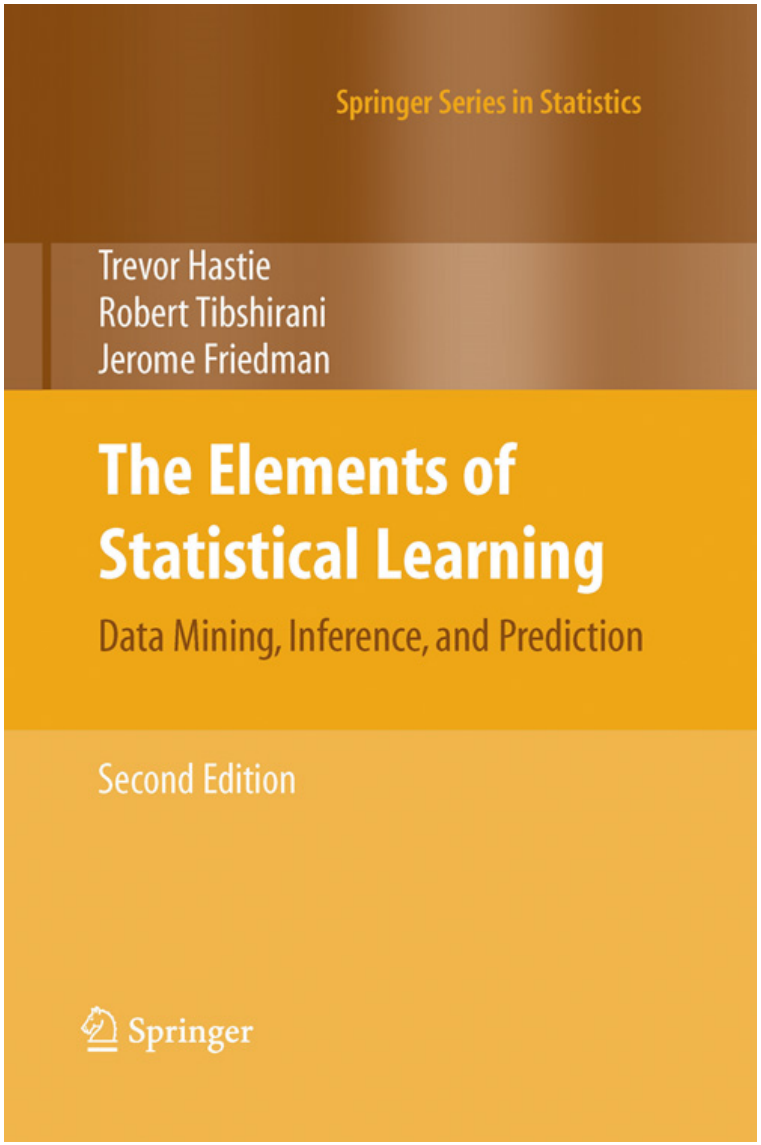
Multiple Linear Regression

David Bethge & Fabio Ferreira

DHBW Karlsruhe

May 7, 2018

Recommended Literature



Overview

1. basics
   - vector and matrix algebra
   - expectation value, variance
2. notation
3. simple linear regression
   - introduction
   - statistical properties
   - model evaluation
   - prediction
4. multiple linear regression
   - Assessing goodness of fit
5. algorithms
6. algorithms
   - ridge regression
   - LASSO
   - nearest neighbor method

vector

We begin by defining a vector, a set of n numbers which we shall write in the form

$$\vec{a} = \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{pmatrix}$$

is called a column vector of the size $(n \times 1)$.
If n numbers are arranged in a horizontal array, as in

$$\vec{a} = \begin{pmatrix} a_1 & a_2 & \dots & a_n \end{pmatrix}$$

then $\vec{a}$ is called a row vector of the size $(1 \times n)$.

matrix

An $n \times p$ matrix A has n columns and p rows:

$$A = \begin{pmatrix} a_{11} & \dots & a_{1p} \\ \dots & \ddots & \dots \\ a_{n1} & \dots & a_{np} \end{pmatrix}$$

We will write a_{ij} for the element in column i and row j.
We will assume $a_{ij} \in \mathbb{R}$.

Most of the datasets we use in machine learning will have the form of the denoted matrix A with n observations and p variables.

mathematical operations on vectors and matrices

- scalar multiplication
- matrix multiplication
- sum of matrices
- tensor matrices
- inverse matrices

expectation value

In probability theory, the expected value of a random variable, intuitively, is the long-run average value of repetitions of the experiment it represents. For example, the expected value in rolling a six-sided dice is 3.5.

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x f(x)\,\mathrm{d}x$$

Assume

$$X = (X_1 \quad X_2 \quad \dots \quad X_p)$$

is a p-dimensional random vector.
Then its expectation value vector is:

$$\mathbb{E}[X] = \begin{pmatrix} \mathbb{E}[X_1] \\ \mathbb{E}[X_2] \\ \dots \\ \mathbb{E}[X_p] \end{pmatrix}$$

mean vector

Often the distribution of X is unknown so expectation value cannot be calculated. To estimate the expectation value we will use the (empirical) mean vector.
The mean of a sample of size n is defined as:

$$\bar{X} = \frac{1}{n}(X_1 + X_2 + ... + X_n) = \frac{1}{n}\sum_{i=1}^{n} X_i$$

As with $\mathbb{E}[X], \bar{X}$ can be also written as a vector:

$$\bar{X} = \begin{pmatrix} \bar{X}_1 \\ \bar{X}_2 \\ ... \\ \bar{X}_p \end{pmatrix} = \begin{pmatrix} \frac{1}{n}\sum_{i=1}^{n} X_{1i} \\ \frac{1}{n}\sum_{i=1}^{n} X_{2i} \\ ... \\ \frac{1}{n}\sum_{i=1}^{n} X_{pi} \end{pmatrix}$$

Here we calculate the mean value of each variable by summing over the observations (and dividing by the number of observations).

variance

In probability theory and statistics, variance is the expectation of the squared deviation of a random variable from its mean. Informally, it measures how far a set of (random) numbers are spread out from their average value.

$$
\begin{aligned}
Var(X) &= \mathbb{E}[(X - \mathbb{E}[X])^2] \\
&= E[X^2 - 2XE[X] + E[X]^2] \\
&= E[X^2] - 2E[X]E[X] + E[X]^2 \\
&= E[X^2] - E[X]^2
\end{aligned}
$$

dataset notation

- i: index of the observation or observation
- j: index of a variable/feature
- x_{ij}: value of the j-th variable for the i-th observation, where $i = 1, 2, ..., n$ and $j = 1, 2, ..., p$.
- X: a $n \times p$ matrix whose (i, j)-th element is x ij
- x^j: column vector of X
- x_i: row vector of X

$$X = \begin{pmatrix} x_{11} & \dots & x_{1j} & \dots & x_{1p} \\ \dots & \dots & \dots & \dots & \dots \\ x_{i1} & \dots & x_{ij} & \dots & x_{ip} \\ \dots & \dots & \dots & \dots & \dots \\ x_{n1} & \dots & x_{nj} & \dots & x_{np} \end{pmatrix}$$

example for dataset notation

- n: number of distinct data points, or observations, in the sample
- p: number of variables that are available for prediction

