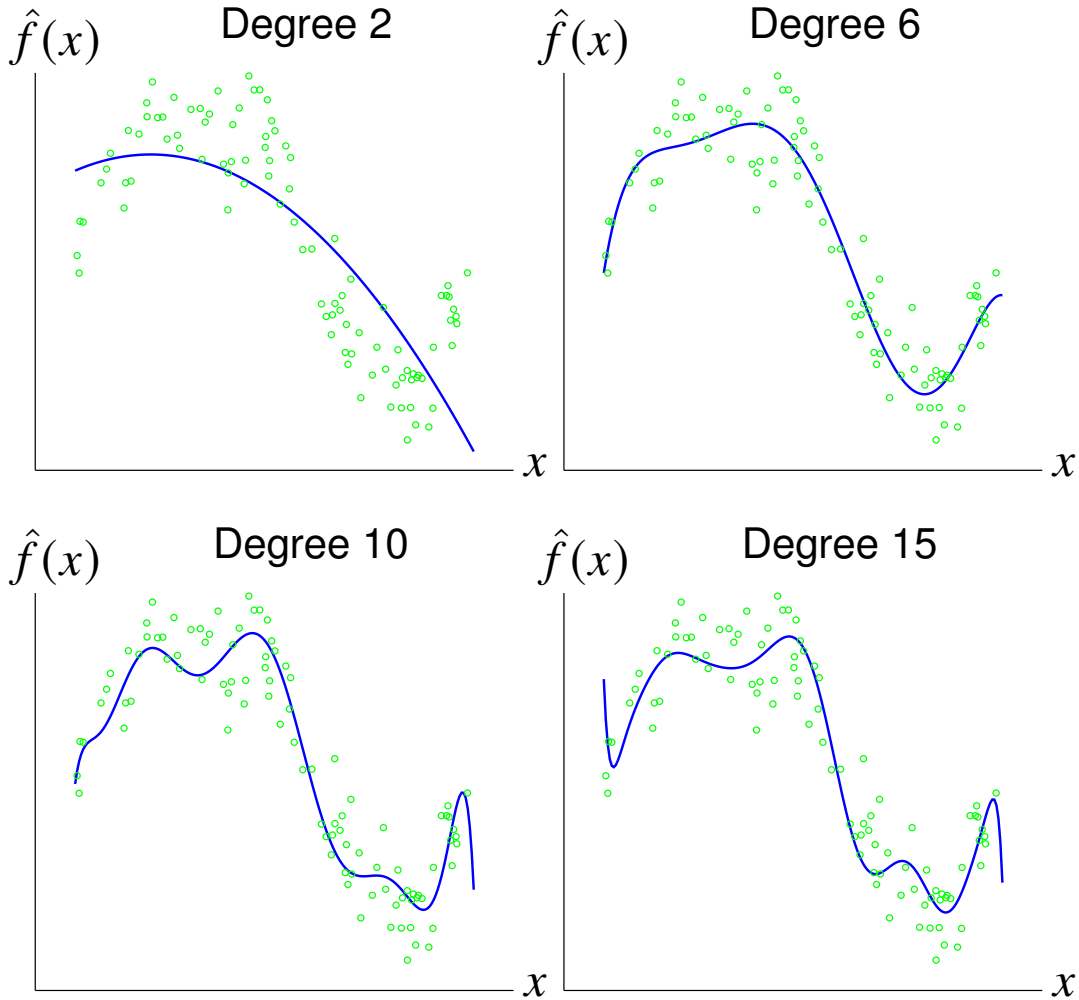
- divide data in two sets: training set and test (validation) set (*e.g.*, $80\%/20\%$)
- use only training set to fit model
- use test set to get an idea of generalization ability
  - compare MSE/RMS prediction error on train and test data
  - if they are similar, we can guess the model will generalize

