Module 2, Part 1: Statistical Learning

TMA4268 Statistical Learning V2020

Stefanie Muff, Department of Mathematical Sciences, NTNU

January xx, 2020

Introduction

Learning material for this module

- James et al (2013): An Introduction to Statistical Learning. Chapter 2.
- Additional material (in this module page) on random variables, covariance matrix and the multivariate normal distribution (known for students who have taken TMA4267 Linear statistical models).

Some of the figures and slides in this presentation are taken (or are inspired) from “An Introduction to Statistical Learning, with applications in R” (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.

What will you learn?

- Statistical learning and examples thereof
- Introduce relevant notation and terminology.
- Prediction accuracy vs. model interpretability
- Bias-variance trade-off
- Classification (a first look):
  - The Bayes classifier
  - K nearest neighbour (KNN) classifier
- The basics of random vectors, covariance matrix and the multivariate normal distribution.

What is statistical learning?

Statistical learning is the process of learning from data.

By applying *statistical methods* on a *data set* (called the *training set*), we would like to *draw conclusions* about the relations between the variables (aka *inference*) or to *find a predictive function* (aka *predition*) for new observations.

Also, we would like to find structures in the data that help us to learn something about the real world.

Statistical learning plays a key role in many areas of science, finance and industry.

Variable types

Variables can be characterized as either *quantitative* or *qualitative.*

Quantitative variables are variables from a continuous set, they have a numerical value.

- Examples: a person's weight, a company's income, the age of a building, the temperature outside, the level of precipitation etc.

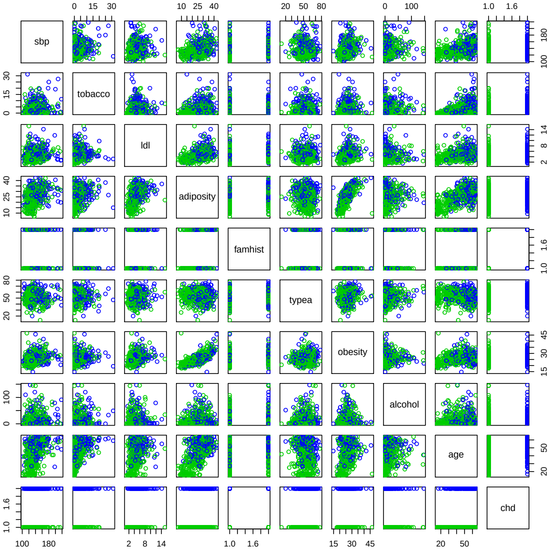
Qualitative variables are variables from a discrete set, from a set of K different classes/labels/categories.

- Examples: type of fruit {apples, oranges, bananas, ...}, sex {male, female, other }, education level.
- Qualitative variables which have only two classes are called *binary* variables and are usually coded by 0 (no) and 1 (yes).

Examples of learning problems

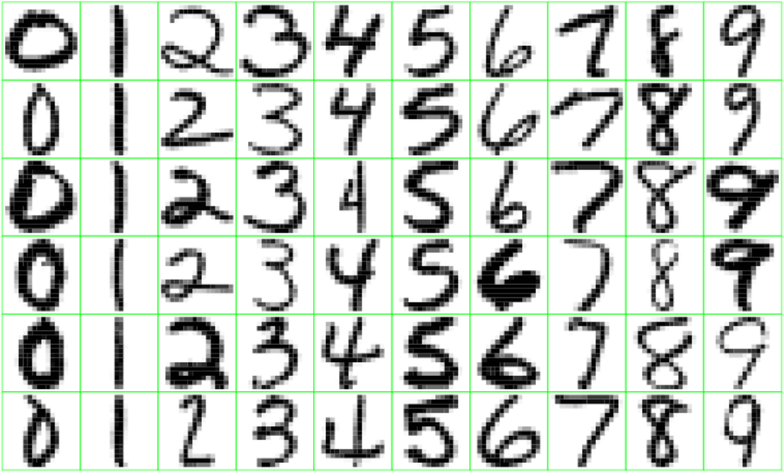
- To predict the price of a stock 3 months from now, based on company performance measures and economic data. Here the response variable is quantitative (price).
- Spam detection for emails.
- To identify the risk factors for Prostate cancer.
- To predict whether someone will suffer from a heart disease, given knowledge about condition, behaviour, age etc.
- To predict whether someone will suffer a heart attack on the basis of demographic, diet and clinical measurements. Here the outcome is binary (yes,no) with both qualitative and quantitative input variables.
- Predict soemeone's body fat, given BMI, weight, age, etc.