Example: Wage data set consists of 12 variables for 3,000 people

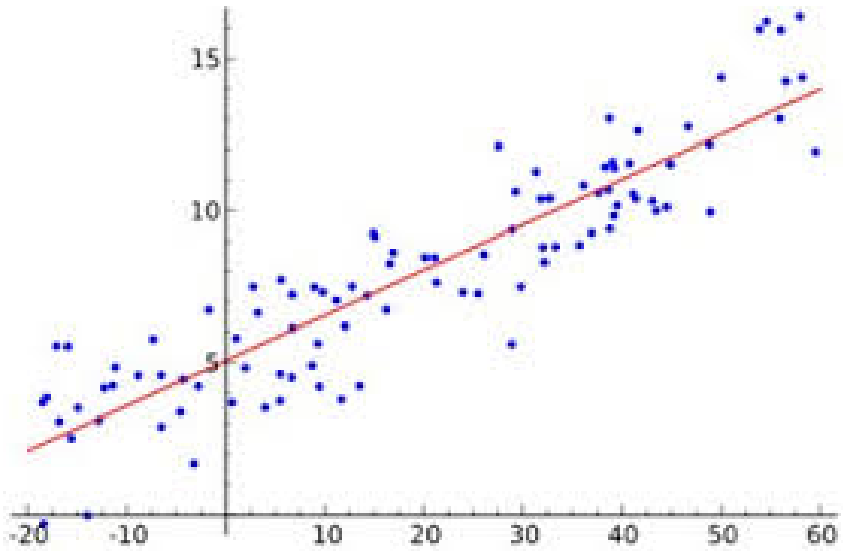
- n = 3,000 observations
- p = 12 variables (such as year, age, wage, and more).

linear regression

"Essentially, all models are wrong, but some are useful" (George Box)

introduction

Now as we introduced the very basics of linear algebra needed for machine learning, we have a quick look at the very popular linear regression method.
Basic idea:
Use data to identify linear relationships among variables and use these relationships to make predictions.



motivation

We look at a model with the following parameters:

- unknown parameters β
- independent variables X
- dependent variable y

and want to estimate a function $f(X, \beta)$, with:

$$y \approx f(X, \beta) \tag{1}$$

model

To further specify the function $f(...)$ **we restrict ourselves to a linear relationship**:

Model assumption

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p + \epsilon = \beta_0 + \sum_{i=1}^{p} \beta_i x_i + \epsilon$$

But in practice the linear assumption does not always hold.
Nonetheless linear regression has proven to be a important tool in practice.

simple linear model

We can observe n data points $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, then we can write for each observation i:

$$y_i = \beta_0 + \beta_1 * x_i + \epsilon_i$$

with

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

- Here, the noise ϵ_i represents the fact that our data wont fit the model perfectly (we model it gaussian).
- Note that the intercept β_0, the slope β_1, and the noise variance σ^2 are all treated as fixed (i.e., deterministic) but unknown quantities

estimating the simple linear model

Since we observe: $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, we need to solve the following optimization problem:

$$\min_{\beta_0,\beta_1} \sum_{i=1}^{n} [y_i - (\beta_0 + \beta_1 x_i)]^2 = \sum_{i=1}^{n} [y_i - \hat{y}_i]^2 = \sum_{i=1}^{n} e_i^2$$

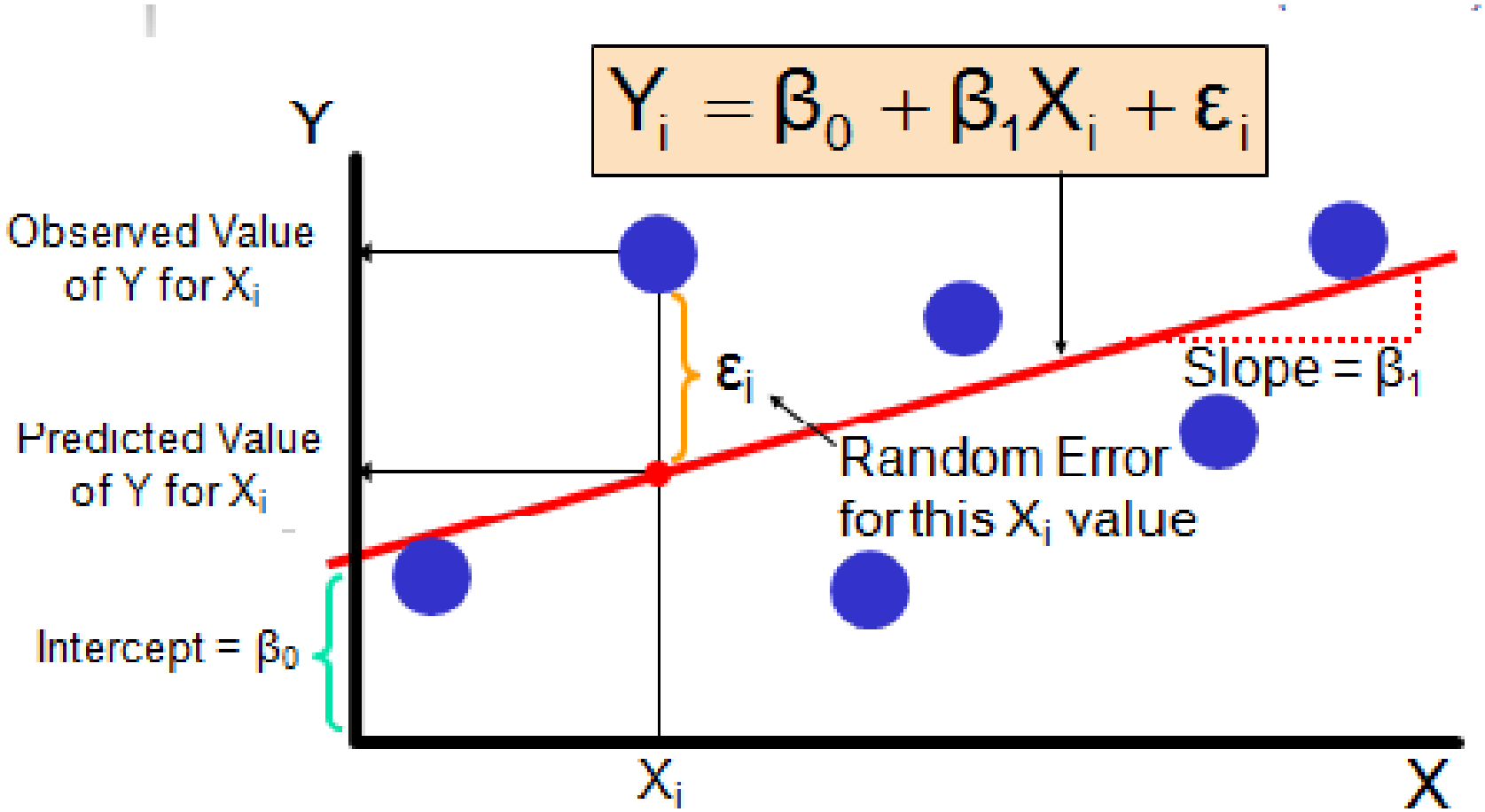
- $\hat{y}_i$ is called the prediction for observation i
- $e_i = y_i - \hat{y}_i$ is the i'th residual

Solution:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} x_i y_i - \frac{1}{n} \sum_{i=1}^{n} \sum_{i=0}^{n} x_i y_i}{\sum_{i=0}^{n} x_i^2 - \frac{1}{n} (\sum_{i=0}^{n} x_i)^2} = r \frac{s_y}{s_x}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

visualization of the simple linear model



("https://madhureshkumar.files.wordpress.com/2015/07/regressioncurv.png")

accuracy of the coefficient estimates

The standard error of an estimator reflects how it varies under repeated sampling.

