12. Least squares

Outline

Least squares problem

Solution of least squares problem

Examples

Least squares problem

- suppose $m \times n$ matrix A is tall, so $Ax = b$ is over-determined
- for most choices of b, there is no x that satisfies $Ax = b$
- *residual* is $r = Ax - b$
- *least squares problem*: choose x to minimize $\|Ax - b\|^2$
- $\|Ax - b\|^2$ is the *objective function*
- $\hat{x}$ is a *solution* of least squares problem if

$$\|A\hat{x} - b\|^2 \leq \|Ax - b\|^2$$

  for any n-vector x
- idea: $\hat{x}$ makes residual as small as possible, if not 0
- also called *regression* (in data fitting context)

Least squares problem

- $\hat{x}$ called *least squares approximate solution* of $Ax = b$
- $\hat{x}$ is sometimes called 'solution of $Ax = b$ in the least squares sense'
  - this is very confusing
  - never say this
  - do not associate with people who say this

- $\hat{x}$ need not (and usually does not) satisfy $A\hat{x} = b$
- but if $\hat{x}$ does satisfy $A\hat{x} = b$, then it solves least squares problem

Column interpretation

- suppose $a_1, \ldots, a_n$ are columns of A
- then

$$\|Ax - b\|^2 = \|(x_1 a_1 + \cdots + x_n a_n) - b\|^2$$

- so least squares problem is to find a linear combination of columns of A that is closest to b
- if $\hat{x}$ is a solution of least squares problem, the m-vector

$$A\hat{x} = \hat{x}_1 a_1 + \cdots + \hat{x}_n a_n$$

  is closest to b among all linear combinations of columns of A

Row interpretation

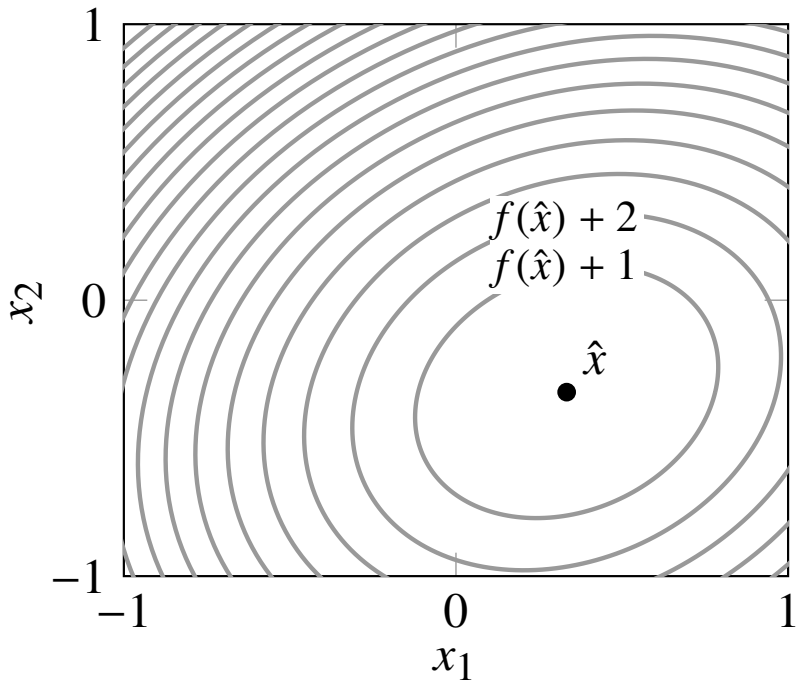
- suppose $\tilde{a}_1^T, \ldots, \tilde{a}_m^T$ are rows of A
- residual components are $r_i = \tilde{a}_i^T x - b_i$
- least squares objective is

$$\|Ax - b\|^2 = (\tilde{a}_1^T x - b_1)^2 + \cdots + (\tilde{a}_m^T x - b_m)^2$$

  the sum of squares of the residuals
- so least squares minimizes sum of squares of residuals
  - solving $Ax = b$ is making all residuals zero
  - least squares attempts to make them all small

Example

$$A = \begin{bmatrix} 2 & 0 \\ -1 & 1 \\ 0 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$



- $Ax = b$ has no solution
- least squares problem is to choose x to minimize

$$\|Ax - b\|^2 = (2x_1 - 1)^2 + (-x_1 + x_2)^2 + (2x_2 + 1)^2$$

- least squares approximate solution is $\hat{x} = (1/3, -1/3)$ (say, via calculus)
- $\|A\hat{x} - b\|^2 = 2/3$ is smallest posible value of $\|Ax - b\|^2$
- $A\hat{x} = (2/3, -2/3, -2/3)$ is linear combination of columns of A closest to b

Outline

Least squares problem

Solution of least squares problem

Examples

Solution of least squares problem

- we make one assumption: A *has linearly independent columns*
- this implies that Gram matrix $A^T A$ is invertible
- unique solution of least squares problem is

$$\hat{x} = (A^T A)^{-1} A^T b = A^\dagger b$$

- cf. $x = A^{-1} b$, solution of square invertible system $Ax = b$

Derivation via calculus

- define

$$f(x) = \|Ax - b\|^2 = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} A_{ij} x_j - b_i \right)^2$$

- solution $\hat{x}$ satisfies

$$\frac{\partial f}{\partial x_k}(\hat{x}) = \nabla f(\hat{x})_k = 0, \quad k = 1, \ldots, n$$

- taking partial derivatives we get $\nabla f(x)_k = \left(2A^T(Ax - b)\right)_k$
- in matrix-vector notation: $\nabla f(\hat{x}) = 2A^T(A\hat{x} - b) = 0$
- so $\hat{x}$ satisfies *normal equations* $(A^T A)\hat{x} = A^T b$
- and therefore $\hat{x} = (A^T A)^{-1} A^T b$

Direct verification

- let $\hat{x} = (A^T A)^{-1} A^T b$, so $A^T(A\hat{x} - b) = 0$
- for any n-vector x we have

$$\begin{aligned} \|Ax - b\|^2 &= \|(Ax - A\hat{x}) + (A\hat{x} - b)\|^2 \\ &= \|A(x - \hat{x})\|^2 + \|A\hat{x} - b\|^2 + 2(A(x - \hat{x}))^T(A\hat{x} - b) \\ &= \|A(x - \hat{x})\|^2 + \|A\hat{x} - b\|^2 + 2(x - \hat{x})^T A^T(A\hat{x} - b) \\ &= \|A(x - \hat{x})\|^2 + \|A\hat{x} - b\|^2 \end{aligned}$$

- so for any x, $\|Ax - b\|^2 \geq \|A\hat{x} - b\|^2$
- if equality holds, $A(x - \hat{x}) = 0$, which implies $x = \hat{x}$ since columns of A are linearly independent

Computing least squares approximate solutions

- compute QR factorization of A: $A = QR$ ($2mn^2$ flops)
- QR factorization exists since columns of A are linearly independent
- to compute $\hat{x} = A^{\dagger} b = R^{-1} Q^T b$
  - form $Q^T b$ ($2mn$ flops)
  - compute $\hat{x} = R^{-1}(Q^T b)$ via back substitution (n^2 flops)
- total complexity $2mn^2$ flops

- identical to algorithm for solving $Ax = b$ for square invertible A
- but when A is tall, gives least squares approximate solution