South African coronary heart disease data: 462 observations and 10 variables.



Handwritten digit recognition:

To identify the numbers in a handwritten ZIP code, from a digitized image. This is a classification problem, where the response variable is categorical with classes {0, 1, 2, ..., 9} and the task is to correctly predict the class membership.



Email classification (spam detection):

The goal is to build a spam filter. This filter can based on the frequencies of words and characters in emails. The table below show the average percentage of words or characters in an email message, based on 4601 emails of which 1813 were classified as a spam.

	you	free	george	!	$	edu
not spam	1.27	0.07	1.27	0.11	0.01	0.29
spam	2.26	0.52	0.00	0.51	0.17	0.01

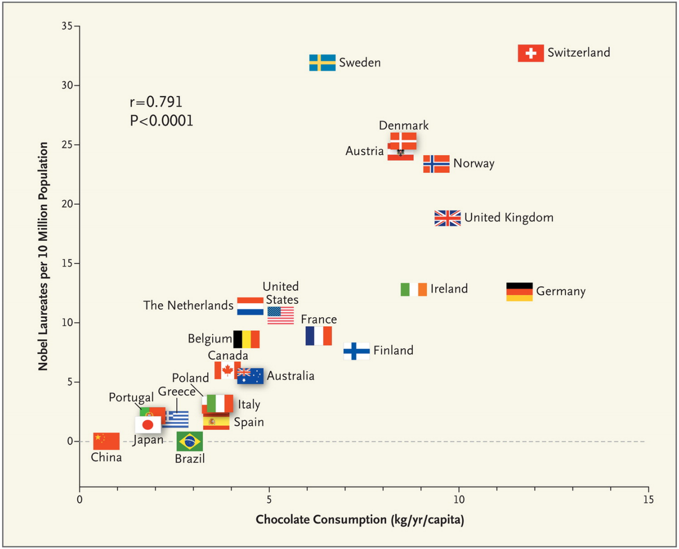
What makes a Nobel Prize winner?

Perseverance, luck, skilled mentors or simply chocolate consumption? An article published in the New England Journal of Medicine have concluded with the following:

> *Chocolate consumption enhances cognitive function, which is a sine qua non for winning the Nobel Prize, and it closely correlates with the number of Nobel laureates in each country. It remains to be determined whether the consumption of chocolate is the underlying mechanism for the observed association with improved cognitive function.*

The figure shows the number of Nobel Laureates per 10 million population against countries' annual per capita chocolate consumption.

You can read the article here and a informal review of the article here.



r=0.791
P<0.0001
Nobel Laureates per 10 Million Population
Chocolate Consumption (kg/yr/capita)
35
30
25
20
15
10
5
0
0
5
10
15
Sweden
Switzerland
Denmark
Austria
Norway
United Kingdom
Ireland
Germany
United States
The Netherlands
France
Belgium
Finland
Canada
Australia
Poland
Greece
Portugal
Italy
Spain
Japan
China
Brazil

Were there common underlying aims and elements of these examples of statistical learning?

- To predict the price of a stock 3 months from now, based on company performance measures and economic data.
- To identify the risk factors for developing diabetes based on diet, physical activity, family history and body measurements.
- The goal is to build a spam filter.
- To predict whether someone will suffer a heart attack on the basis of demographic, diet and clinical measurements.
- To identify the numbers in a handwritten ZIP code, from a digitized image.
- What makes a Nobel Prize winner? Perseverance, luck, skilled mentors or simply chocolate consumption?

The Supervised Learning Problem

Starting point:

- Outcome measurement Y (also called dependent variable, response, target).
- Vector of p predictor measurements X (also called inputs, regressors, covariates, features, independent variables).
- In the **regression problem**, Y is quantitative (e.g price, blood pressure).
- In the **classification problem**, Y takes values in a finite, unordered set (survived/died, digit 0-9, cancer class of tissue sample).
- We have training data $(x_1, y_1), \ldots, (x_N, y_N)$. These are observations (examples, instances) of these measurements.

Supervised learning and its objectives

Our data set (training set) consists of n measurement of the response variable Y and of p covariates x:

$$(y_1, x_{11}, x_{12}, \dots, x_{1p}), (y_2, x_{21}, \dots, x_{2p}), \dots, (y_n, x_{n1}, x_{n2}, \dots, x_{np}).$$

On the basis of the *training data* we would like to:

- **Accurately predict** unseen test cases.
- **Understand** which input affects the outcomes, and how.
- **Assess the quality** of your predictions and inference.