$$SE(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=0}^{n}(x_i - \bar{x})^2}$$

$$SE(\hat{\beta}_0)^2 = \sigma^2[\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=0}^{n}(x_i - \bar{x})^2}]$$

with $\sigma^2 = Var(\epsilon)$

We can also show that our estimate satisfies:

$$\hat{\beta}_i \sim N(\beta_i, \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2})$$

Now with the standard errors we can calculate confidence intervals: A 95% confidence interval ($\alpha = 0.05$) is defined as a range of values such that with 95% probability, the range will contain the true unknown value of the parameter:

$$\beta_1 \pm t_{n-p}^{1-\alpha/2}\sqrt{SE(\hat{\beta_1}^2)}$$

accuracy of the coefficient estimates (2)

That is, there is approximately a 95% chance that the interval

$$[\beta_1 - t_{n-p}^{1-\alpha/2}\sqrt{SE(\hat{\beta}_1)^2}, \beta_1 + t_{n-p}^{1-\alpha/2}\sqrt{SE(\hat{\beta}_1)^2}]$$

will contain the true value of β_1 (under a scenario where we got repeated samples like the present sample)

hypothesis testing

Standard errors can also be used to perform hypothesis tests on the coefficients. The most common hypothesis test involves testing the null hypothesis of:

- H_o: There is no relationship between X and Y versus the alternative hypothesis
- H_1: There is some relationship between X and Y .

Mathematically, this corresponds to testing:

- H_o: $\beta_1 = 0$
- H_1: $\beta_1 \neq 0$

since if $\beta_1 = 0$ then the model reduces to $y = \beta_0 + \epsilon$, and X is not associated with Y .

hypothesis testing

To test the null hypothesis, we compute a t-statistic, given by:

$$t = \frac{\hat{\beta}_1 - 0}{SE(\hat{\beta}_1)}$$

This will have a t-distribution with $n - 2$ degrees of freedom, assuming $\beta_1 = 0$.
Using statistical software, it is easy to compute the probability of observing any value equal to $\mid t \mid$ or larger. We call this probability the p-value.

assessing the overall accuracy of the model

- We compute the Residual Standard Error:

$$RSE = \sqrt{\frac{1}{n-2}RSS} = \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

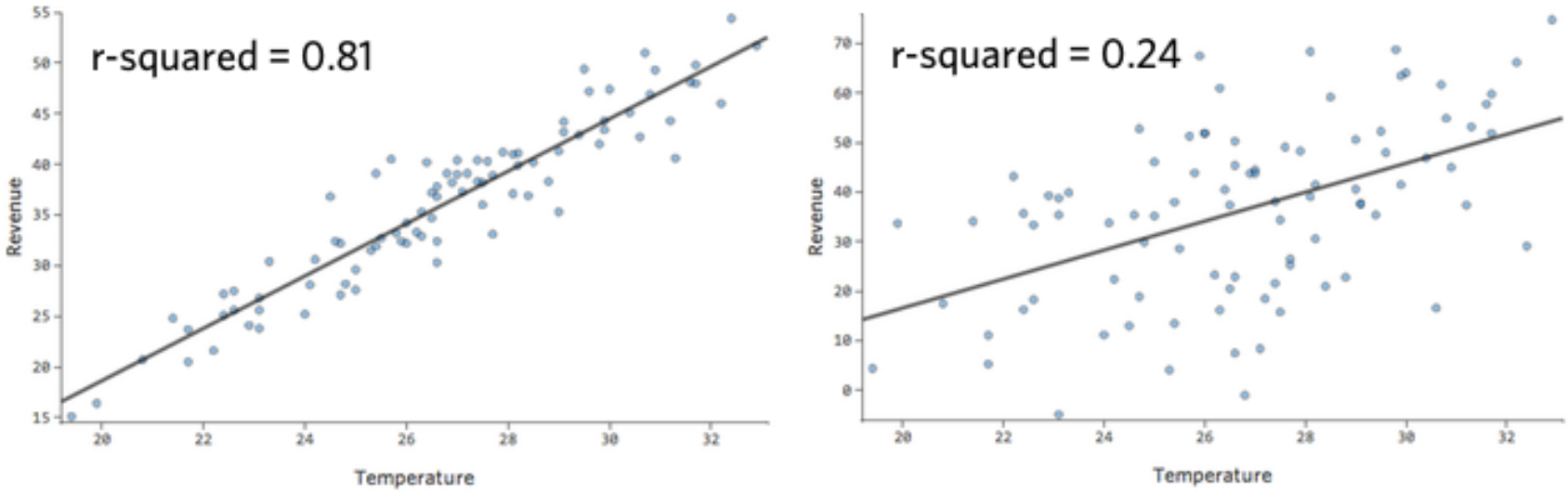
- R-squared or fraction of variance explained is:

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}$$

  where TSS is the total sum of squares: $\sum_{i=1}^{n}(y_i - \bar{y})^2$
- R-squared is always $\in [0, 1]$, whereby 1 represents a perfect fit
- decomposition of the variance:

$$\underbrace{\sum_{i=1}^{n}(y_i - \bar{y})^2}_{\text{TSS}} = \underbrace{\sum_{i=1}^{n}\hat{e_i}^2}_{\text{RSS}} + \underbrace{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}_{\text{ESS}}$$