Outline

Least squares problem

Solution of least squares problem

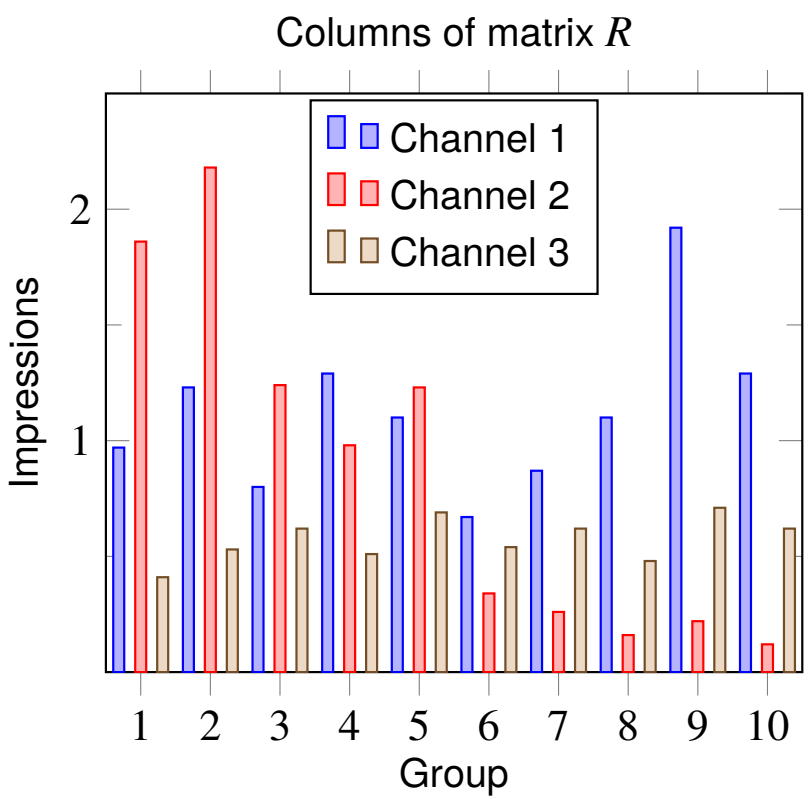
Examples

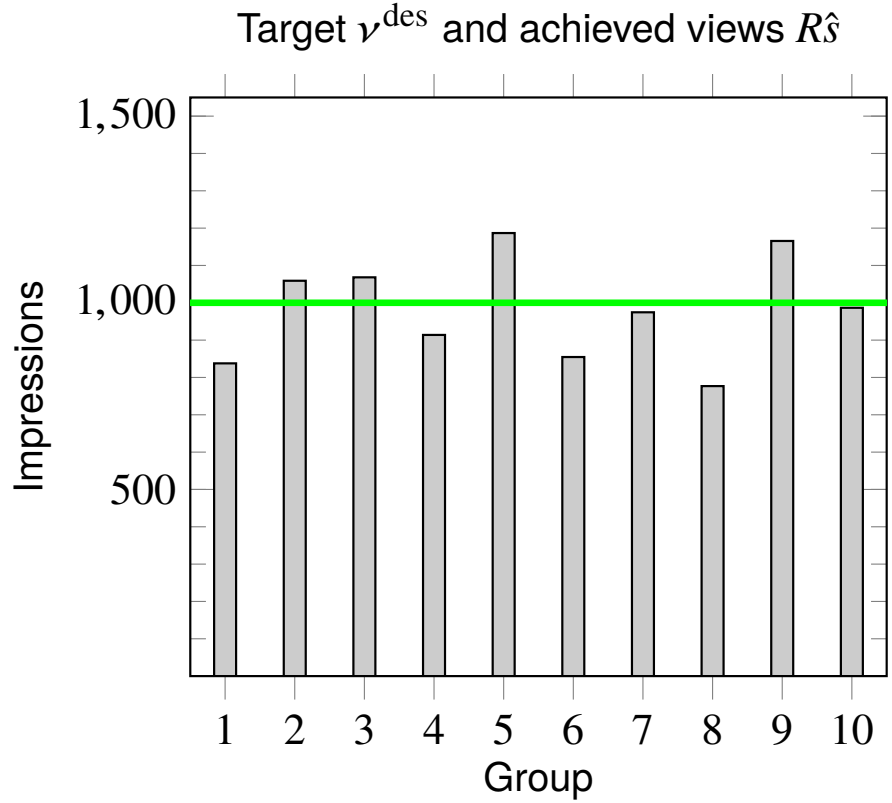
Advertising purchases

- m demographics groups we want to advertise to
- v^{des} is m-vector of target views or impressions
- n-vector s gives spending on n advertising channels
- $m \times n$ matrix R gives demographic reach of channels
- R_{ij} is number of views per dollar spent (in 1000/$)
- $v = Rs$ is m-vector of views across demographic groups
- $\|v^{\text{des}} - Rs\| / \sqrt{m}$ is RMS deviation from desired views
- we'll use least squares spending $\hat{s} = R^{\dagger} v^{\text{des}}$ (need not be ≥ 0)

Example

- $m = 10$ groups, $n = 3$ channels
- target views vector $v^{\text{des}} = 10^3 \times \mathbf{1}$
- optimal spending is $\hat{s} = (62, 100, 1443)$





Illumination

- n lamps illuminate an area divided in m regions
- A_{ij} is illumination in region i if lamp j is on with power 1, other lamps are off
- x_j is power of lamp j
- $(Ax)_i$ is illumination level at region i
- b_i is target illumination level at region i

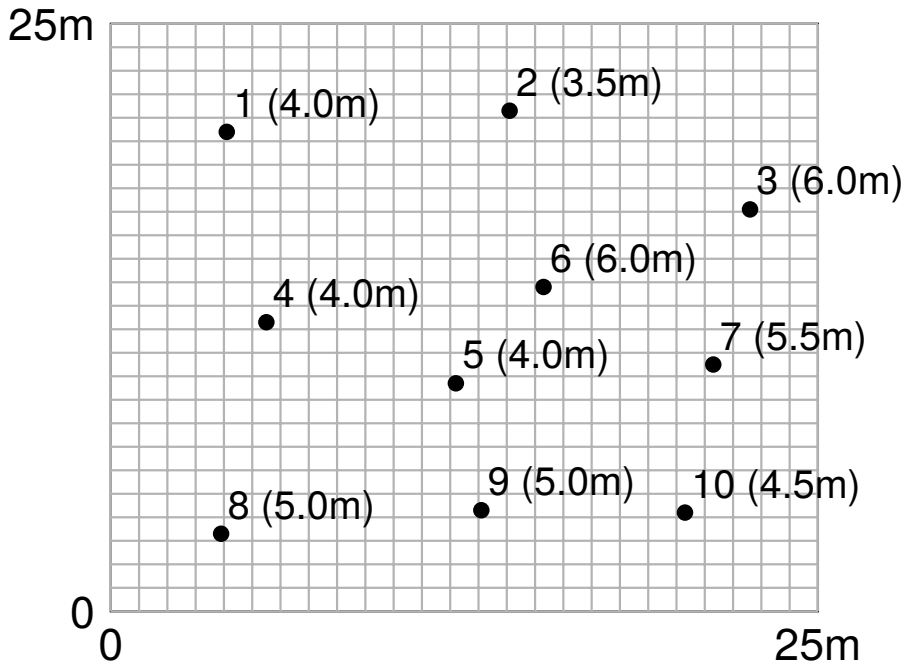
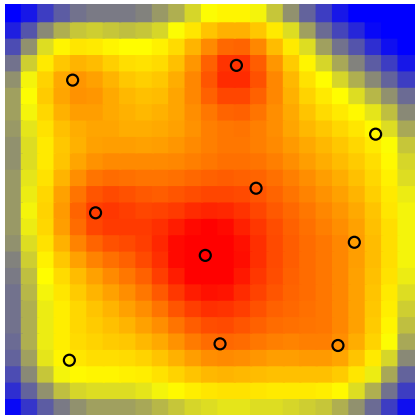


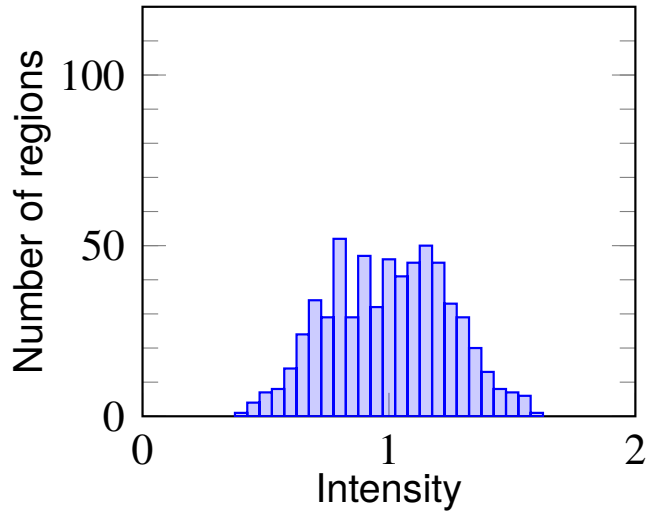
figure shows lamp positions for example with

$$m = 25^2, \quad n = 10$$

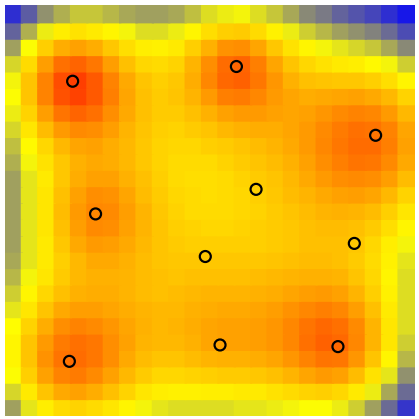
Illumination

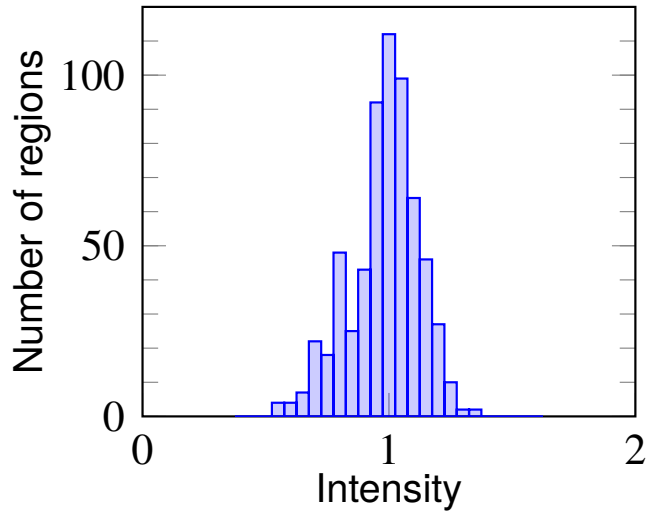
- equal lamp powers ($x = \mathbf{1}$)