Supervised learning examples (we will study):

- Linear regression (M3), Logistic regression (M4), Generalized additive models (M7)
- Classification trees, bagging, boosting (M8), K-nearest neighbor classifier (M2, M4)
- Support vector machines (M9)

The Unsupervised Learning Problem

- There is **no outcome variable** y, just a set of predictors (features) x_i measured on a set of samples.
- Objective is more fuzzy – find (hidden) patterns or groupings in the data - in order to *gain insight and understanding*. There is no *correct* answer.
- Difficult to know how well your are doing.
- Different from supervised learning, but can be useful as a pre-processing step for supervised learning.

Examples in the course:

- Clustering (M10)
- Principal component analysis (M10)

Semi-supervised learning

(Not considered in this course)

- Our data set consists of a input data, and some of the data has labelled responses.
- This situation can for example occur if the measurement of input data is cheap, while the output data is expensive to collect.
- Classical solutions (likelihood-based) to this problem exists in statistics (missing at random observations).

Task

Find examples to explain the difference between supervised and unsupervised learning.

-
-
-

Overall philosophy

It is important to understand the ideas behind the various techniques, in order to know how and when to use them.

- One has to understand the simpler methods first, in order to grasp the more sophisticated ones.
- It is important to accurately assess the performance of a method, to know how well or how badly it is working

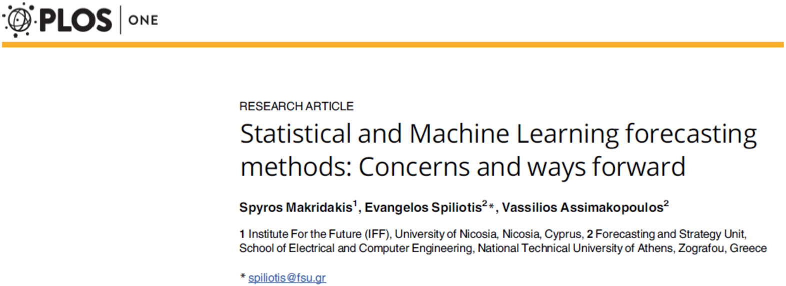
→ **Simpler methods often perform as well as fancier ones!**

- This is an exciting research area, having important applications in science, industry and finance.
- Statistical learning is a fundamental ingredient in the training of a modern data scientist.

Statistical Learning vs. Machine Learning

- Machine learning arose as a subfield of Artificial Intelligence.
- Statistical learning arose as a subfield of Statistics.
- There is much overlap — both fields focus on supervised and unsupervised problems:
    - Machine learning has a greater emphasis on large scale applications and prediction accuracy.
    - Statistical learning emphasizes models and their interpretability, and precision and uncertainty.
- The distinction has become more and more blurred, and there is a great deal of “cross-fertilization”.
- Machine learning has the upper hand in Marketing!

- There is a controversy and some scepticism against “too fancy” ML methods.
- Criticism: ML often re-invents existing methods and names them differently, but often without awareness of existing methods in statistics.
- Almost weekly new literature that delivers comparison. Often, the “simple” statistcal methods “win”.



PLOS | ONE

RESEARCH ARTICLE

Statistical and Machine Learning forecasting methods: Concerns and ways forward

Spyros Makridakis[1], Evangelos Spiliotis[2]*, Vassilios Assimakopoulos[2]

1 Institute For the Future (IFF), University of Nicosia, Nicosia, Cyprus, **2** Forecasting and Strategy Unit, School of Electrical and Computer Engineering, National Technical University of Athens, Zografou, Greece

* spiliotis@fsu.gr

What is the aim in statistical learning?

Assume:

- we observe one *quantitative* response Y and
- p different predictors $x_1, x_2, ..., x_p$.

We assume that there is a function f that relates the response and the predictor variables:

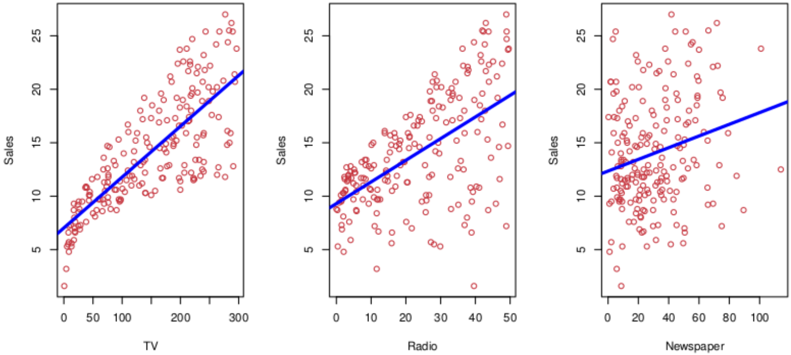
$$Y = f(x) + \varepsilon,$$

where ε is a random error term with mean 0 and independent of x.