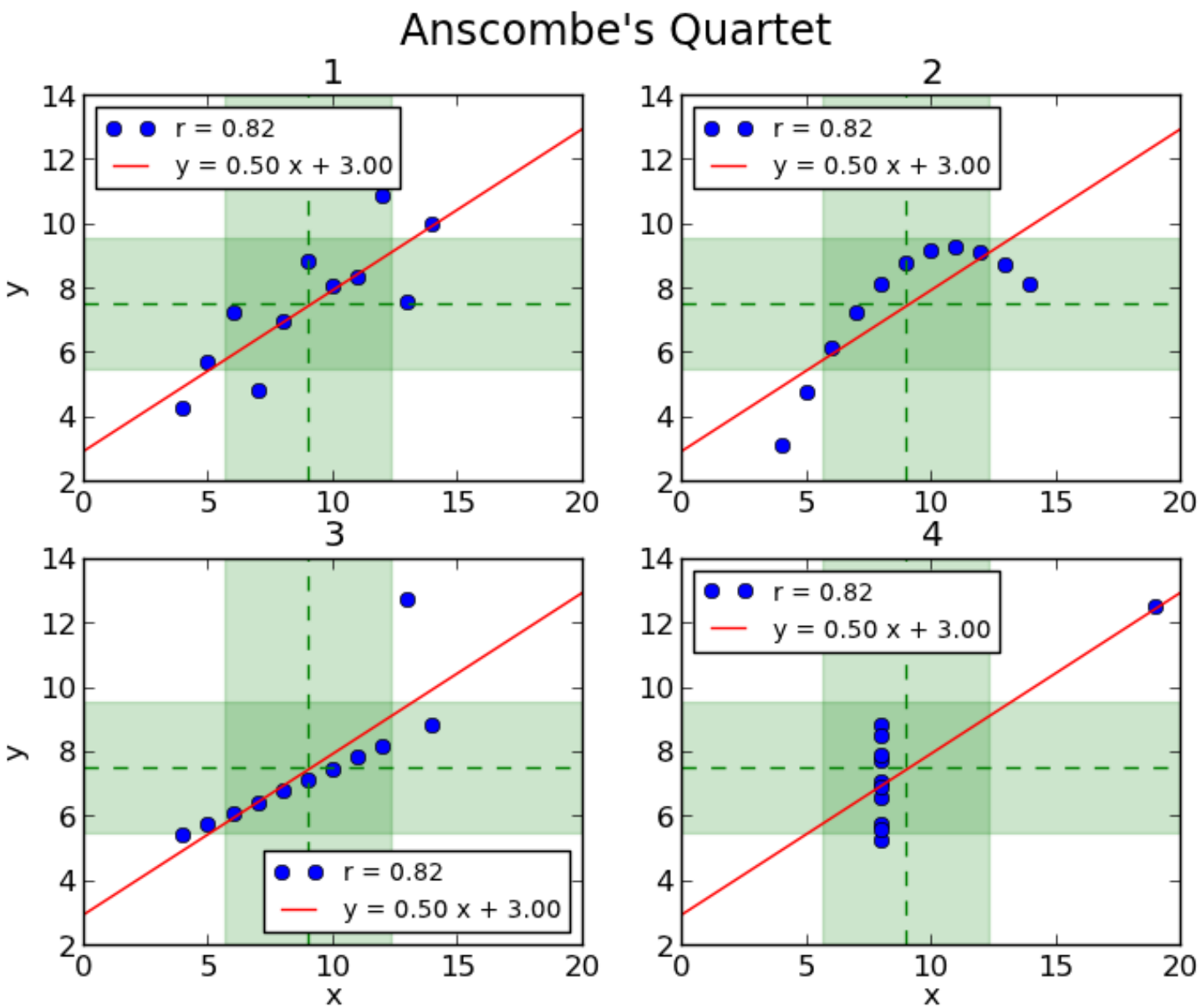
R-squared



be careful using R-squared as the only measure for quality of the model

- R-squared increases inherently with the number of predictors, even if the added variables are not explaining something
- use the adjusted $R^2_{adj.}$ instead: $1 - (1 - R^2)\frac{n-1}{n-p-1}$

linear regression is intriguing



prediction

If the β parameters are estimated on the training data, now we can make predictions on new (unseen) data. From now on we call the estimated parameters $\hat{\beta}$ and a new unseen data points X_0. So the prediction for the given y is:

$$\hat{y} = \hat{\beta}_o + X_0\hat{\beta}_1$$

Example: We want to find out how the number of customers affects the sales of a supermarket (#sales $\approx$ #customers).
So let's say we found out that $(\hat{\beta}_0, \hat{\beta}_1) = (-2, 0.4)$, which is a very considerate estimate. So if we see 20 customers coming in the supermarket: the estimate sale would be: $\hat{y} = -2 + 0.4 * 20 = 6$.

multiple linear regression

Beyond dimension 2

notation

In the multivariate case we want to use more than one variable (it is also called multiple regression):

Linear model form

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p + \epsilon = \beta_0 + \sum_{i=1}^{p} \beta_i x_i + \epsilon$$

Can be written as:

Matrix notation

$$y = X\beta + e$$

example

Let's recall our matrix notation:

	Coffein Consumption (x_1)	noise level (x_2)	power of concentration (y)
student 1	3	0	85
student 2	0	6	41
student 3	1	3	60
student 4	4	9	53
...	...	...	...

variable assignment

This is how we initialize our matrices for the multiple regression:

$$X = \begin{bmatrix} 1 & x_{11} & \dots & x_{1j} & \dots & x_{1p} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & x_{i1} & \dots & x_{ij} & \dots & x_{ip} \\ \dots & \dots & \dots & \dots & \dots & \\ 1 & x_{n1} & \dots & x_{nj} & \dots & x_{np} \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix} \quad e = \begin{bmatrix} e_1 \\ e_2 \\ \dots \\ e_n \end{bmatrix} \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \dots \\ \beta_p \end{bmatrix}$$

Question: Why is there a column of ones in X?

Assumptions

Regressor matrix:

- X is stochastic
- X has full rank: $Rank(X) = p$

Error Term: $e|X \sim (0, \sigma^2 I_n)$

- strict exogeneity: $\mathbb{E}[e_i|X] = 0 \quad \forall i$
- conditioned homoskedacity: $Var[e_i|X] = \sigma^2 \quad i \neq j$
- conditionially uncorrelated: $\mathbb{E}[e_i e_j|X] = 0 \quad \forall i$
- sometimes we assume: $e|X \sim N(0, \sigma^2 I_n)$

OLS estimation

Goal: minimize the distances between true and predicted values (ordinary least squares)

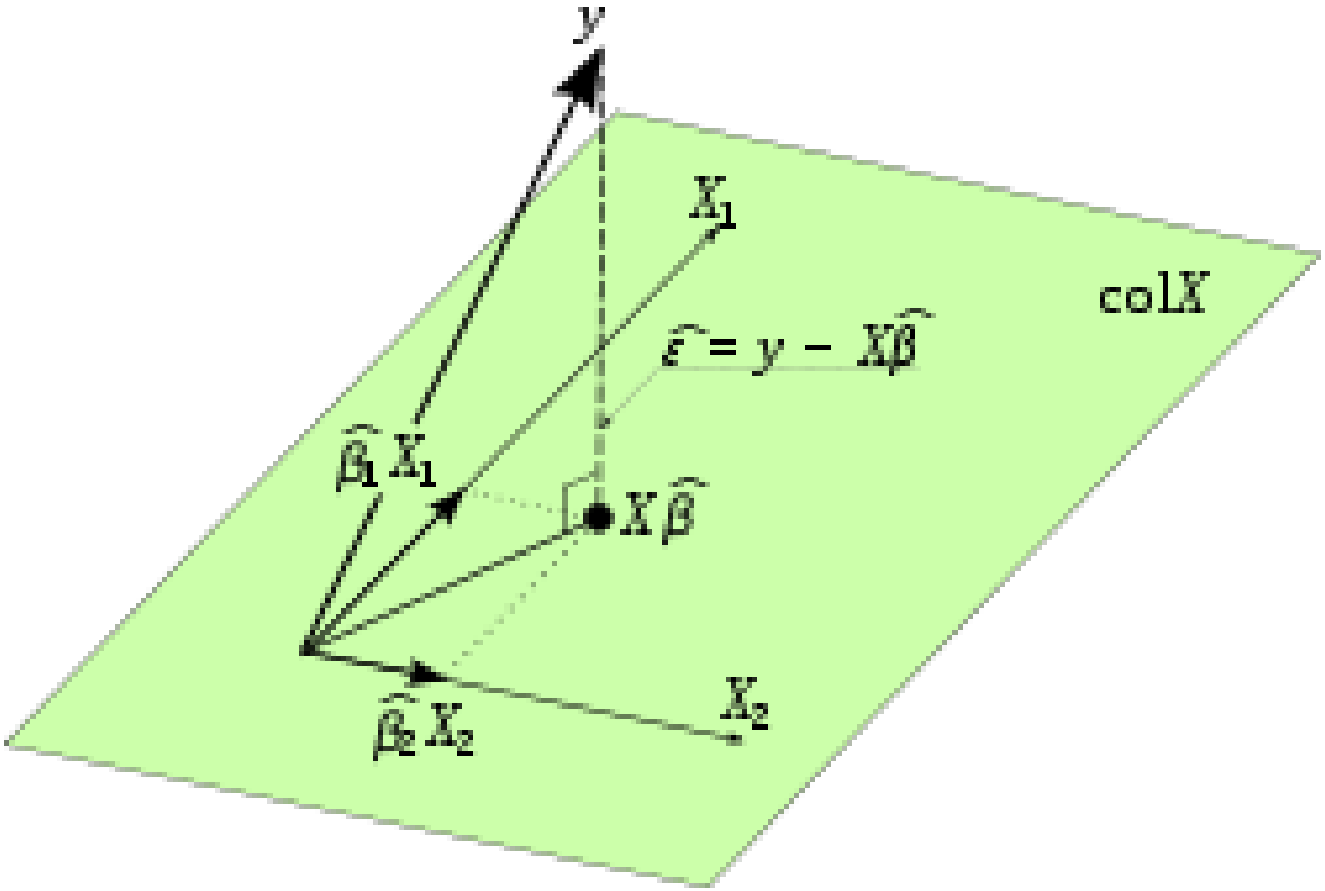
$$
\begin{aligned}
min_\beta &= \frac{\partial}{\partial \beta}(y - X\beta)'(y - X\beta) = \frac{\partial}{\partial \beta}\hat{e}'\hat{e} \\
&= \frac{\partial}{\partial \beta}(y'y - 2\beta'X'y + \beta'X'X\beta) \\
&= -2X'y + 2X'Xb \\
X'y &= X'Xb
\end{aligned}
$$

optimizing β yields the regression coefficient to be:

Coefficient estimator

$$b = (X'X)^{-1}X'y$$

geometrical understanding



properties of the OLS estimator b

It is (under some specific assumptions) the best linear unbiased estimator (BLUE):
b is unbiased:

$$\mathbb{E}[b] = \mathbb{E}[(X'X)^{-1}X'y] = \mathbb{E}[(X'X)^{-1}X'(X\beta + e)] = (X'X)^{-1}X'X\beta = \beta$$

$$\begin{aligned} Var[b] =& Var[(X'X)^{-1}X'y] \\ =& Var[\beta + (X'X)^{-1}X'e] \\ =& Var[(X'X)^{-1}X'e] \\ =& (X'X)^{-1}X'Var[e]X(X'X)^{-1} \\ =& \sigma^2(X'X)^{-1} \end{aligned}$$

We can also write (normality not always given):

properties of the OLS estimate

$b \sim N(\beta, \sigma^2(X'X)^{-1})$

prediction

Now we want to do a prediction for a given data matrix X_0. First we estimate b and then we can infer the predicted values $\hat{y}$:

$$\begin{aligned}\hat{y} =& X_0 b \\ =& X_0 (X'X)^{-1} X'y\end{aligned}$$

We can show that the prediction is unbiased:

$$\begin{aligned}\mathbb{E}[\hat{y}] =& \mathbb{E}[X_0 b] \\ =& \mathbb{E}[X_0 (X'X)^{-1} X'y] \\ =& X_0 (X'X)^{-1} X' \mathbb{E}[X\beta + e] \\ =& X_0 (X'X)^{-1} X'X\beta + X_0 (X'X)^{-1} X'X \mathbb{E}[e] \\ =& X_0 (X'X)^{-1} X'X\beta = X_0 \beta\end{aligned}$$

prediction interval (1)

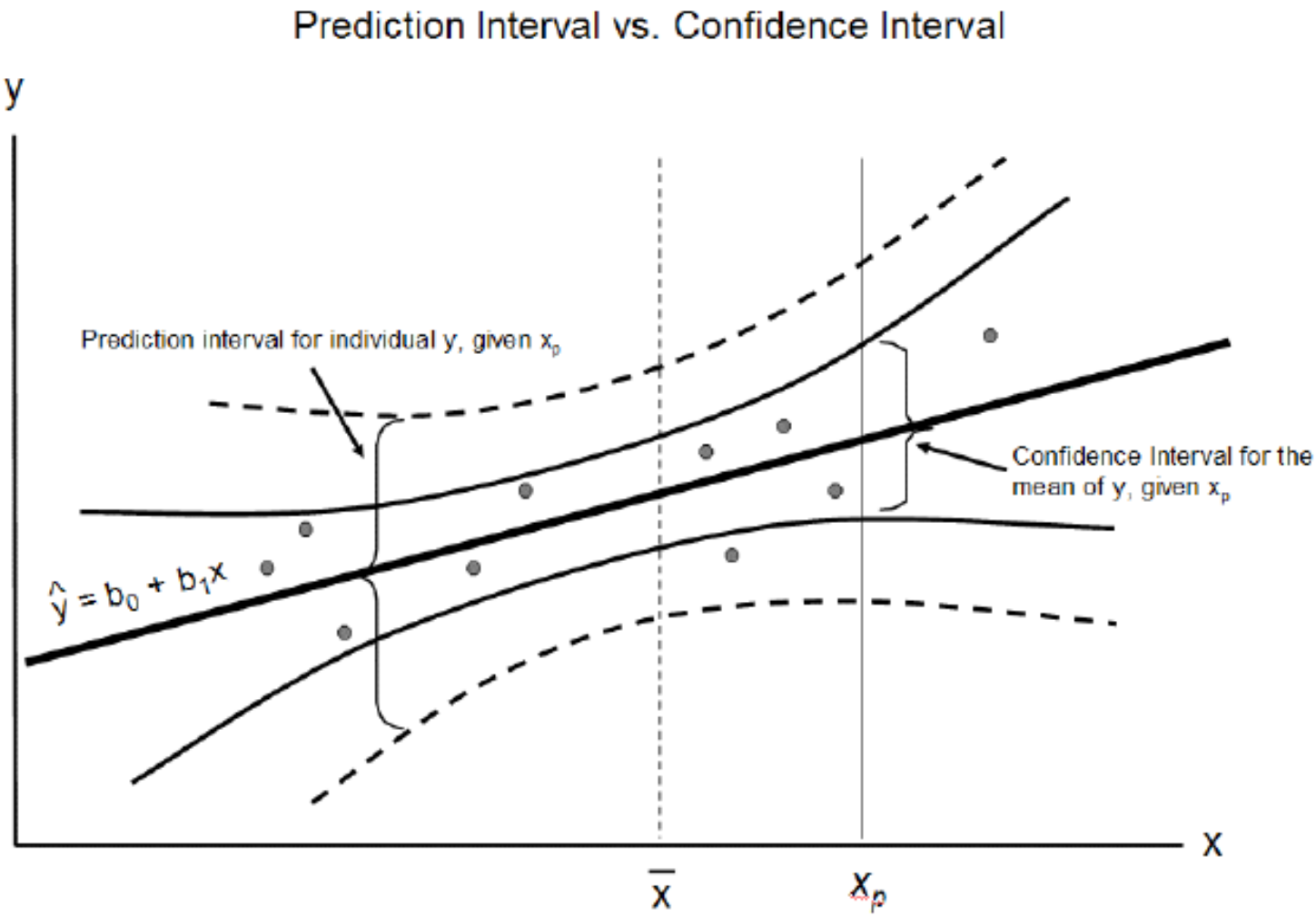
We learned how to calculate a point estimate for y, but it is often far more useful to get an prediction interval:

$$[\hat{y_0} - t_{n-p}^{1-\alpha/2}\sqrt{\hat{Var}_{pred}(X_0)}, \hat{y_0} + t_{n-p}^{1-\alpha/2}\sqrt{\hat{Var}_{pred}(X_0)}]$$

and

$$\hat{Var}_{pred}(X_0) = \sigma^2(I_0 + X_0(X'X)^{-1}X_0'$$

prediction interval (2)



R^2

In statistics, the coefficient of determination, denoted R^2, is the proportion of the variance in the dependent variable that is predictable from the independent variable(s):

$$R^2 = \frac{ESS}{TSS} = 1 - \frac{RSS}{TSS} = 1 - \frac{(y - \hat{y})'(y - \hat{y})}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = 1 - \frac{\hat{e}'\hat{e}}{y'y - n\bar{y}^2} \quad (2)$$