Over-fit model

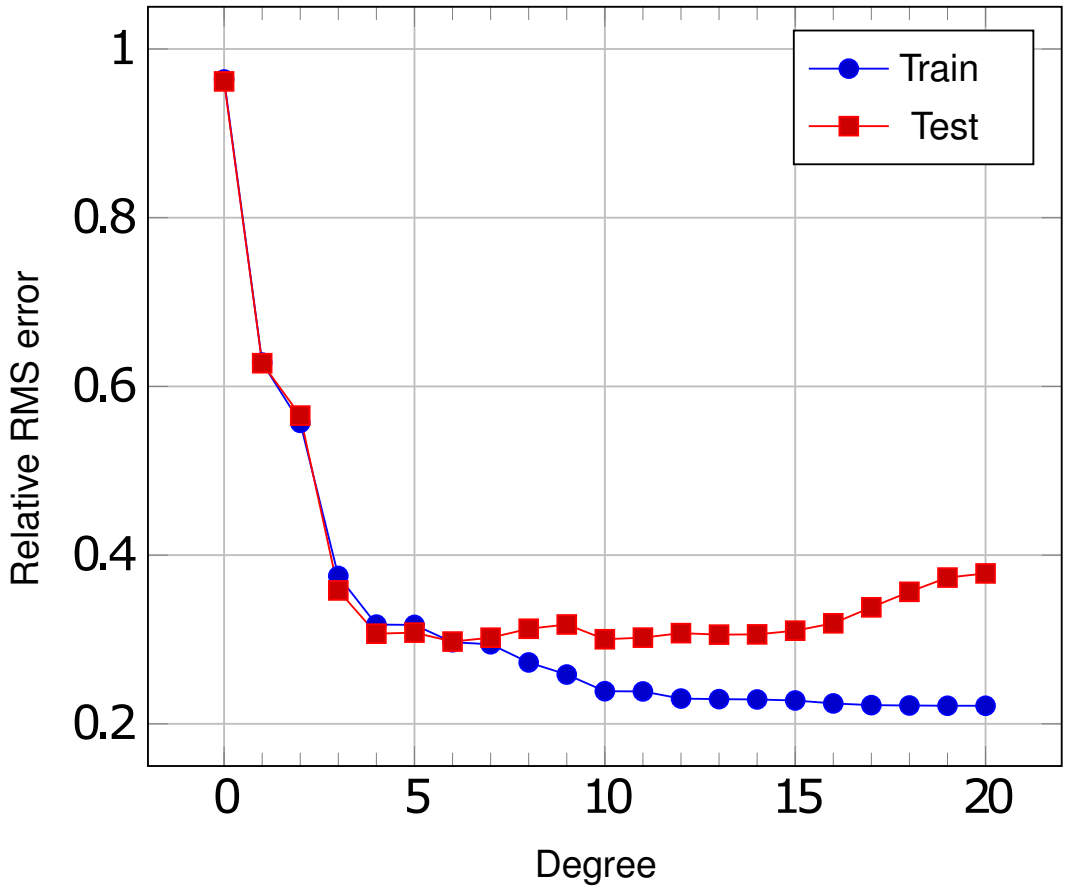
- model that makes poor predictions on new, unseen data suffers from over-fit
- prediction error on training set is much smaller than on test set

Example

polynomial fit using training set of 100 points; circles show test set of 100 points



Example



- suggests degree 4, 5, or 6 are reasonable choices
- models with degrees 0, 1, and 2 have good generalization ability, but worse prediction performance

Cross validation

Cross validation

- divide data into K *folds* (typically $5 \leq K \leq 10$)
- for $i = 1$ to K, fit model i using fold i as test set and other data as training set
- compare parameters and train/test RMS errors for the K models

Notes

- cross-validation is used to asses choice of basis functions used in the model
- if training and test set errors are similar, then our model is not over-fit
- RMS cross-validation error is defined as

$$\sqrt{(\epsilon_1^2 + \cdots + \epsilon_K^2)/K}$$

  where ϵ_i is training RMS error for model i

Example: house price prediction

$$\hat{y} = v + \beta_1 x_1 + \beta_2 x_2$$

- y is the selling price; $\hat{y}$ is prediction
- x_1 is the area (1000 square feet); x_2 is the number of bedrooms
- 774 sales, divided into 5 folds of 155 sales each
- fit 5 regression models, removing each fold

	Model parameters			RMS error	
Fold	v	β_1	β_2	Train	Test
1	60.65	143.36	-18.00	74.00	78.44
2	54.00	151.11	-20.30	75.11	73.89
3	49.06	157.75	-21.10	76.22	69.93
4	47.96	142.65	-14.35	71.16	88.35
5	60.24	150.13	-21.11	77.28	64.20

- ls fit over all data gives RMS error 74.8

Regularized data fitting

consider linear-in-parameters model

$$\hat{f}(x) = \theta_1 + \theta_2 f_2(x) + \cdots + \theta_p f_p(x)$$

- we fit the model $\hat{f}(x)$ to examples $(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})$
- large coefficient θ_i makes model more sensitive to changes in $f_i(x)$
- keeping $\theta_2, \ldots, \theta_p$ small helps avoid over-fitting
- this leads to two objectives:

$$J_1(\theta) = \sum_{k=1}^{N} \left(\hat{f}(x^{(k)}) - y^{(k)}\right)^2, \quad J_2(\theta) = \sum_{j=2}^{p} \theta_j^2 = \|\theta_{2:p}\|^2$$

  primary objective $J_1(\theta)$ is sum of squares of prediction errors

Weighted least squares formulation

$$\text{minimize} \quad J_1(\theta) + \lambda J_2(\theta) = \sum_{k=1}^{N} \left(\hat{f}(x^{(k)}) - y^{(k)}\right)^2 + \lambda \sum_{j=2}^{p} \theta_j^2$$

- λ is positive regularization parameter
- equivalent to least squares problem:

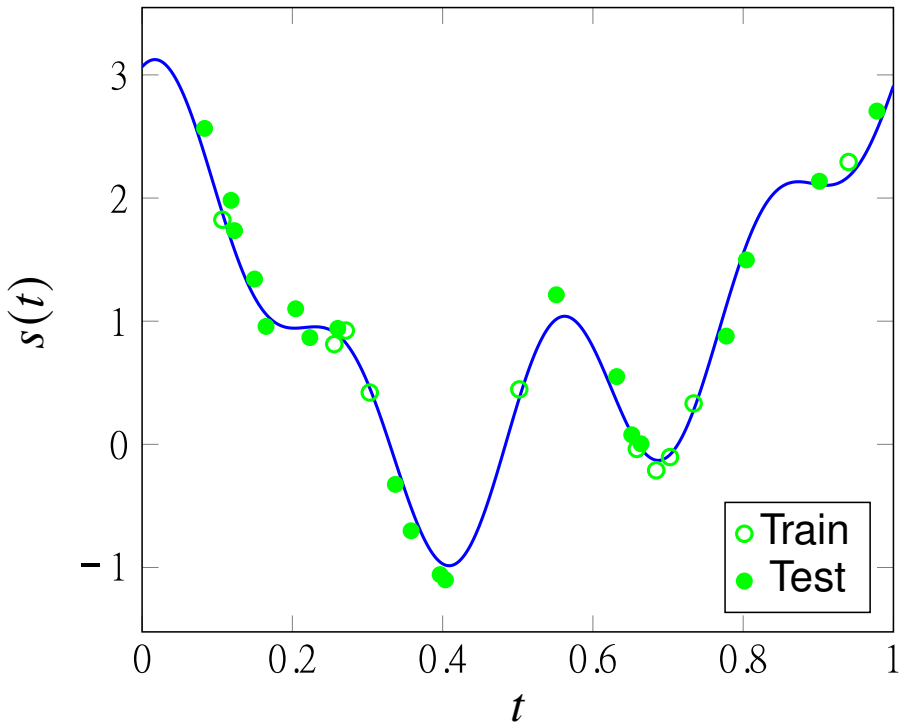
$$\text{minimize} \quad \left\| \begin{bmatrix} A_1 \\ \sqrt{\lambda} A_2 \end{bmatrix} \theta - \begin{bmatrix} y^{\mathrm{d}} \\ 0 \end{bmatrix} \right\|^2$$

with $y^{\mathrm{d}} = (y^{(1)}, \ldots, y^{(N)})$

$$A_1 = \begin{bmatrix} 1 & f_2(x^{(1)}) & \cdots & f_p(x^{(1)}) \\ 1 & f_2(x^{(2)}) & \cdots & f_p(x^{(2)}) \\ \vdots & \vdots & & \vdots \\ 1 & f_2(x^{(N)}) & \cdots & f_p(x^{(N)}) \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

- stacked matrix has linearly independent columns (for positive λ)
- value of λ can be chosen by (out-of-sample) validation or cross-validation

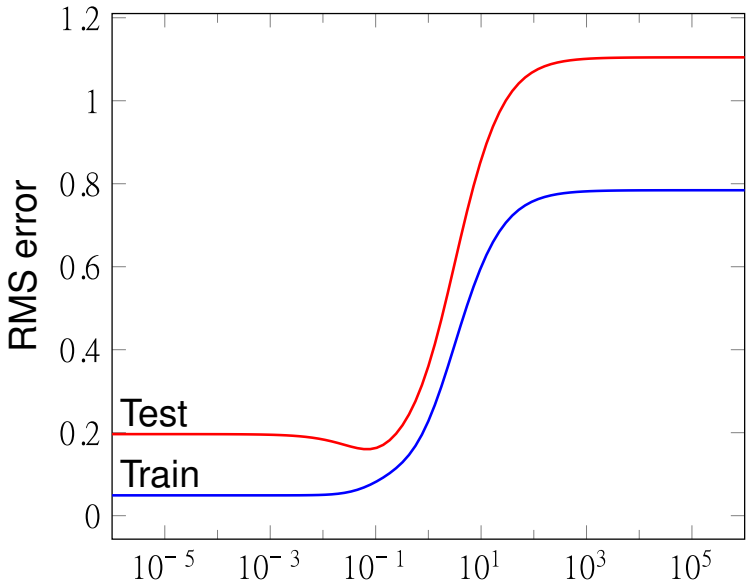
Example

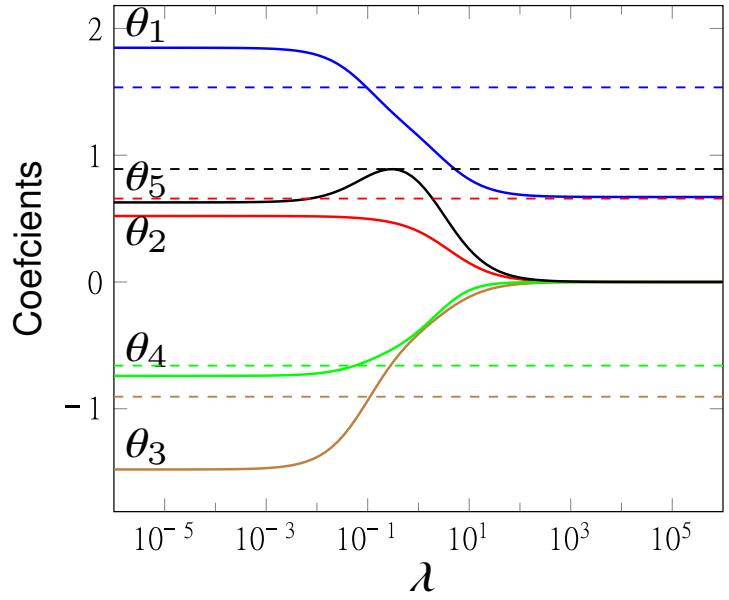


- solid line is signal used to generate synthetic (simulated) data
- 10 circle points are used as training set; 20 bullet points are used as test set
- we fit a model with five parameters $\theta_1, \ldots, \theta_5$:

$$\hat{f}(x) = \theta_1 + \sum_{k=1}^{4} \theta_{k+1} \sin(\omega_k x + \phi_k) \quad \text{(with given } \omega_k, \phi_k)$$