- least squares solution $\hat{x}$, with $b = \mathbf{1}$





13. Least squares data fitting

Outline

Least squares model fitting

Validation

Feature engineering

Setup

- we believe a scalar y and an n-vector x are related by *model*

$$y \approx f(x)$$

- x is called the *independent variable*
- y is called the *outcome* or *response variable*
- $f : \mathbf{R}^n \to \mathbf{R}$ gives the relation between x and y
- often x is a feature vector, and y is something we want to predict
- we don't know f, which gives the 'true' relationship between x and y

Data

- we are given some *data*

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

  also called *observations*, *examples*, *samples*, or *measurements*

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

Model

- choose *model* $\hat{f} : \mathbf{R}^n \to \mathbf{R}$, a *guess* or *approximation* of f
- *linear in the parameters* model form:

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

- $f_i : \mathbf{R}^n \to \mathbf{R}$ are *basis functions* that we choose
- θ_i are *model parameters* that we choose
- $\hat{y}^{(i)} = \hat{f}(x^{(i)})$ is (the model's) *prediction* of $y^{(i)}$
- we'd like $\hat{y}^{(i)} \approx y^{(i)}$, *i.e.*, model is consistent with observed data

Least squares data fitting

- *prediction error* or *residual* is $r_i = y^{(i)} - \hat{y}^{(i)}$
- *least squares data fitting*: choose model parameters θ_i to minimize RMS prediction error on data set

$$\left(\frac{(r^{(1)})^2 + \cdots + (r^{(N)})^2}{N} \right)^{1/2}$$

- this can be formulated (and solved) as a least squares problem

Least squares data fitting

- express $y^{(i)}$, $\hat{y}^{(i)}$, and $r^{(i)}$ as N-vectors
  - $y^{\mathrm{d}} = (y^{(1)}, \ldots, y^{(N)})$ is vector of outcomes
  - $\hat{y}^{\mathrm{d}} = (\hat{y}^{(1)}, \ldots, \hat{y}^{(N)})$ is vector of predictions
  - $r^{\mathrm{d}} = (r^{(1)}, \ldots, r^{(N)})$ is vector of residuals
- $\mathbf{rms}(r^{\mathrm{d}})$ is *RMS prediction error*
- define $N \times p$ matrix A with elements $A_{ij} = f_j(x^{(i)})$, so $\hat{y}^{\mathrm{d}} = A\theta$
- least squares data fitting: choose θ to minimize

$$\|r^{\mathrm{d}}\|^2 = \|y^{\mathrm{d}} - \hat{y}^{\mathrm{d}}\|^2 = \|y^{\mathrm{d}} - A\theta\|^2 = \|A\theta - y^{\mathrm{d}}\|^2$$

- $\hat{\theta} = (A^T A)^{-1} A^T y$ (if columns of A are linearly independent)
- $\|A\hat{\theta} - y\|^2/N$ is *minimum mean-square (fitting) error*

Fitting a constant model

- simplest possible model: $p = 1, f_1(x) = 1$, so model $\hat{f}(x) = \theta_1$ is a constant
- $A = \mathbf{1}$, so

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

- the mean of $y^{(1)}, \ldots, y^{(N)}$ is the least squares fit by a constant
- MMSE is $\mathbf{std}(y^{\mathrm{d}})^2$; RMS error is $\mathbf{std}(y^{\mathrm{d}})$
- more sophisticated models are judged against the constant model

Fitting univariate functions

- when $n = 1$, we seek to approximate a function $f : \mathbf{R} \to \mathbf{R}$
- we can plot the data (x_i, y_i) and the model function $\hat{y} = \hat{f}(x)$

Straight-line fit

- $p = 2$, with $f_1(x) = 1, f_2(x) = x$
- model has form $\hat{f}(x) = \theta_1 + \theta_2 x$
- matrix A has form

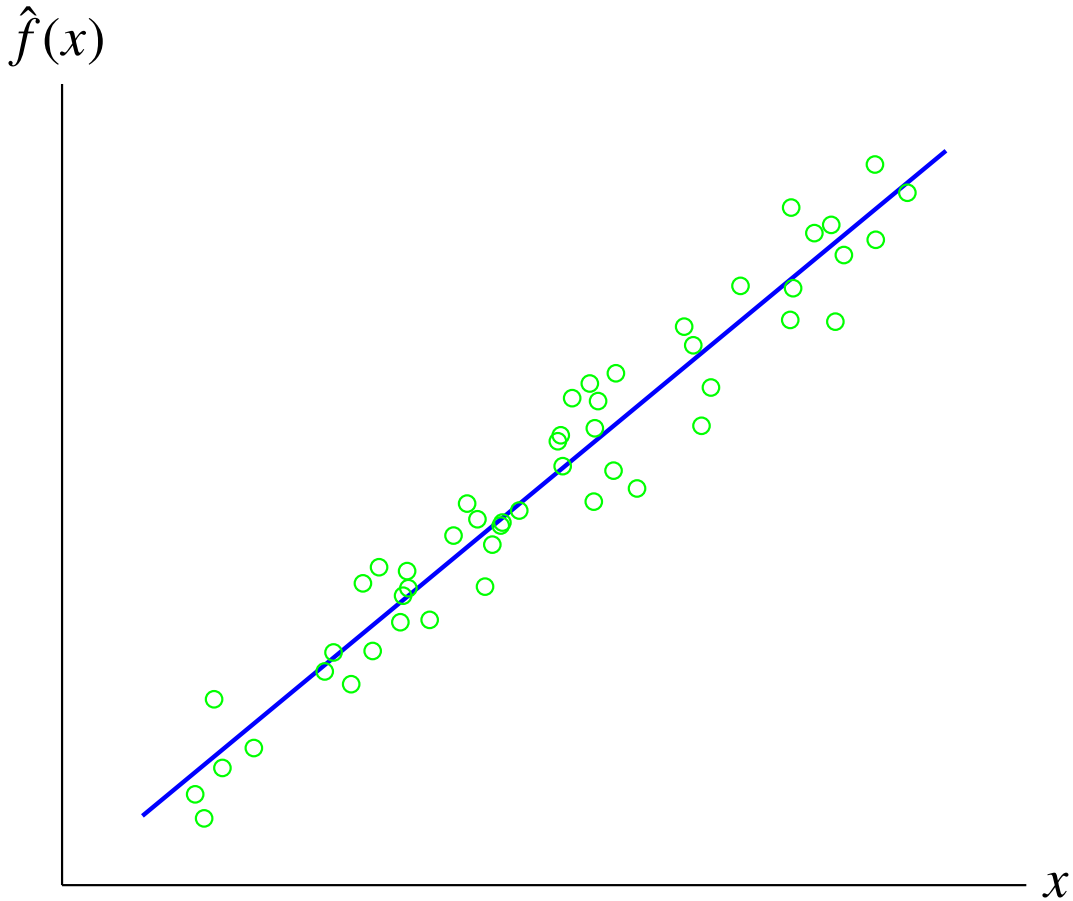
$$A = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ \vdots & \vdots \\ 1 & x^{(N)} \end{bmatrix}$$

- can work out $\hat{\theta}_1$ and $\hat{\theta}_2$ explicitly:

$$\hat{f}(x) = \mathbf{avg}(y^{\mathrm{d}}) + \rho \frac{\mathbf{std}(y^{\mathrm{d}})}{\mathbf{std}(x^{\mathrm{d}})}(x - \mathbf{avg}(x^{\mathrm{d}}))$$

where $x^{\mathrm{d}} = (x^{(1)}, \ldots, x^{(N)})$

Example



Asset α and β

- x is return of whole market, y is return of a particular asset
- write straight-line model as