Ideally R^2 is close to 1, since then the predicted value $\hat{y}$ and the true values y are very close (residuals $\hat{e}'\hat{e} \to 0$).

adjusted R^2

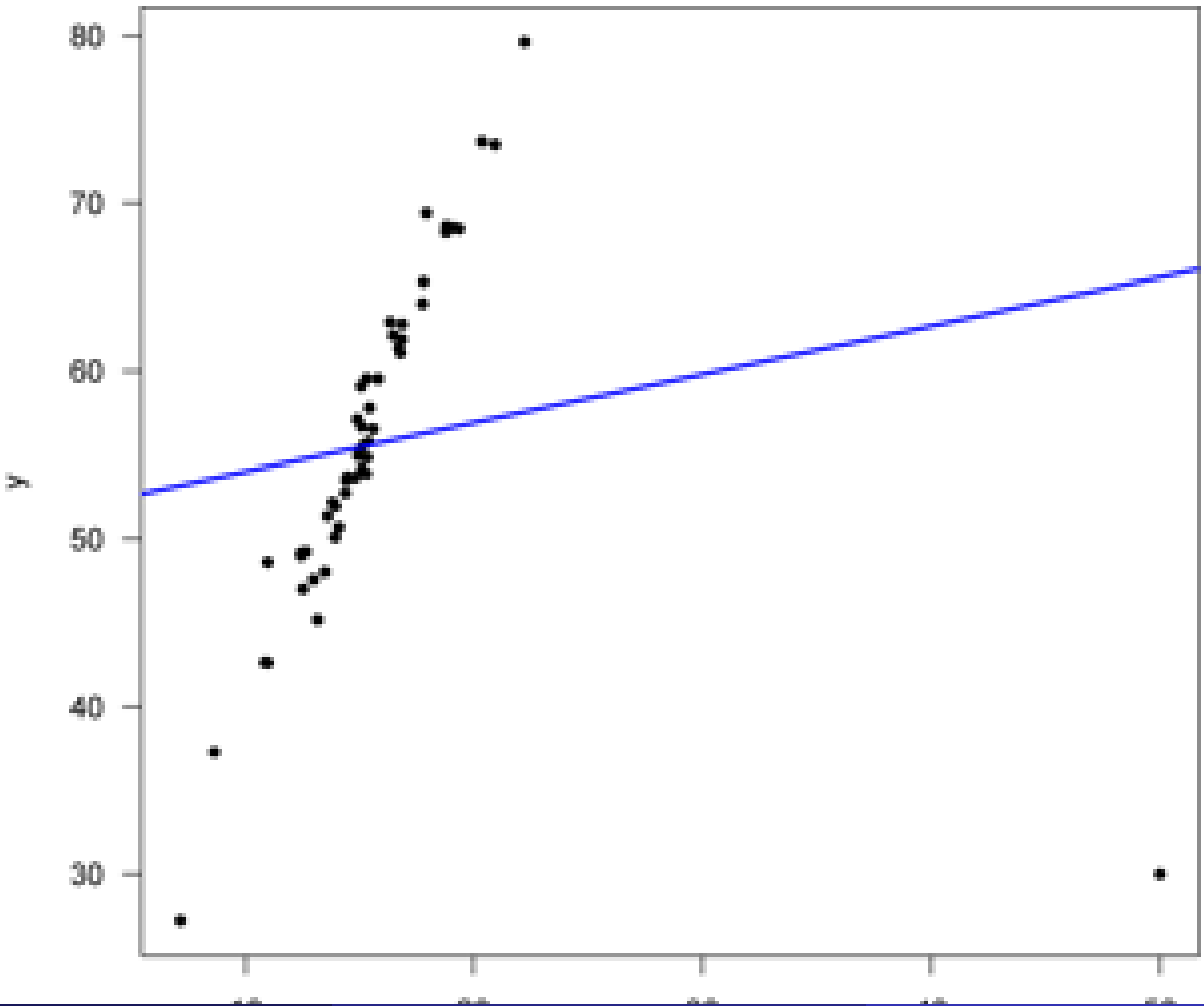
Unfortunately R^2 has some shortcomings, that are corrected with an adjusted R^2

- R2 increases with every predictor added to a model. As R2 always increases and never decreases, it can appear to be a better fit with the more terms you add to the model (misleading).
- Similarly, if your model has too many terms and too many high-order polynomials you can run into the problem of over-fitting the data.

Adjusted R2 also indicates how well terms fit a curve or line, but adjusts for the number of terms in a model. If you add more and more useless variables to a model, adjusted r-squared will decrease. If you add more useful variables, adjusted r-squared will increase.

$$R^2_{adj.} = 1 - (1 - R^2)\frac{n-1}{n-p-1} \tag{3}$$

the effect of outliers



pros & cons of multiple linear regression

- easy and intuitive "simple" approach
- it works in many cases: even when it doesnt fit the data exactly, we can use it to find the nature of the relationship between the two variables
- It assumes there is a linear relationship between the variables which is incorrect sometimes
- it is very sensitive to the anomalies in the data (or outliers)
- it is prone to overfitting of the data (i.e some noisy data also considered as useful data)

Algorithms: Ridge Regression, LASSO, Nearest Neighbor

Digging deeper

ridge regression - motivation

Motivation: too many predictors

- It is not unusual to see the number of input variables greatly exceed the number of observations, e.g. micro-array data analysis, environmental pollution studies.
- With many predictors, fitting the full model without penalization will result in large prediction intervals, and OLS regression estimator may not uniquely exist.

Motivation: ill-conditioned X

- Because the OLS estimates depend upon $(XX)^{-1}$, we would have problems in computing b if XX is not invertable.

ridge regression - derivation

One way out of this situation is to shrink the regression coefficients via **ridge regression**.
In ridge regression we penalize big values for the coefficient estimator.
Thus our optimization problem looks like this:

$$
\begin{aligned}
min_\beta &= \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p}(\beta_j^2) \\
&= \frac{\partial}{\partial \beta}(y - X\beta)'(y - X\beta) + \lambda \beta'\beta \\
&= \frac{\partial}{\partial \beta}(y'y - 2\beta' X'y + \beta' X'X\beta + \lambda \beta'\beta) \\
&= -2X'y + 2X'X\hat{\beta}_R + 2\lambda \hat{\beta}_R \\
X'y &= \hat{\beta}_R(X'X + \lambda I_p)
\end{aligned}
$$