Result of regularized least squares fit





- minimum test RMS error is for λ around 0.08
- increasing λ 'shrinks' the coefficients $\theta_2, \ldots, \theta_5$
- dashed lines show coefficients used to generate the data (true coefficients)
- for λ near 0.08, estimated coefficients are close to these 'true' values
- any choice between 0.065 and 0.1 is reasonable

Outline

- data-fitting (model fitting)
- examples
- validation and over-fitting
- **feature engineering**
- classification

Feature engineering

$$\hat{y} = \hat{f}(x) = \theta_1 f_1(x) + \cdots + \theta_p f_p(x)$$

choosing the feature mapping functions $f_1, \ldots, f_p$ is called *feature engineering*

- start with original or base feature n-vector x
- choose basis functions $f_1, \ldots, f_p$ to create feature p-vector $(f_1(x), \ldots, f_p(x))$
- now fit linear in parameters model with mapped features
- check the model using validation

Remarks

- common to let $f_1(x) = 1$
- common to initially use $f_i(x) = x_{i-1}$, $i = 2, \ldots, n-1$ (basic regression model)
- if n is very large, feature mapping can be used to reduce dimension $p < n$

Transforming features

Standardizing features: replace x_i with z-scores

$$(x_i - b_i)/a_i$$

- $b_i \approx$ mean value of feature across data $x_i^{(1)}, \ldots, x_i^{(N)}$
- $a_i \approx$ standard deviation of feature across data
- standardization is first step in feature engineering
- constant feature $f_1(x) = 1$ cannot be standardized

Winsorizing (clipping) features

- clip the data values that include some very large errors
- example: replace standarized x_i with

$$\tilde{x}_i = \begin{cases} x_i & |x_i| \leq 3 \\ 3 & x_i > 3 \\ -3 & x_i < -3 \end{cases}$$

Log transform: if x_i is nonnegative and spans a wide range, replace it with

$$\log(x_i), \quad i = 2, \ldots, n-1$$

- use $\log(1 + x_i)$ if features has zero values
- compresses the range of values that we encounter

hi and lo features: create new features given by

$$\max\{x_i - b, 0\}, \quad \min\{x_i + a, 0\}$$

- example: $\hat{y} = \psi_1(x_1) + \cdots + \psi_n(x_n)$, where ψ_i is piecewise-linear function

$$\psi_i(x_i) = \theta_{n+i} \min\{x_i + a, 0\} + \theta_i x_i + \theta_{2n+i} \max\{x_i - b, 0\}$$

- model has $3n$ parameters (original plus two additional features per original feature)