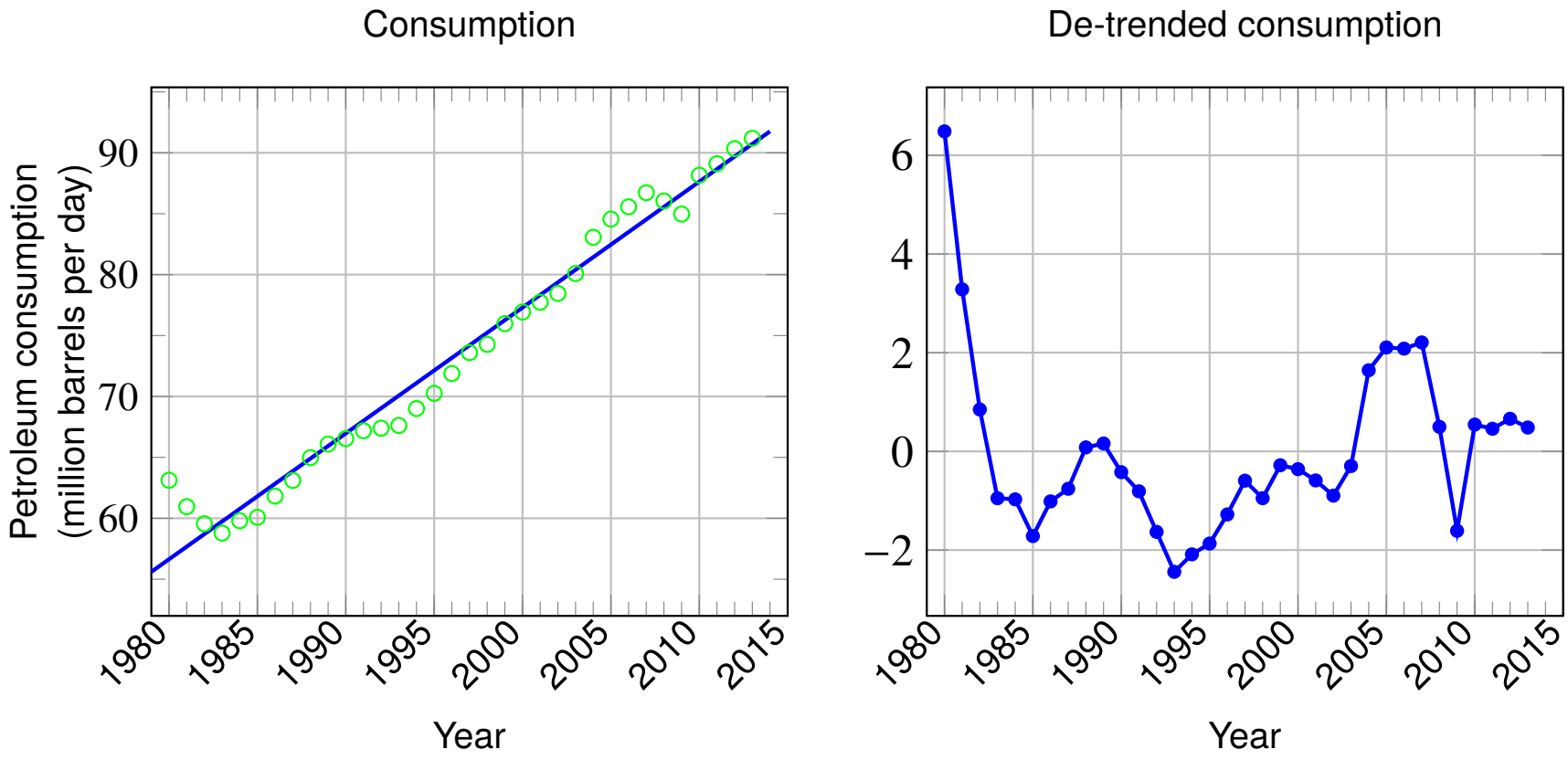
$$\hat{y} = (r^{\mathrm{rf}} + \alpha) + \beta(x - \mu^{\mathrm{mkt}})$$

  - μ^{mkt} is the average market return
  - r^{rf} is the risk-free interest rate
  - several other slightly different definitions are used
- called asset 'α' and 'β', widely used

Time series trend

- $y^{(i)}$ is value of quantity at time $x^{(i)} = i$
- $\hat{y}^{(i)} = \hat{\theta}_1 + \hat{\theta}_2 i, \quad i = 1, \ldots, N$, is called *trend line*
- $y^{\mathrm{d}} - \hat{y}^{\mathrm{d}}$ is called *de-trended time series*
- $\hat{\theta}_2$ is *trend coefficient*

World petroleum consumption



Polynomial fit

- $f_i(x) = x^{i-1}, \quad i = 1, \ldots, p$
- model is a polynomial of degree less than p

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

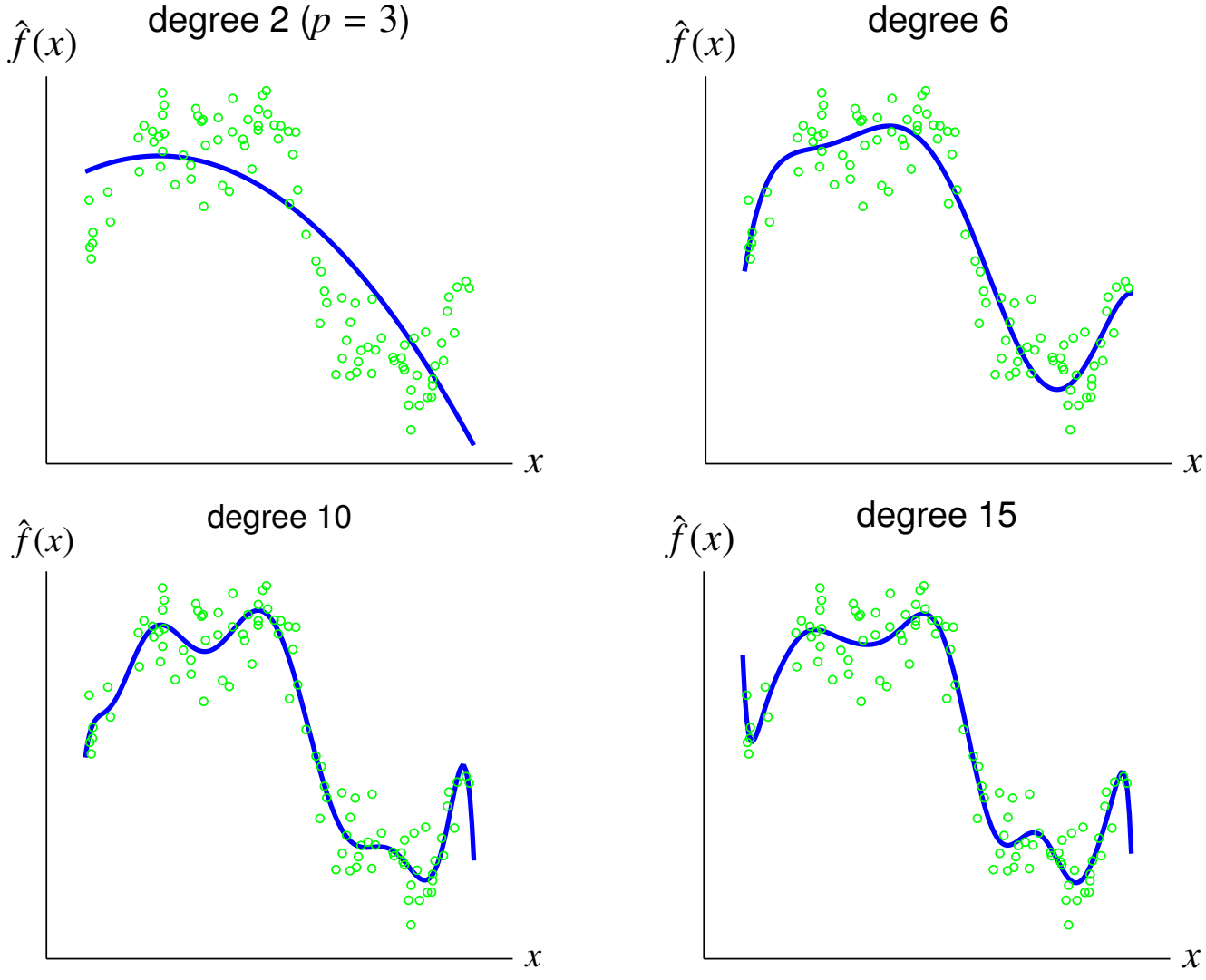
(here x^i means scalar x to ith power; $x^{(i)}$ is ith data point)

- 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



Regression as general data fitting

- regression model is affine function $\hat{y} = \hat{f}(x) = x^T\beta + v$
- fits general fitting form with basis functions

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

so model is

$$\hat{y} = \theta_1 + \theta_2 x_1 + \cdots + \theta_{n+1} x_n = x^T \theta_{2:n} + \theta_1$$

- $\beta = \theta_{2:n+1}$, $v = \theta_1$

General data fitting as regression

- general fitting model $\hat{f}(x) = \theta_1 f_1(x) + \cdots + \theta_p f_p(x)$
- common assumption: $f_1(x) = 1$
- same as regression model $\hat{f}(\tilde{x}) = \tilde{x}^T \beta + v$, with
  - $\tilde{x} = (f_2(x), \ldots, f_p(x))$ are 'transformed features'
  - $v = \theta_1$, $\beta = \theta_{2:p}$

Auto-regressive time series model

- time zeries $z_1, z_2, \ldots$
- *auto-regressive* (AR) prediction model:

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

- M is *memory* of model
- $\hat{z}_{t+1}$ is prediction of next value, based on previous M values
- we'll choose β to minimize sum of squares of prediction errors,

