



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*
- ▶ $f : \mathbf{R}^n \rightarrow \mathbf{R}$ 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

- vectors*
- scalars.*
- ▶ we are given some *data*
 - ▶ $x^{(1)}, \dots, x^{(N)}$, $y^{(1)}, \dots, y^{(N)}$
also called *observations*, *examples*, *samples*, or *measurements*
 - ▶ $x^{(i)}, y^{(i)}$ is *i*th *data pair*
 - ▶ $x_j^{(i)}$ is the *j*th component of *i*th data point $x^{(i)}$

Feature engineering

- ▶ start with original or base feature n -vector x
- ▶ choose basis functions f_1, \dots, f_p to create ‘mapped’ feature p -vector

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

- ▶ now fit linear in parameters model with mapped features

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

- ▶ check the model using validation

often want
to do linear
modelling with
a set of features
extracted from
the original
 x -vectors.

Model

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

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

- ▶ $f_i : \mathbf{R}^n \rightarrow \mathbf{R}$ 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

Minimization of the RMS
(Root of the Mean of Squares).

- ▶ 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

$$\vec{r} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_N \end{bmatrix} : \text{RMS} = \sqrt{\frac{\vec{r}^T \vec{r}}{N}} = \left(\frac{(r^{(1)})^2 + \dots + (r^{(N)})^2}{N} \right)^{1/2}$$

Essential: Minimize $\vec{r}^T \vec{r}$

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

We solve

$$A\vec{\theta} = \vec{y}$$

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

Fitting a constant model

Can we do better with a lin. model
than just guessing the average?

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

- ▶ $A = \mathbf{1}$ so

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

- ▶ the mean of $y^{(1)}, \dots, y^{(N)}$ is the least squares fit by a constant

- ▶ MMSE is $\text{std}(\mathbf{y}^d)^2$; RMS error is $\text{std}(\mathbf{y}^d)$

← Standard dev. of the
observed y-values.

- ▶ more sophisticated models are judged against the constant model

Fitting univariate functions

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

$i = 1, \dots, N$ i.e. N data-points
available for plotting to
get some idea of the functional
relationship between x and y .

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

intercept

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

See 13.1.1 in VMLS

slope

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

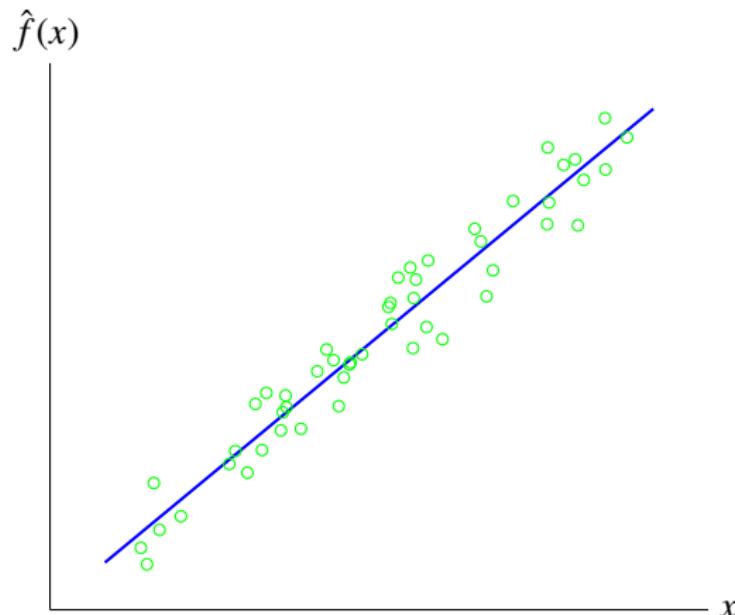
$$\hat{\theta} = \overset{\wedge}{\theta} = (A^T A)^{-1} A^T \hat{y}$$

||
 $\begin{bmatrix} \hat{\theta}_1 \\ \hat{\theta}_2 \end{bmatrix}$

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

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

Example



Asset α and β

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

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

- 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

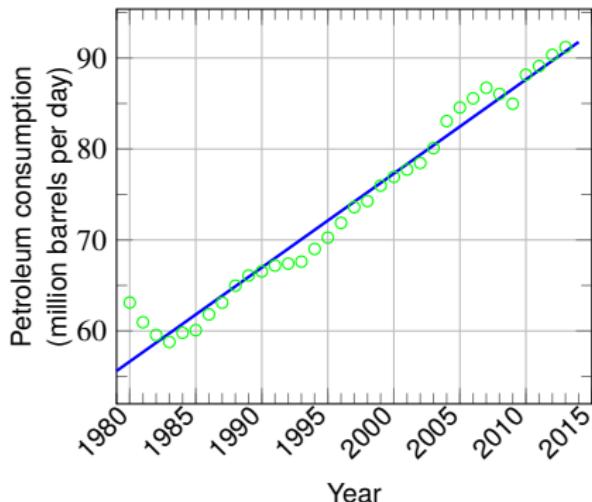
- ▶ $y^{(i)}$ is value of quantity at time $x^{(i)} = i$
- ▶ $\hat{y}^{(i)} = \hat{\theta}_1 + \hat{\theta}_2 i, \quad i = 1, \dots, N$, is called *trend line*
- ▶ $y^d - \hat{y}^d$ is called *de-trended time series*
- ▶ $\hat{\theta}_2$ is *trend coefficient*

 ← Slope of the lin. model.