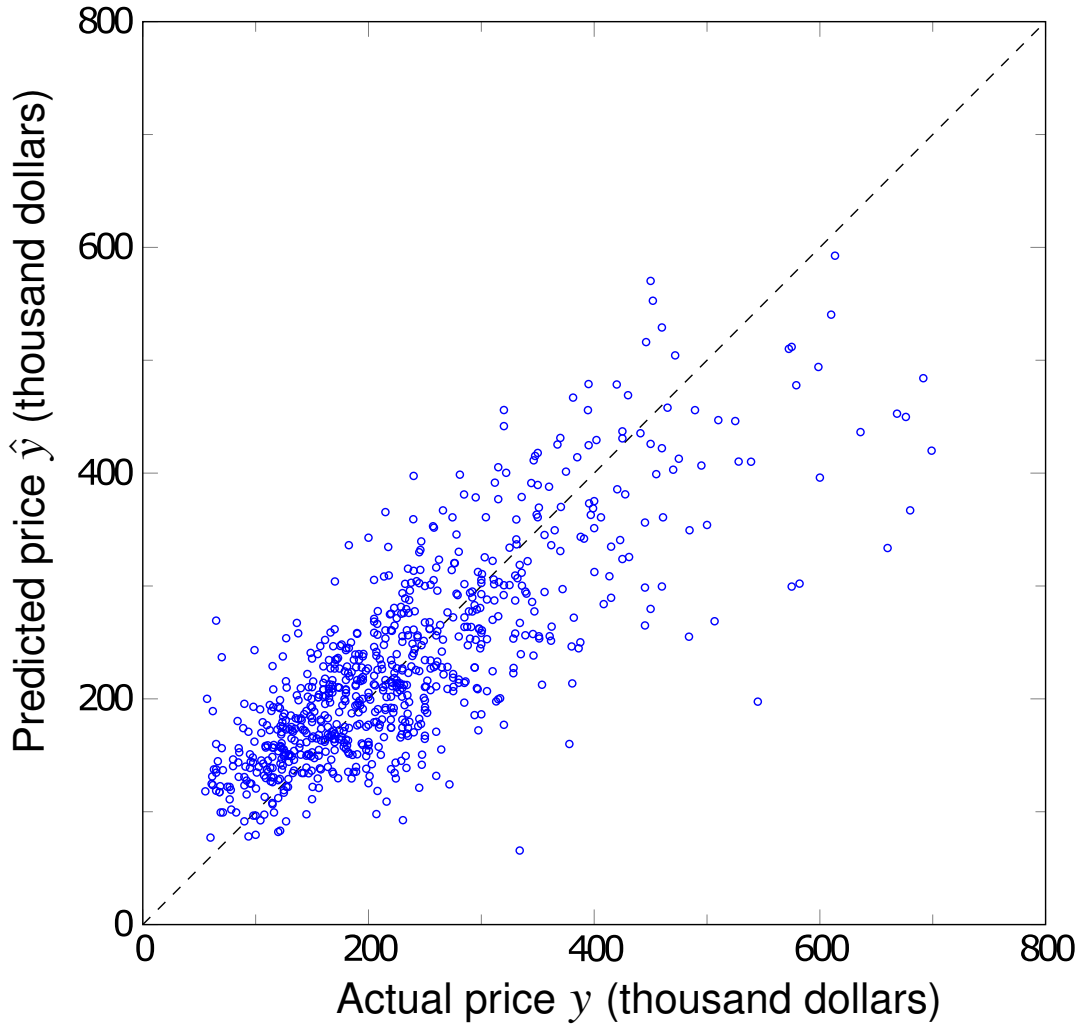
Example: house price regression model

$$\hat{y} = v + \tilde{x}_1\beta_1 + \cdots + \beta_7\tilde{x}_7$$

- $\tilde{x}_1$ is area of the house (in 1000 square feet)
- $\tilde{x}_2 = \max\{\tilde{x}_1 - 1.5, 0\}$, *i.e.*, area in excess of 1.5 (in 1000 square feet)
- $\tilde{x}_3$ is number of bedrooms
- $\tilde{x}_4$ is one for a condo; zero otherwise
- $\tilde{x}_5, \tilde{x}_6, \tilde{x}_7$ specify location (four groups of ZIP codes)

Location	$\tilde{x}_5$	$\tilde{x}_6$	$\tilde{x}_7$
A	0	0	0
B	1	0	0
C	0	1	0
D	0	0	1

Example: house price regression model



RMS fitting error is 68.3 (improved over original model 74.8 on page 10.24)

Outline

- data-fitting (model fitting)
- examples
- validation and over-fitting
- feature engineering
- **classification**

Classification

Classification

- N training data points $(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})$
- outcome y_i takes on finite number of values called *labels* or *categories*
  - TRUE or FALSE
  - SPAM or NOT SPAM
  - DOG, HORSE, or MOUSE
- data fitting $f(x^{(i)}) \approx y^{(i)}$ is called *classification*

Boolean (2-way) classification

- two possible outcomes only encoded as $y \in \{+1, -1\}$
- Boolean classifier has form $\hat{y} = \hat{f}(x)$, $\hat{f} : \mathbb{R}^n \to \{-1, +1\}$

Applications

Email spam detection

x contains features of an email message (word counts, origin of email, ...)

Financial transaction fraud detection

x contains features of proposed transaction, initiator, average balance

Document classification (*e.g.*, sport or politics or ...)

x is word count histogram of document

Disease detection

x contains patient features, results of medical tests, age, specific symptoms

Digital communications receiver

y is transmitted bit; x contain n measurements of received signal

Prediction errors

data point (x, y) with predicted outcome $\hat{y} = \hat{f}(x)$; only four possibilities:

	Prediction	
Outcome	$\hat{y} = +1$	$\hat{y} = -1$
$y = +1$	true positive	false negative
$y = -1$	false positive	true negative

Confusion matrix: count each of the four outcomes on data-set

	$\hat{y} = +1$	$\hat{y} = -1$	Total
$y = +1$	N_{tp}	N_{fn}	N_{p}
$y = -1$	N_{fp}	N_{tn}	N_{n}
All	$N_{\mathrm{tp}} + N_{\mathrm{fp}}$	$N_{\mathrm{fn}} + N_{\mathrm{tn}}$	N

- *error rate* is $(N_{\mathrm{fp}} + N_{\mathrm{fn}})/N$
- *true positive rate* or *recall rate* is $N_{\mathrm{tp}}/N_{\mathrm{p}}$
- *false positive rate* or *false alarm rate* is $N_{\mathrm{fp}}/N_{\mathrm{n}}$
- a classifier is judged by its error rate(s) on a test set

Example

spam filter performance on a test set (say)

	$\hat{y} = +1$ (SPAM)	$\hat{y} = -1$ (not SPAM)	Total
$y = +1$ (SPAM)	95	32	127
$y = -1$ (not SPAM)	19	1120	1139
All	114	1152	1266

- error rate is $(19 + 32)/1266 = 4.03\%$
- false positive rate is $19/1139 = 1.67\%$

Least squares Boolean classifier

- we are given the data points $(x^{(i)}, y^{(i)})$, $i = 1, \ldots, N$
- determine basis functions $f_1, \ldots, f_p$ for linear-in-parameter model