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

- put in general form with

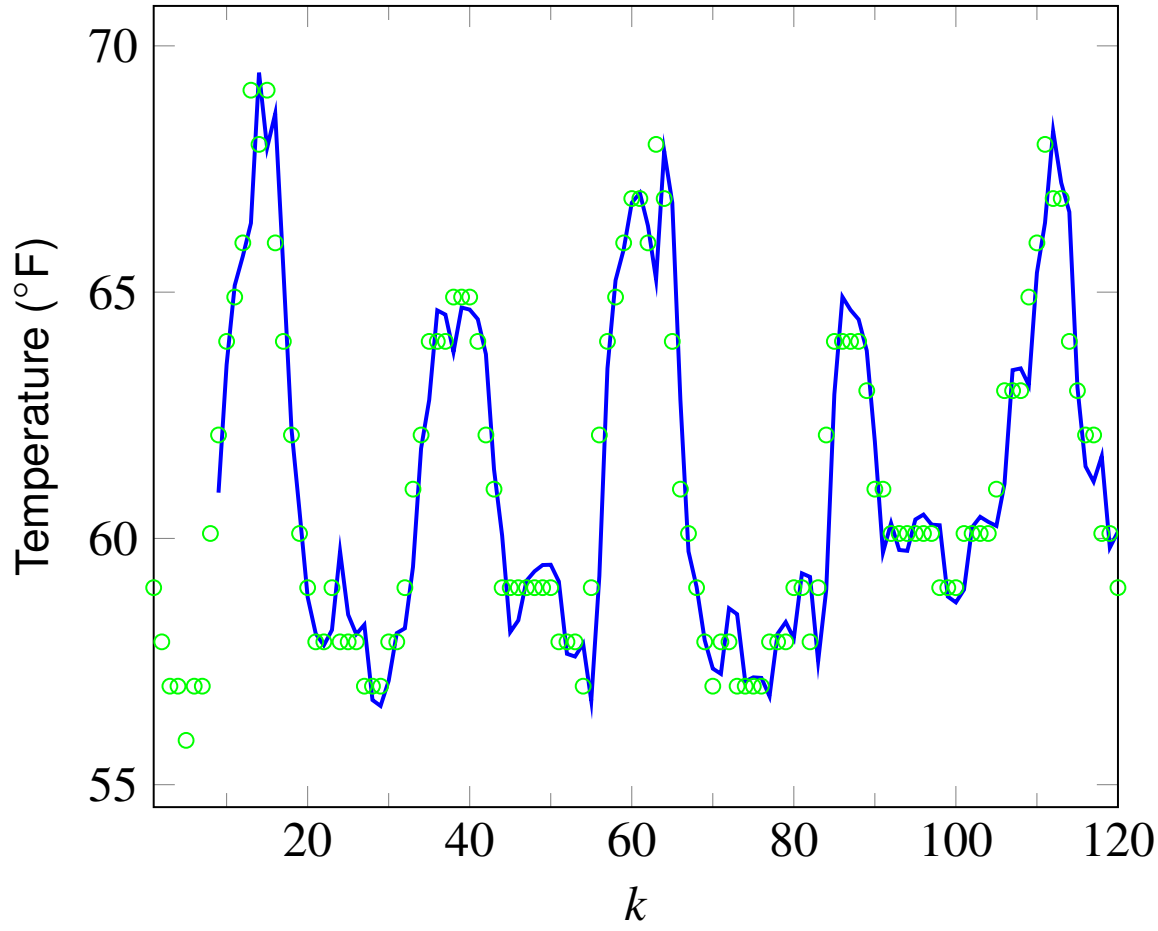
$$y^{(i)} = z_{M+i}, \quad x^{(i)} = (z_{M+i-1}, \ldots, z_i), \quad i = 1, \ldots, T-M$$

Example

- hourly temperature at LAX in May 2016, length 744
- average is $61.76°$F, standard deviation $3.05°$F
- predictor $\hat{z}_{t+1} = z_t$ gives RMS error $1.16°$F
- predictor $\hat{z}_{t+1} = z_{t-23}$ gives RMS error $1.73°$F
- AR model with $M = 8$ gives RMS error $0.98°$F

Example

solid line shows one-hour ahead predictions from AR model, first 5 days



Outline

Least squares model fitting

Validation

Feature engineering

Generalization

basic idea:

- goal of model is *not* to predict outcome for the given data
- instead it is to *predict the outcome on new, unseen data*

- a model that makes reasonable predictions on new, unseen data has *generalization ability*, or *generalizes*
- a model that makes poor predictions on new, unseen data is said to suffer from *over-fit*

Validation

a simple and effective method to guess if a model will generalize

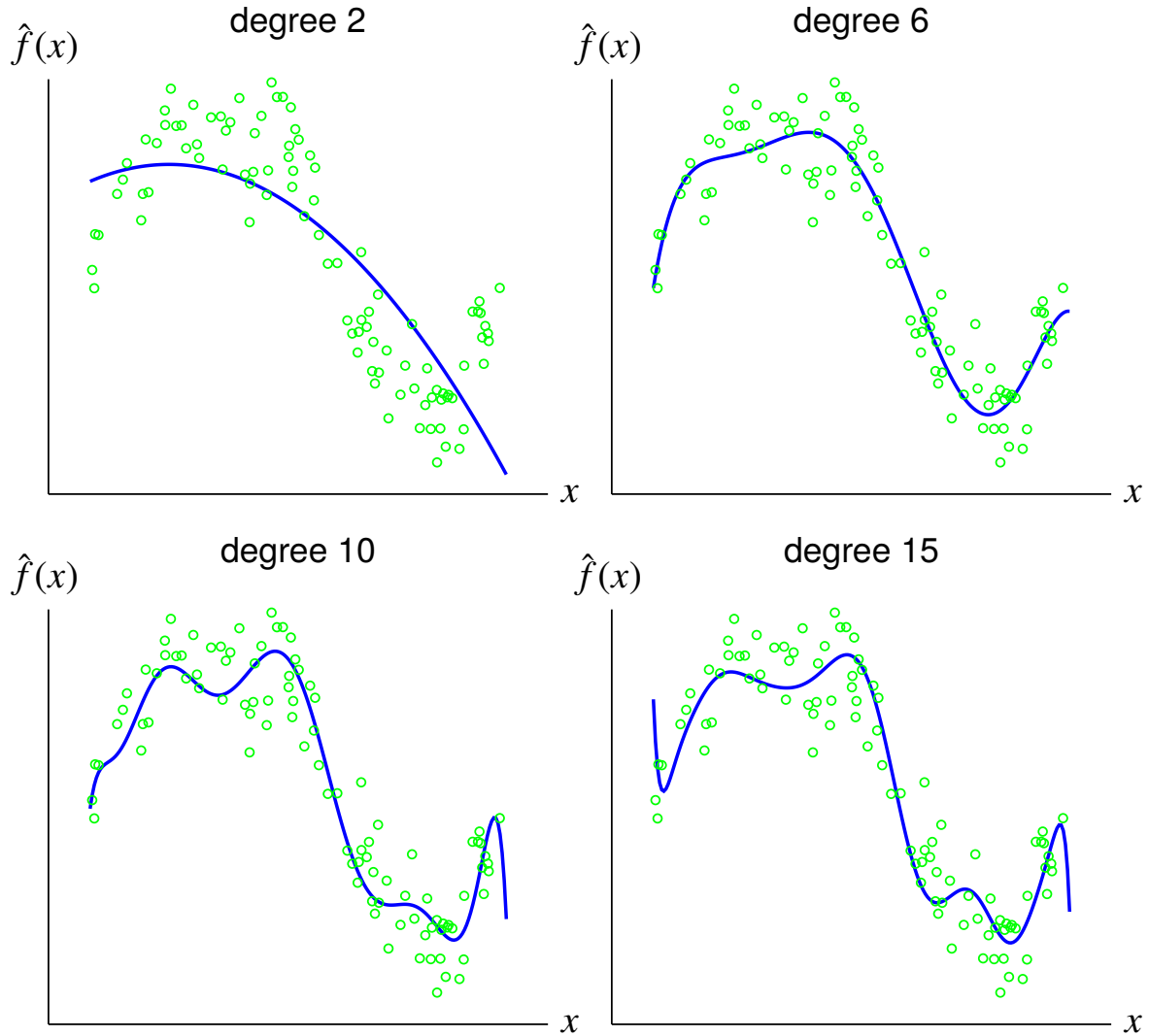
- split original data into a *training set* and a *test set*
- typical splits: 80%/20%, 90%/10%
- build ('train') model on training data set
- then *check the model's predictions on the test data set*
- (can also compare RMS prediction error on train and test data)
- if they are similar, we can *guess* the model will generalize

Validation

- can be used to choose among different candidate models, *e.g.*
  - polynomials of different degrees
  - regression models with different sets of regressors
- we'd use one with low, or lowest, test error