ridge regression

ridge regression coefficient

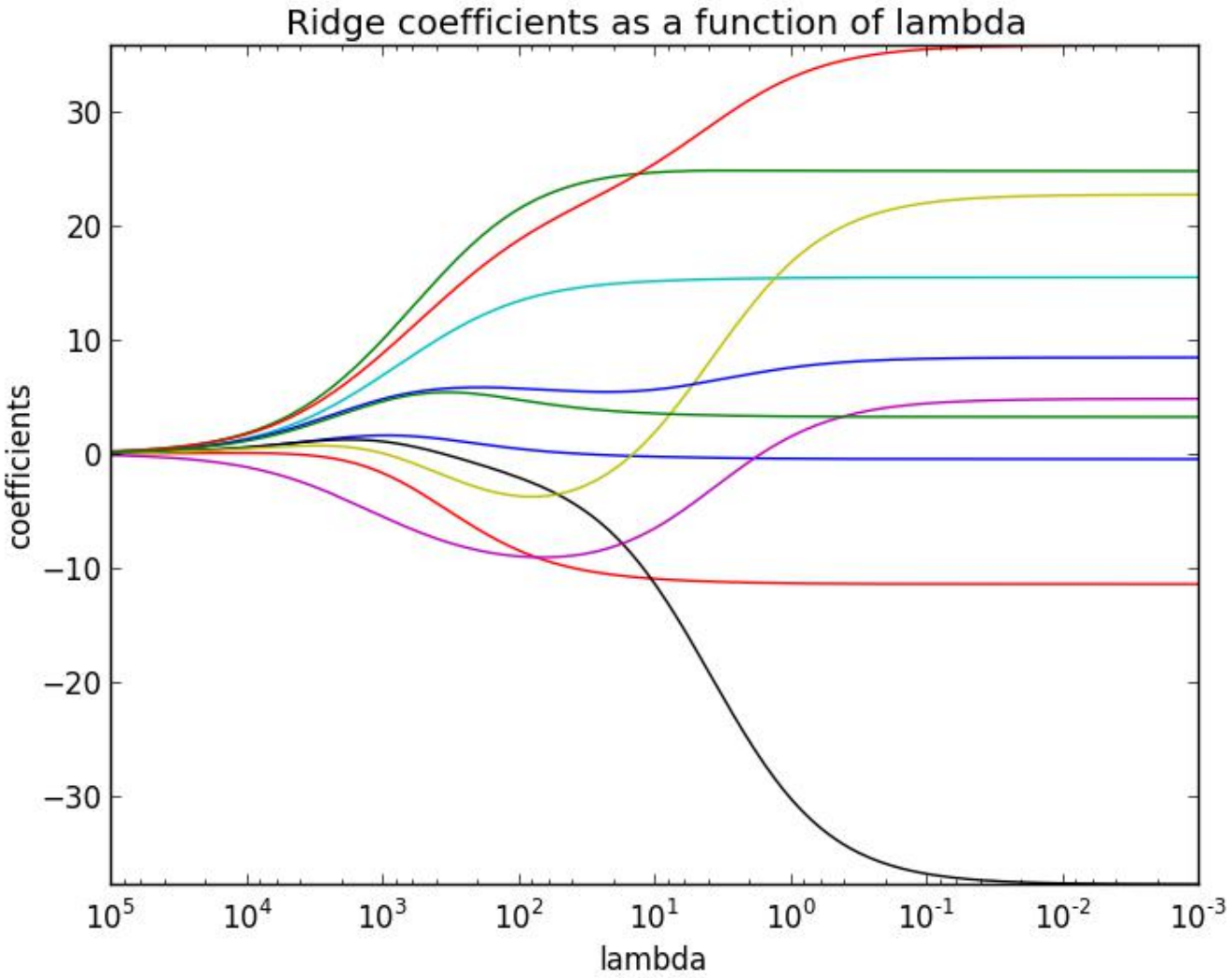
$$\hat{\beta}_R = (X'X + \lambda I_p)^{-1} X'y$$

Thus via Ridge regression potential instability in the OLS estimator $b = (X'X)^{-1}X'y$ could be improved by adding a small constant value λ to the diagonal entries of the matrix XX before taking its inverse.
The penalty term is λ (a pre-chosen constant) times the squared norm of the β vector. This means that if the β_j's take on large values, the optimization function is penalized. We would prefer to take smaller β_j's, or β_j's that are close to zero to drive the penalty term small.

Question: Which λ value results in the normal OLS regression?
Hint:

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p}(\beta_j^2)$$

shrinking of coefficient estimates



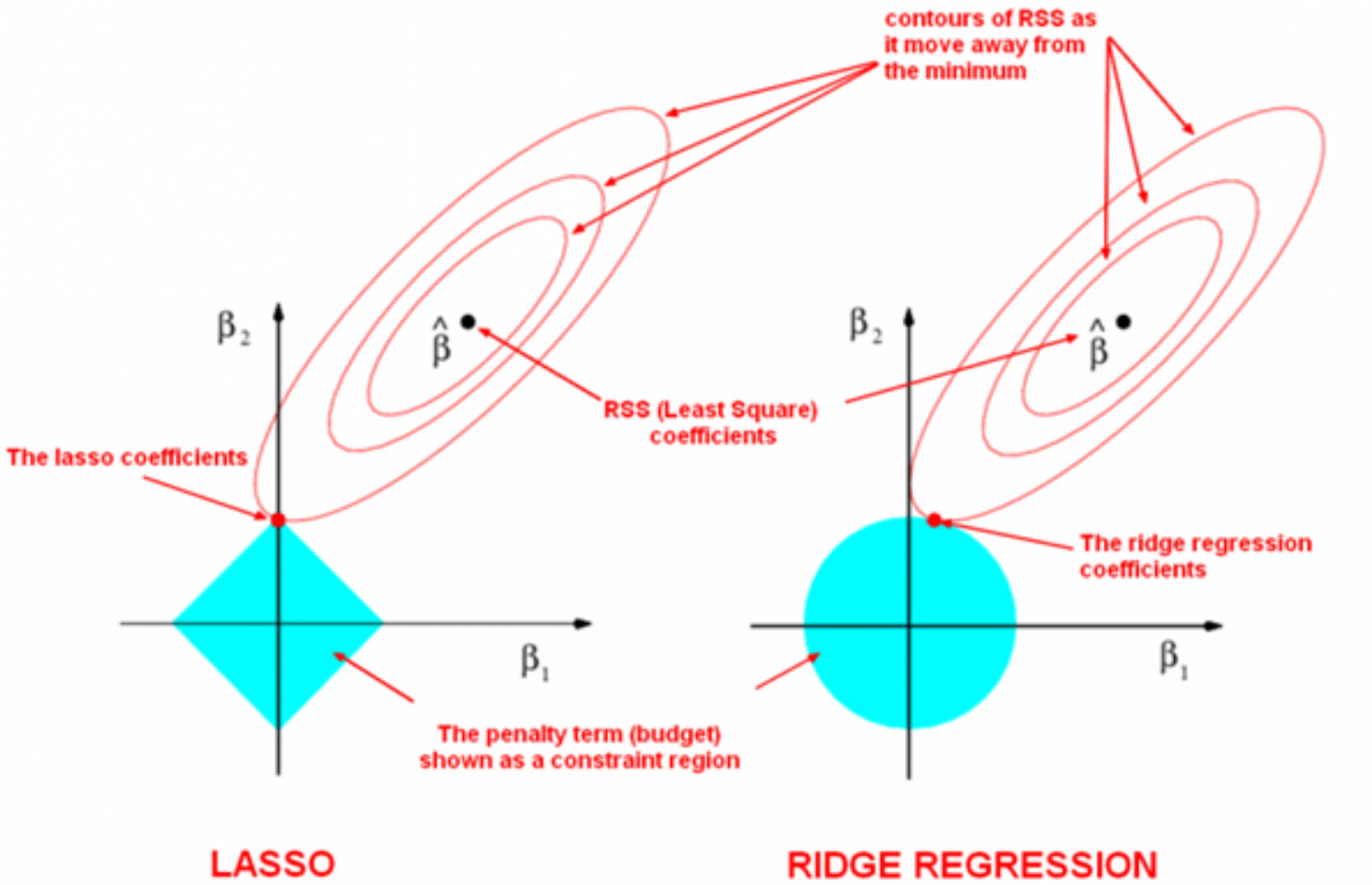
LASSO

LASSO (least absolute shrinkage and selection operator) is similar to ridge regression. In LASSO our optimization problem looks like this:

$$min_\beta = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

- LASSO forces certain coefficients to be set to zero effectively choosing a simpler model that does not include those coefficients (better interpretability).
- This idea is similar to ridge regression, in which the sum of the squares of the coefficients is forced to be less than a fixed value, though in the case of ridge regression, this only shrinks the size of the coefficients, it does not set any of them to zero.