$$\tilde{f}(x) = \theta_1 f_1(x) + \theta_2 f_2(x) + \cdots + \theta_p f_p(x)$$

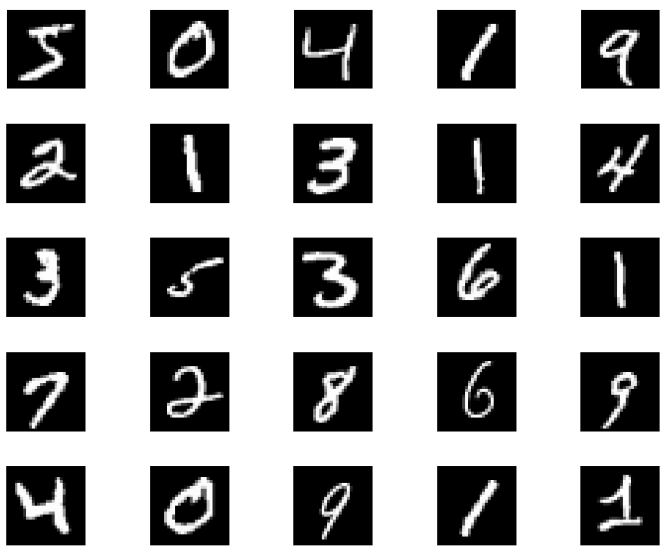
- use least squares data-fitting to find parameters $\theta_1, \ldots, \theta_n$
- take the sign of $\tilde{f}(x)$ to get the *Boolean classifier:*

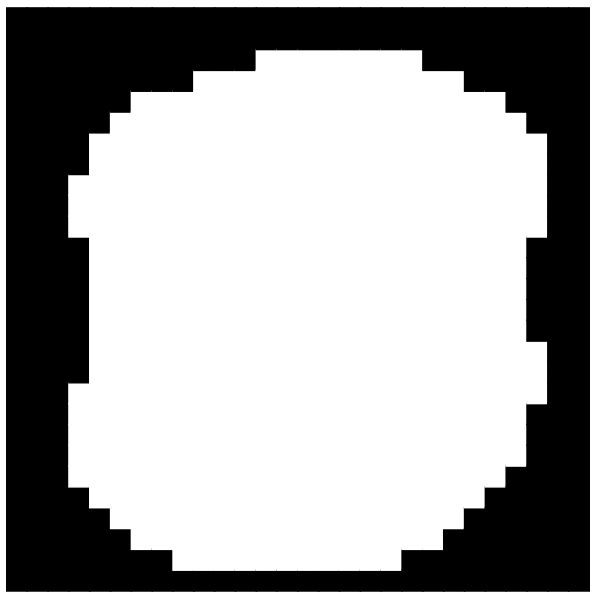
$$\hat{f}(x) = \mathrm{sign}(\tilde{f}(x)) = \begin{cases} +1 & \text{if } \tilde{f}(x) \geq 0 \\ -1 & \text{if } \tilde{f}(x) < 0 \end{cases}$$

- often used with regression model $\tilde{f}(x) = x^T\beta + v$

Handwritten digits example

- MNIST data set of 70000, $28 \times 28 = 784$ images of digits $0, \ldots, 9$
- divided into training set (60000) and test set (10000)
- only 493 nonzero pixels in at least 600 examples are used (shown on right)
- $y = +1$ if digit is 0, $y = -1$ otherwise





Least squares classifier results

results for regression model $\hat{f}(x^{(i)}) = \mathrm{sign}(\tilde{f}(x^{(i)})) = \mathrm{sign}((x^{(i)})^T\hat{\beta} + \hat{v})$

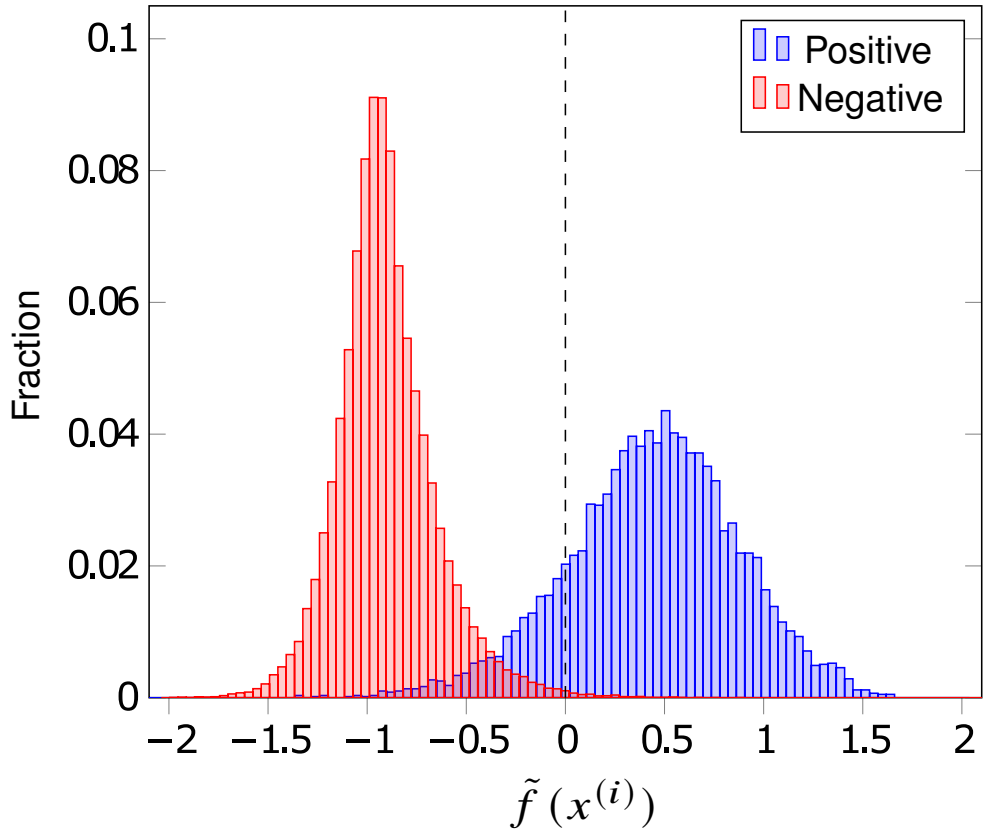
training set results (error rate 1.6%)