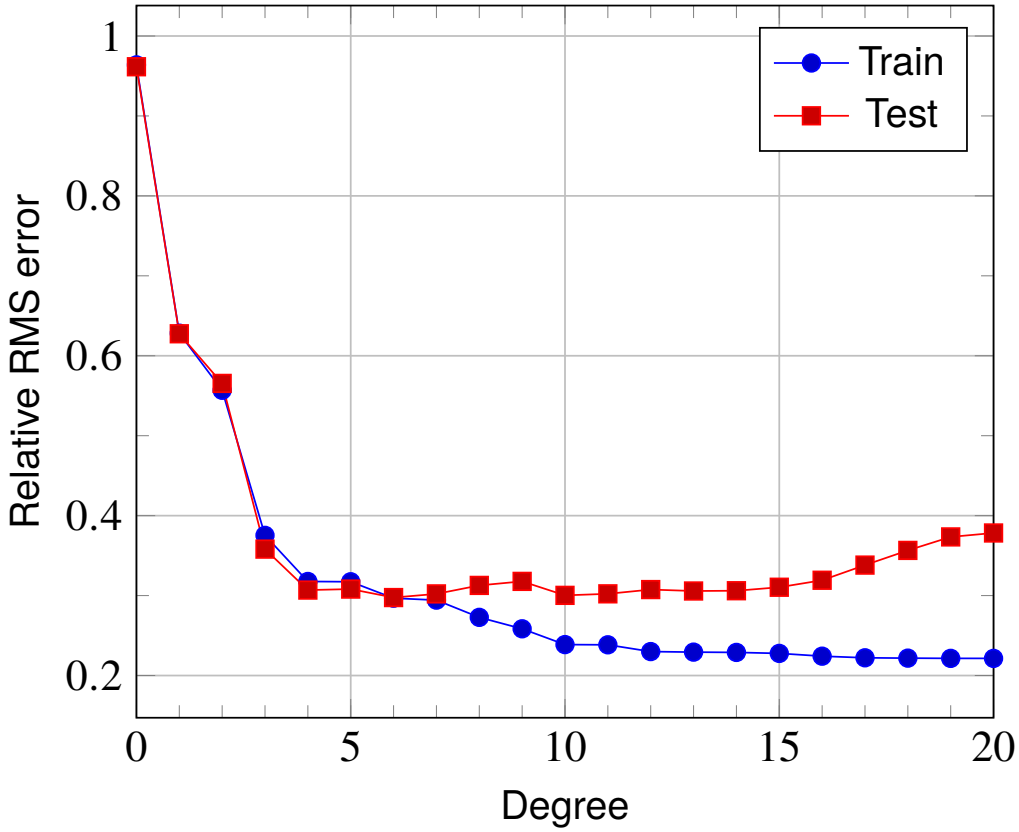
Example

models fit using *training set* of 100 points; plots show *test set* of 100 points



Example

- suggests degree 4, 5, or 6 are reasonable choices



Cross validation

to carry out cross validation:

- divide data into 10 *folds*
- for $i = 1, \ldots, 10$, build (train) model using all folds except i
- test model on data in fold i

interpreting cross validation results:

- if test RMS errors are much larger than train RMS errors, model is over-fit
- if test and train RMS errors are similar and consistent, we can *guess* the model will have a similar RMS error on future data

Example

- house price, regression fit with $x = (\text{area}/1000\ \text{ft.}^2, \text{bedrooms})$
- 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

Outline

Least squares model fitting

Validation

Feature engineering

Feature engineering

- start with original or base feature n-vector x
- choose basis functions $f_1, \ldots, f_p$ to create 'mapped' feature p-vector

$$(f_1(x), \ldots, f_p(x))$$

- now fit linear in parameters model with mapped features

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

- *check the model using validation*

Transforming features

- *standardizing features*: replace x_i with

$$(x_i - b_i)/a_i$$

  - $b_i \approx$ mean value of the feature across the data
  - $a_i \approx$ standard deviation of the feature across the data

  new features are called z*-scores*

- *log transform*: if x_i is nonnegative and spans a wide range, replace it with

$$\log(1 + x_i)$$

- *hi and lo features*: create new features given by

$$\max\{x_1 - b, 0\}, \qquad \min\{x_1 - a, 0\}$$

  (called hi and lo versions of original feature x_i)

Example

- house price prediction
- start with base features
  - x_1 is area of house (in 1000ft.^{2})
  - x_2 is number of bedrooms
  - x_3 is 1 for condo, 0 for house
  - x_4 is zip code of address (62 values)
- we'll use $p = 8$ basis functions:
  - $f_1(x) = 1, f_2(x) = x_1, f_3(x) = \max\{x_1 - 1.5, 0\}$
  - $f_4(x) = x_2, f_5(x) = x_3$
  - $f_6(x), f_7(x), f_8(x)$ are Boolean functions of x_4 which encode 4 groups of nearby zip codes (*i.e.*, neighborhood)
- five fold model validation

Example

	Model parameters								RMS error	
Fold	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	Train	Test
1	122.35	166.87	−39.27	−16.31	−23.97	−100.42	−106.66	−25.98	67.29	72.78
2	100.95	186.65	−55.80	−18.66	−14.81	−99.10	−109.62	−17.94	67.83	70.81
3	133.61	167.15	−23.62	−18.66	−14.71	−109.32	−114.41	−28.46	69.70	63.80
4	108.43	171.21	−41.25	−15.42	−17.68	−94.17	−103.63	−29.83	65.58	78.91
5	114.45	185.69	−52.71	−20.87	−23.26	−102.84	−110.46	−23.43	70.69	58.27

14. Least squares classification

Outline

Classification

Least squares classification

Multi-class classifiers

Classification

- data fitting with outcome that takes on (non-numerical) values like
  - TRUE or FALSE
  - SPAM or NOT SPAM
  - DOG, HORSE, or MOUSE
- outcome values are called *labels* or *categories*
- data fitting is called *classification*

- we start with case when there are two possible outcomes
- called *Boolean* or *2-way* classification
- we encode outcomes as $+1$ (TRUE) and -1 (FALSE)
- classifier has form $\hat{y} = \hat{f}(x), f : \mathbf{R}^n \to \{-1, +1\}$

Applications

- email spam detection
  - x contains features of an email message (word counts, . . .)
- financial transaction fraud detection
  - x contains features of proposed transaction, initiator
- document classification (say, politics or not)
  - x is word count histogram of document
- disease detection
  - x contains patient features, results of medical tests
- digital communications receiver
  - y is transmitted bit; x contain n measurements of received signal

Prediction errors

- data point (x, y), predicted outcome $\hat{y} = \hat{f}(x)$
- only four possibilities:
  - *True positive.* $y = +1$ and $\hat{y} = +1$.
  - *True negative.* $y = -1$ and $\hat{y} = -1$.

  (in these two cases, the prediction is *correct*)

  - *False positive.* $y = -1$ and $\hat{y} = +1$.
  - *False negative.* $y = +1$ and $\hat{y} = -1$.

  (in these two cases, the prediction is *wrong*)
- the errors have many other names, like Type I and Type II

Confusion matrix

- given data set $x^{(1)}, \ldots, x^{(N)}$, $y^{(1)}, \ldots, y^{(N)}$ and classifier $\hat{f}$
- count each of the four outcomes

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

- off-diagonal terms are prediction errors
- many error rates and accuracy measures are used
  - *error rate* is $(N_{\text{fp}} + N_{\text{fn}})/N$
  - *true positive* (or *recall*) rate is $N_{\text{tp}}/N_{\text{p}}$
  - *false positive rate* (or *false alarm rate*) is $N_{\text{fp}}/N_{\text{n}}$
- a proposed 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 $19/1139 = 1.67\%$

Outline

Classification

Least squares classification

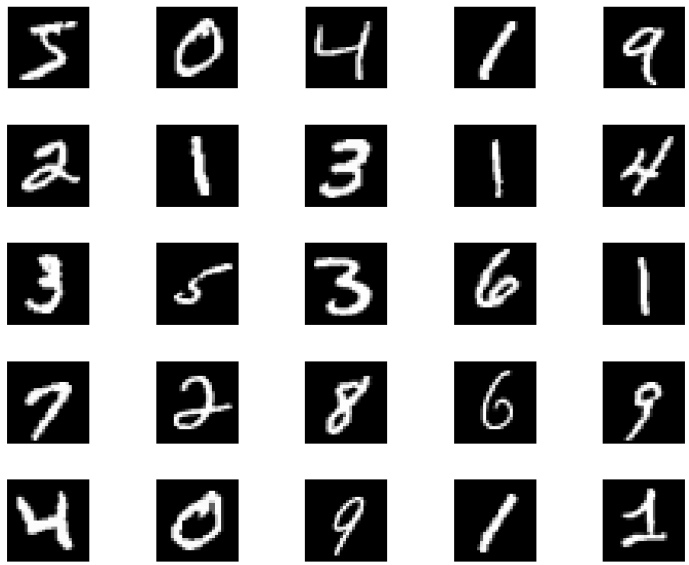
Multi-class classifiers

Least squares classification

- fit model $\tilde{f}$ to encoded (± 1) $y^{(i)}$ values *using standard least squares data fitting*
- $\tilde{f}(x)$ should be near $+1$ when $y = +1$, and near -1 when $y = -1$
- $\tilde{f}(x)$ is a *number*
- use model $\hat{f}(x) = \mathbf{sign}(\tilde{f}(x))$
- (size of $\tilde{f}(x)$ is related to the 'confidence' in the prediction)