LASSO vs. Ridge



Nearest Neighbor Method

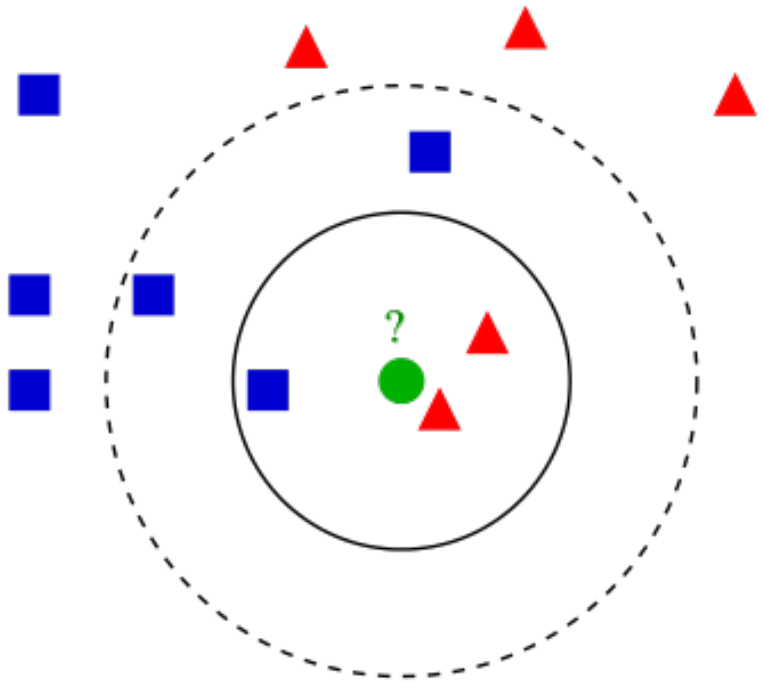
"Sometimes our neighbors define who we are."

motivation

Nearest neighbor search (NNS), as a form of **proximity** search, is the optimization problem of finding the point in a given set that is closest (or most similar) to a given point.
Closeness is typically expressed in terms of a dissimilarity function: the less similar the objects, the larger the function values.
The most popular method is the **k-nearest neighbor method**.

k-nearest neighbor

identifies the top k nearest neighbors to the query:



Example of k-NN classification. The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles. If $k = 3$ (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If k = 5 (dashed line circle) it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle).

k-nearest neighbor algorithm

Suppose we have pairs $(X_1, Y_1), \ldots, (X_n, Y_n)$ and we want to calculate the predicted value for x_0

- calculate the distance between each pair and x_0
- reorder the training data such that: $\|X_{(1)} - x_0\| \leq \cdots \leq \|X_{(n)} - x_0\|$
- select the first k data points (those with minimal distance)
- assign x_0 the value of the k-nearest (Y_i) values:
  - **classification**: assign the label which is most frequent among the k training samples nearest to x_0 (majority voting)
  - **regression** assign the predicted value to be the mean value of its k-neighbors

Under some circumstances, it can be advantageous to weight points such that nearby points contribute more to the regression than faraway points

k-nearest neighbors

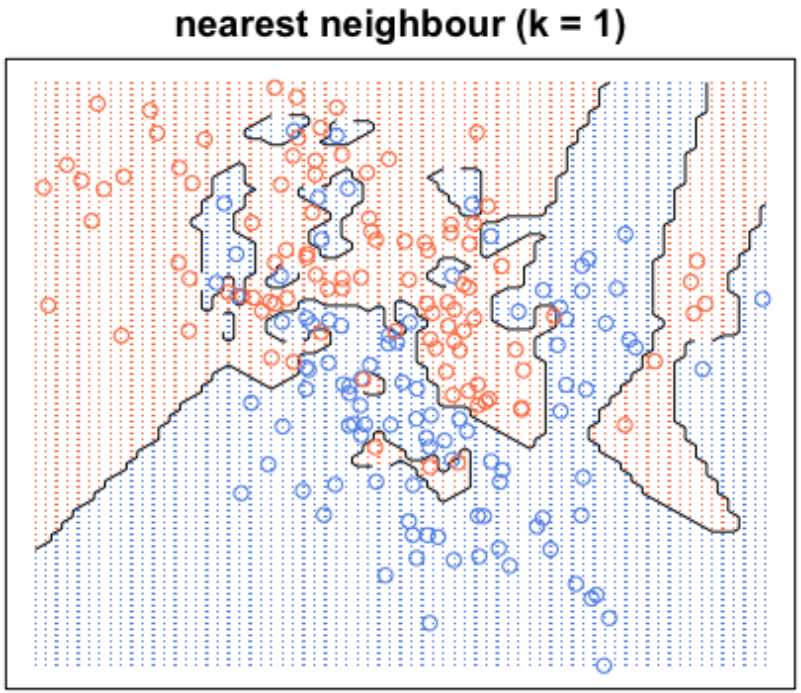


Figure: When k is small, we are restraining the region of a given prediction and forcing our classifier to be more blind to the overall distribution. A small value for k provides the most flexible fit, which will have low bias but high variance. Graphically, our decision boundary will be more jagged.

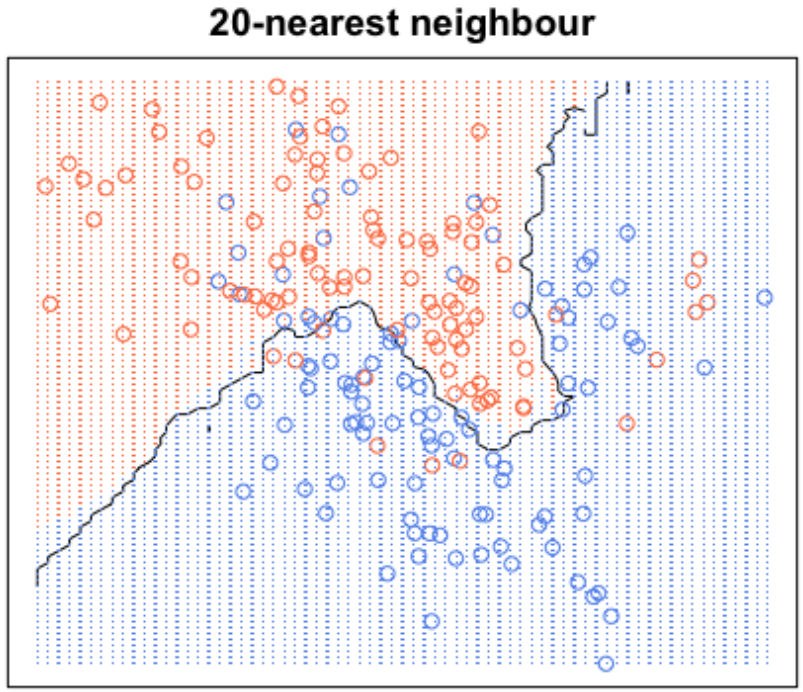


Figure: On the other hand, a higher k averages more voters in each prediction and hence is more resilient to outliers. Larger values of k will have smoother decision boundaries which means lower variance but increased bias

pros & cons of k-NN

- simple to understand and easy to implement
- it can be a useful tool for off-the-bat analysis of some data set you are planning to run more complex algorithms on
- computationally expensive testing phase which is impractical in industry settings (need to calculate a distance for each data point)
- Furthermore, KNN can suffer from skewed class distributions. For example, if a certain class is very frequent in the training set, it will tend to dominate the majority voting of the new example (large number = more common)

The End