	$\hat{y} = +1$	$\hat{y} = -1$	Total
$y = +1$	5158	765	5923
$y = -1$	167	53910	54077
All	5325	54675	60000

test set results (error rate 1.6%)

	$\hat{y} = +1$	$\hat{y} = -1$	Total
$y = +1$	864	116	980
$y = -1$	42	8978	9020
All	906	9094	10000

Distribution of least squares fit



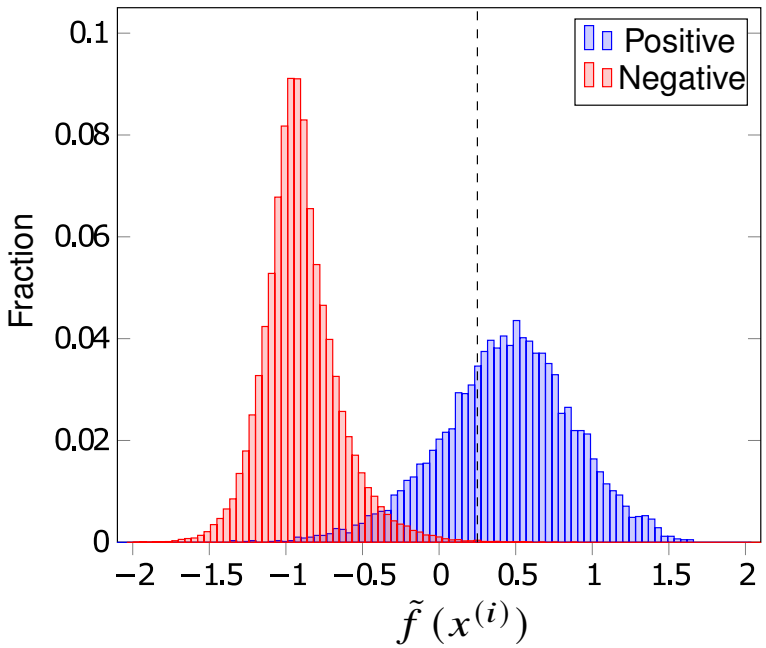
- distribution of values of $\tilde{f}(x^{(i)}) = (x^{(i)})^T\hat{\beta} + \hat{v}$ over training set
- blue bars to the left of dashed line are false negatives (misclassified digits zero)
- red bars to the right of dashed line are false positives (misclassified non-zeros)

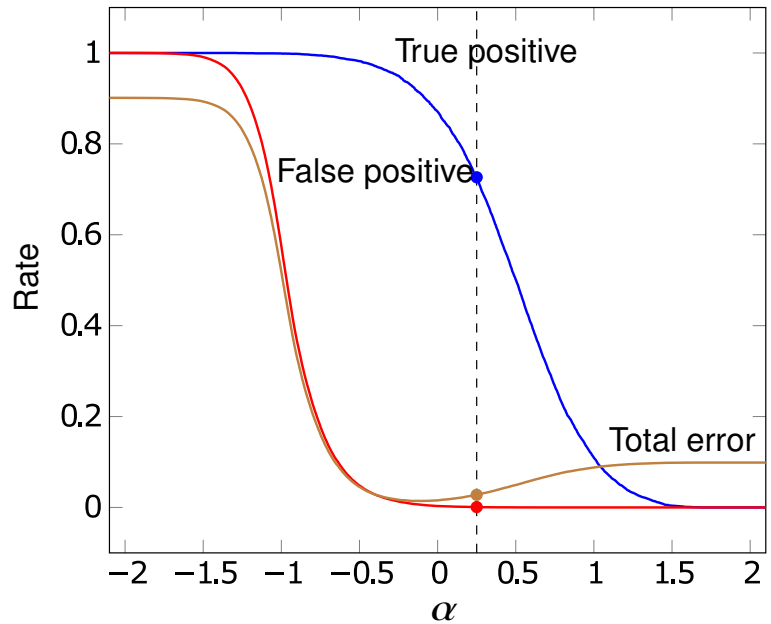
Skewed decision threshold

$$\hat{f}(x) = \text{sign}(\tilde{f}(x) - \alpha) = \begin{cases} +1 & \tilde{f}(x) \geq \alpha \\ -1 & \tilde{f}(x) < \alpha \end{cases}$$

- α is the decision threshold
- for positive α, false positive rate is lower but so is true positive rate
- for negative α, false positive rate is higher but so is true positive rate

Example (error rate 1.4% with $\alpha = -0.1$, dashed line $\alpha = 0.25$)