Handwritten digits example

- MNIST data set of 70000 28×28 images of digits 0, . . . , 9



- divided into training set (60000) and test set (10000)
- x is 494-vector, constant 1 plus the 493 pixel values with nonzero values in at least 600 training examples
- $y = +1$ if digit is 0; -1 otherwise

Least squares classifier results

- 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

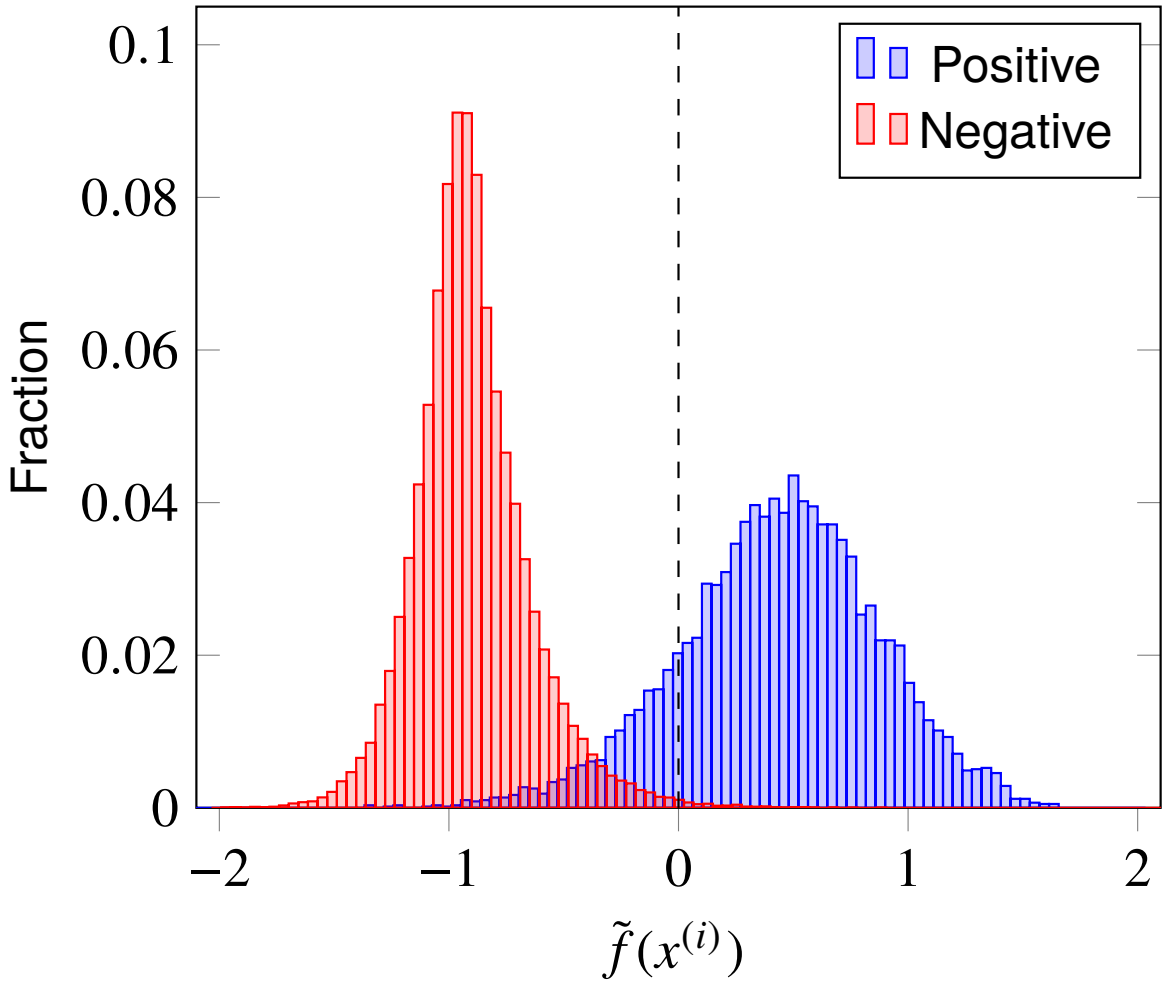
- test set results (error rate 1.6%)

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

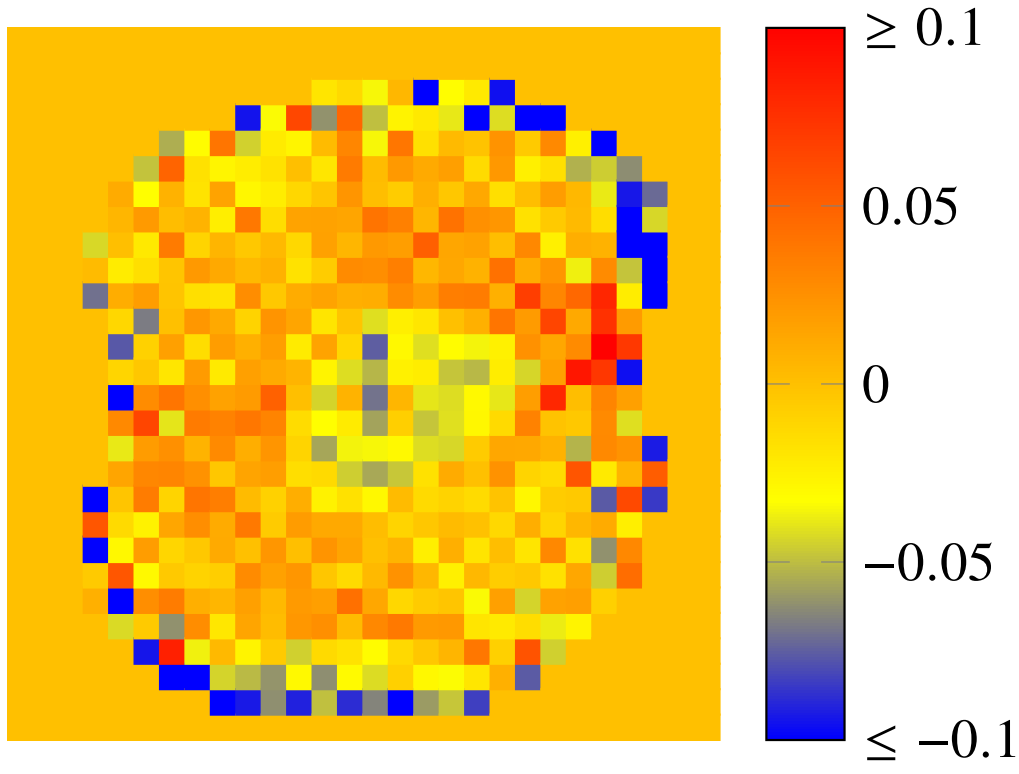
- we can likely achieve 1.6% error rate on unseen images

Distribution of least squares fit

distribution of values of $\tilde{f}(x^{(i)})$ over training set



Coefficients in least squares classifier



Skewed decision threshold

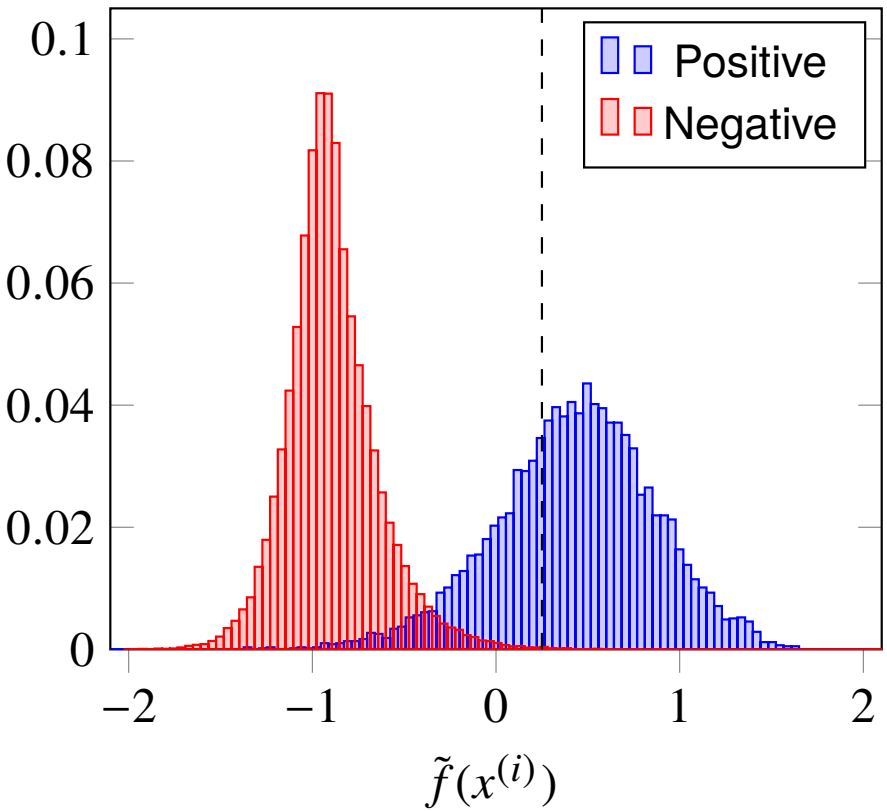
- use predictor $\hat{f}(x) = \mathbf{sign}(\tilde{f}(x) - \alpha)$, *i.e.*,