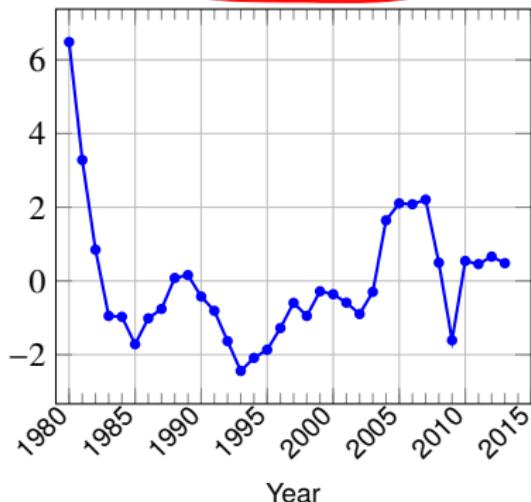
World petroleum consumption

Residual plot.

Consumption



De-trended consumption



Polynomial fit

Orig. data.

(x_1, y_1)

(x_2, y_2)

:

(x_N, y_N)

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

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

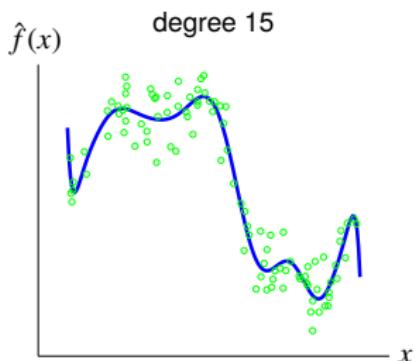
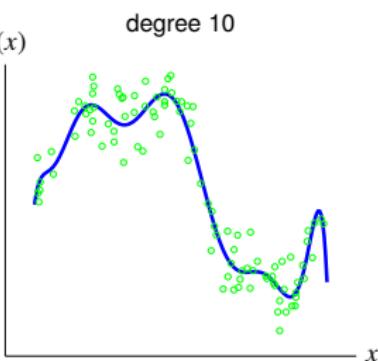
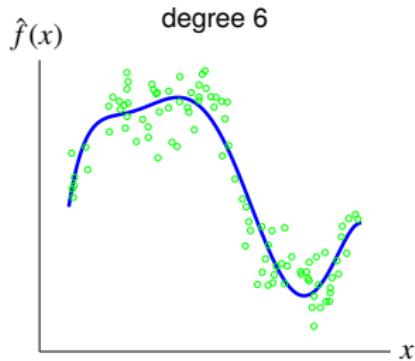
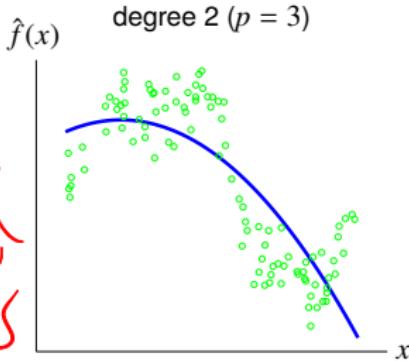
(here x^i means scalar x to i th power; $x^{(i)}$ is i th data point)

- A is Vandermonde matrix

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

Example

$N = 100$ data points



~~Guershoni
Which mode
is the better
choice?~~

Regression as general data fitting

This is actually
an inner product

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

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

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
 - $\tilde{x} = (f_2(x), \dots, f_p(x))$ are 'transformed features'
 - $v = \theta_1$, $\beta = \theta_{2:p}$

Auto-regressive time series model

- ▶ time series $z_1, z_2, \dots, z_n, z_{n+1}, \dots, \dots, \dots$.
- ▶ *auto-regressive* (AR) prediction model:

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

- ▶ 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,

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

- ▶ put in general form with

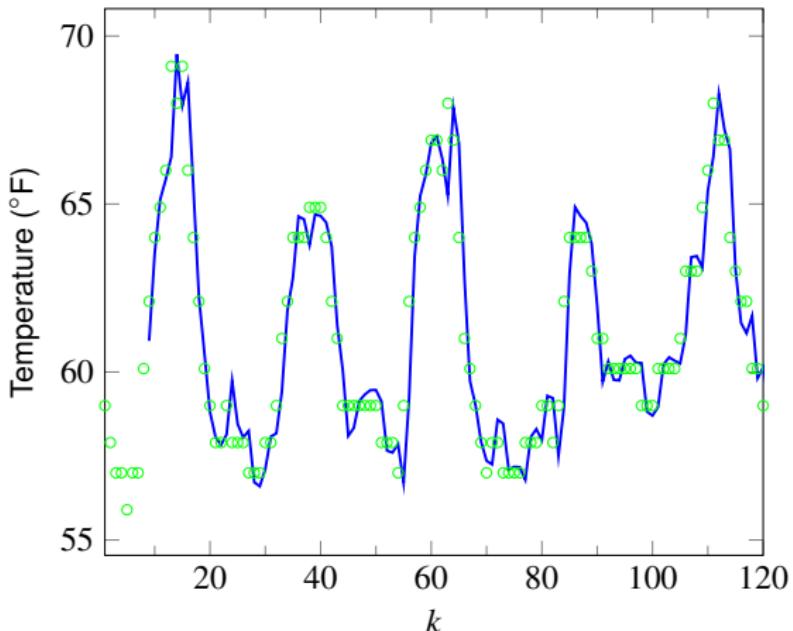
$$y^{(i)} = z_{M+i}, \quad x^{(i)} = (z_{M+i-1}, \dots, z_i), \quad i = 1, \dots, T - M$$

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*

80% or 90% of available data.

Don't forget to compare with the constant model.

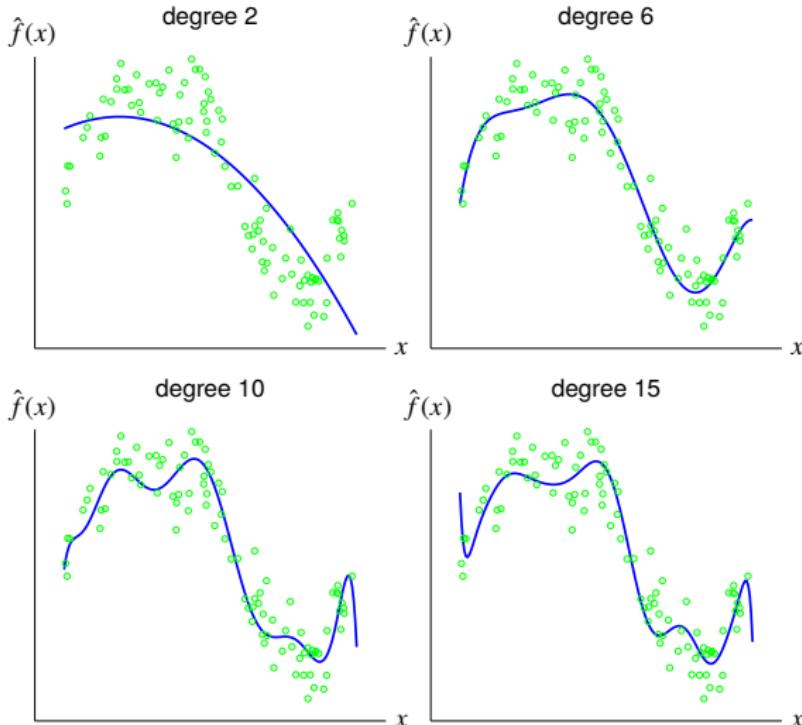
Validation

* USE: For model selection ! *

- ▶ 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

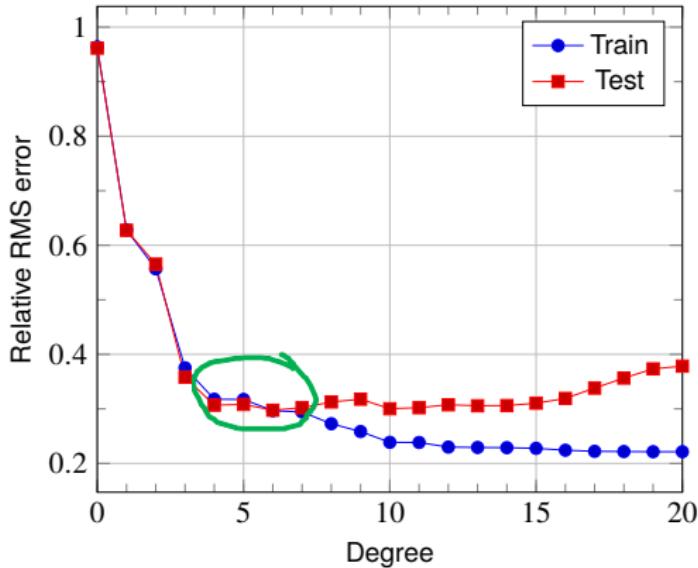
Example

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 $i = 1, \dots, 10$, 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

Fold	Model parameters			RMS error	
	ν	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

Transforming features

Some examples:

- ▶ *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

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

(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

Fold	Model parameters								RMS error	
	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