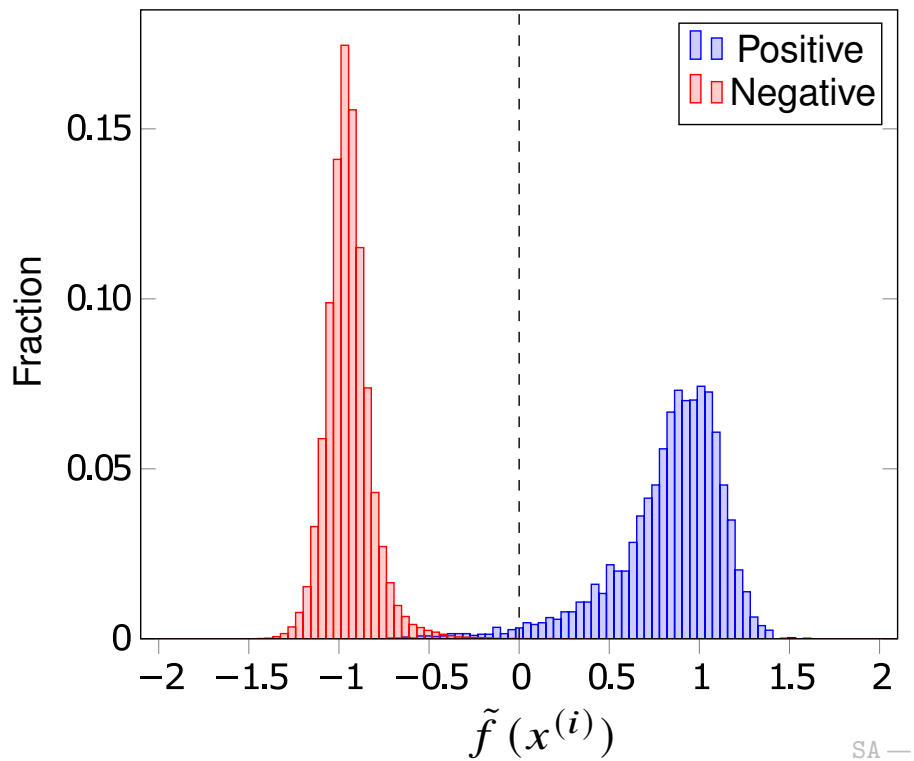
Classifier with additional nonlinear features

$$\hat{f}(x) = \mathrm{sign}(\tilde{f}(x)) = \mathrm{sign}\Big(\sum_{i=1}^{p} \theta_i f_i(x)\Big)$$

- basis functions include constant, 493 elements of x, plus 5000 functions

  $f_i(x) = \max\{0, r_i^T x + s_i\}$ with randomly generated r_i, s_i

- error rate is 0.21% on training test and 0.24% on test set



Multi-class classifiers

- a data fitting problem where the outcome $y \in \{1, \ldots, K\}$
- values of y represent K labels or categories
- multi-class classifier $\hat{y} = \hat{f}(x)$ maps x to an element of $\{1, 2, \ldots, K\}$

Least squares multi-class classifier

- for $k = 1, \ldots, K$, compute Boolean classifier to distinguish class k from not k

$$\hat{f}_k(x) = \mathrm{sign}(\tilde{f}_k(x))$$

- define multi-class classifier as

$$\hat{f}(x) = \operatorname*{argmax}_{k=1,\ldots,K} \tilde{f}_k(x)$$

  i.e., choose k with largest value of $\tilde{f}_k(x)$

Example: handwritten digit classification

- compute least squares Boolean classifier for each digit versus the rest

$$\hat{f}_k(x) = \text{sign}(x^T\beta_k + v_k), \quad k = 1, \ldots, K$$

- table shows results for test set (error rate 13.9%)

	Prediction										
Digit	0	1	2	3	4	5	6	7	8	9	Total
0	944	0	1	2	2	8	13	2	7	1	980
1	0	1107	2	2	3	1	5	1	14	0	1135
2	18	54	815	26	16	0	38	22	39	4	1032
3	4	18	22	884	5	16	10	22	20	9	1010
4	0	22	6	0	883	3	9	1	12	46	982
5	24	19	3	74	24	656	24	13	38	17	892
6	17	9	10	0	22	17	876	0	7	0	958
7	5	43	14	6	25	1	1	883	1	49	1028
8	14	48	11	31	26	40	17	13	756	18	974
9	16	10	3	17	80	0	1	75	4	803	1009
All	1042	1330	887	1042	1086	742	994	1032	898	947	10000

Example: handwritten digit classification

- ten least squares Boolean classifiers use 5000 new features
- table shows results for test set (error rate 2.6%)

	Prediction										
Digit	0	1	2	3	4	5	6	7	8	9	Total
0	972	0	0	2	0	1	1	1	3	0	980
1	0	1126	3	1	1	0	3	0	1	0	1135
2	6	0	998	3	2	0	4	7	11	1	1032
3	0	0	3	977	0	13	0	5	8	4	1010
4	2	1	3	0	953	0	6	3	1	13	982
5	2	0	1	5	0	875	5	0	3	1	892
6	8	3	0	0	4	6	933	0	4	0	958
7	0	8	12	0	2	0	1	992	3	10	1028
8	3	1	3	6	4	3	2	2	946	4	974
9	4	3	1	12	11	7	1	3	3	964	1009
All	997	1142	1024	1006	977	905	956	1013	983	997	10000

References and further readings

- S. Boyd and L. Vandenberghe. *Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares,* Cambridge University Press, 2018.

- L. Vandenberghe. *EE133A lecture notes,* Univ. of California, Los Angeles. (`http://www.seas.ucla.edu/~vandenbe/ee133a.html`)