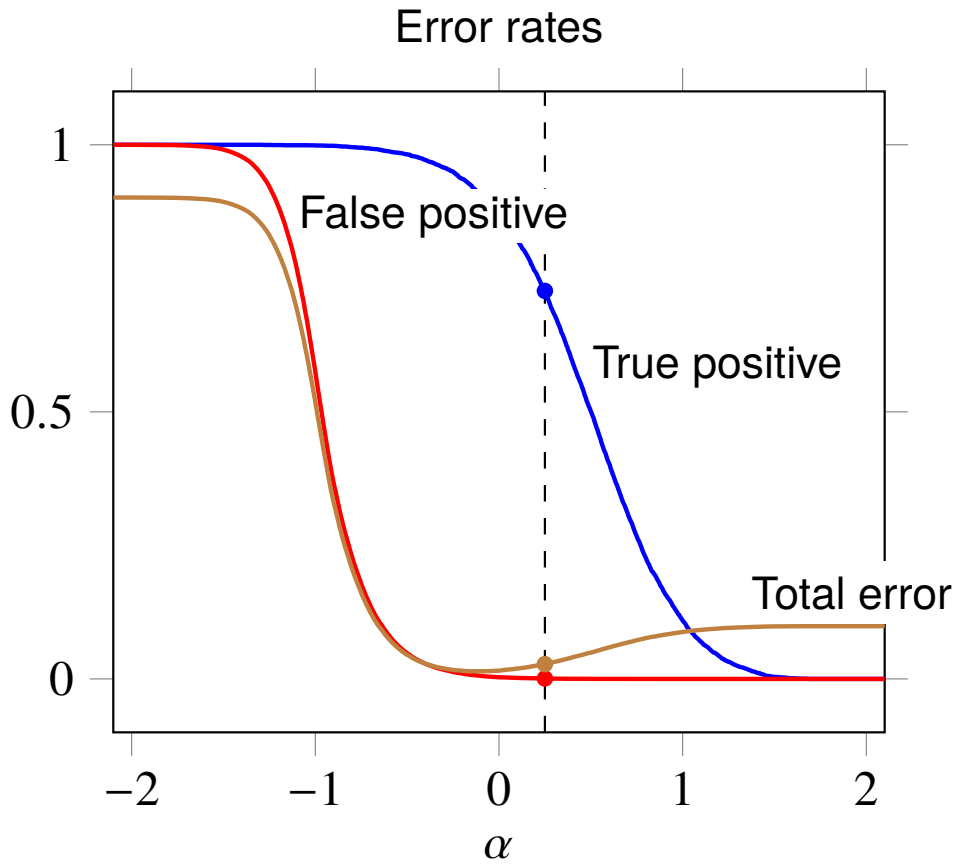
$$\hat{f}(x) = \begin{cases} +1 & \tilde{f}(x) \geq \alpha \\ -1 & \tilde{f}(x) < \alpha \end{cases}$$

- 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

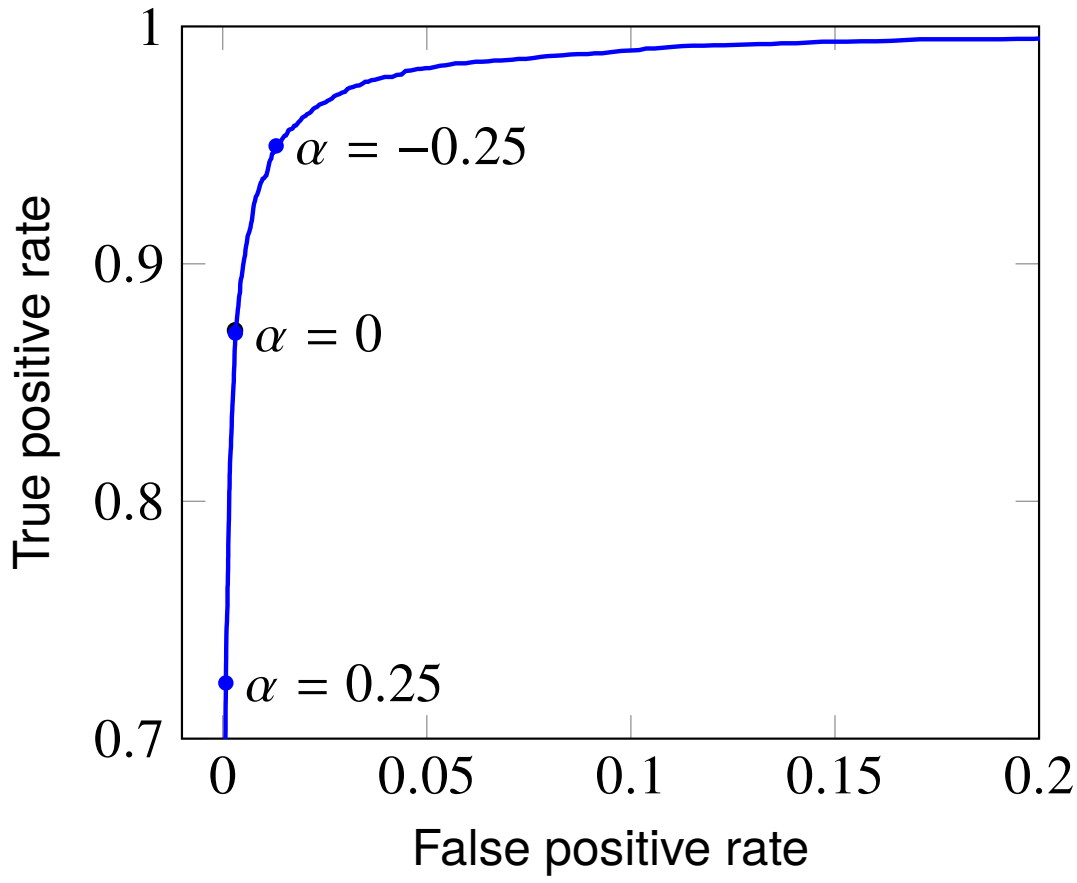
- trade off curve of true positive versus false positive rates is called *receiver operating characteristic* (ROC)

Example


0.1
0.08
0.06
0.04
0.02
0
−2
−1
0
1
2
Positive
Negative
$\tilde{f}(x^{(i)})$


Error rates
1
0.5
0
−2
−1
0
1
2
False positive
True positive
Total error
α

ROC curve


1
0.9
0.8
0.7
True positive rate
$\alpha = -0.25$
$\alpha = 0$
$\alpha = 0.25$
0
0.05
0.1
0.15
0.2
False positive rate

Outline

Classification

Least squares classification

Multi-class classifiers

Multi-class classifiers

- we have $K > 2$ possible labels, with label set $\{1, \ldots, K\}$
- predictor is $\hat{f} : \mathbf{R}^n \to \{1, \ldots, K\}$
- for given predictor and data set, confusion matrix is $K \times K$
- some off-diagonal entries may be much worse than others

Examples

- handwritten digit classification
  - guess the digit written, from the pixel values
- marketing demographic classification
  - guess the demographic group, from purchase history
- disease diagnosis
  - guess diagnosis from among a set of candidates, from test results, patient features
- translation word choice
  - choose how to translate a word into several choices, given context features
- document topic prediction
  - guess topic from word count histogram

Least squares multi-class classifier

- create a least squares classifier for each label versus the others
- take as classifier

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

  (*i.e.*, choose ℓ with largest value of $\tilde{f}_\ell(x)$)

- for example, with

$$\tilde{f}_1(x) = -0.7, \qquad \tilde{f}_2(x) = +0.2, \qquad \tilde{f}_3(x) = +0.8$$

  we choose $\hat{f}(x) = 3$

Handwritten digit classification

confusion matrix, test set

	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

error rate is around 14% (same as for training set)

Adding new features

- let's add 5000 random features (!), $\max\{(Rx)_j, 0\}$
  - R is 5000×494 matrix with entries ± 1, chosen randomly
- now use least squares classification with 5494 feature vector

- results: training set error 1.5%, test set error 2.6%
- can do better with a little more thought in generating new features
- indeed, even better than humans can do (!!)

Results with new features

confusion matrix, test set

	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