The aim is to estimate f.

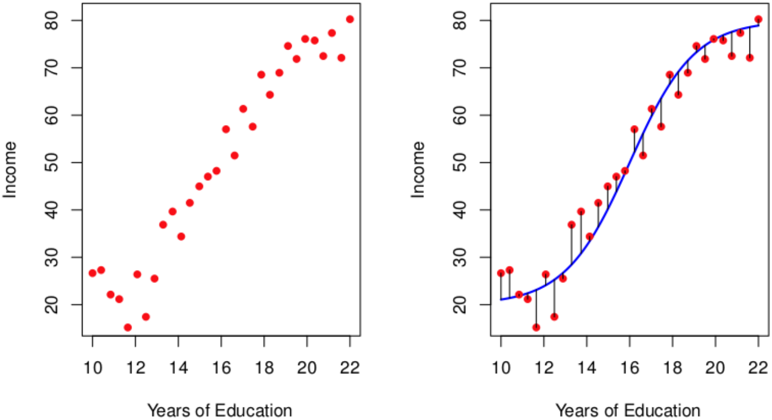
Example 1

Sales of a product, given advertising budgets in different media.



Example 2

Income for given levels of education.



There are two main reasons for estimating f:

- **Prediction**
- **Inference**

Reason 1: Prediction

Aim: predict a response Y given new observations x of the covariates as accurately as possible.

Notation:

$$\hat{Y} = \hat{f}(x).$$

- $\hat{f}$: estimated f
- $\hat{Y}$ prediction for Y given x.
- We do not really care about the shape of f ("black box"). → no interpretation of regression parameters when the aim is purely prediction!

There are two quantities which influence the accuracy of $\hat{Y}$ as a prediction of Y:

- The *reducible error* has to do with our estimate $\hat{f}$ of f. This error can be reduced by using the most *appropriate* statistical learning technique.
- The *irreducible error* comes from the error term ε and cannot be reduced by improving f. This is related to the unobserved quantities influencing the response and possibly the randomness of the situation.

For a given $\hat{f}$ and a set of predictors X which gives $\hat{Y} = \hat{f}(X)$, we have

$$\mathrm{E}[(Y - \hat{Y})^2] = \underbrace{\mathrm{E}[(f(X) - \hat{f}(X))^2]}_{reducible} + \underbrace{Var(\epsilon)}_{irreducible}$$

Q: If there were a *deterministic* relationship between the response and a set of predictors, would there then be both reducible and irreducible error?

Reason 2: Inference

Aim: *understand* how the response variable is affected by the various predictors (covariates).

The *exact form* of $\hat{f}$ is of *main interest*.

- Which predictors are associated with the response?
- What is the relationship between the response and each predictor?
- Can the relationship be linear, or is a more complex model needed?

Estimating f

Overall idea:

- Using available *training data* $(x_1, y_n), \ldots (x_n, y_n)$ to estimate $\hat{f}$, such that $Y \approx \hat{f}(X)$ for any (X, Y) (also those that have not yet been observed).

Two main approaches:

- Parametric methods
- Non-parametric methods

Parametric methods

Asumption about the form or shape of the function f.
The multiple linear model (M3) is an example of a parametric method:

$$f(x) = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p + \varepsilon,$$

with $\varepsilon \sim N(0, \sigma^2)$.

The task simplifies to finding estimates of the $p + 1$ coefficients $\beta_0, \beta_1, .., \beta_p$. To do this we use the training data to fit the model, such that

$$Y \approx \hat{\beta}_0 + \hat{\beta}_1 x_1 + ... + \hat{\beta}_p x_p.$$

Fitting a parametric models is thus done in two steps:

1. Select a form for the function f.
2. Estimate the unknown parameters in f using the training set.

Non-parametric methods

- Non-parametric methods seek an estimate of f that gets close to the data points, but without making explicit assumptions about the form of the function f.
- The K-nearest neighbour algorithm is an example of a non-parametric model. Used in classification, this algorithm predicts a class membership for a new observation by making a majority vote based on its K nearest neighbours. We will discuss the K-nearest neighbour algorithm later in this module.

Q: What are advantages and disadvantages of parametric and non-parametric methods?

Hints: interpretability, amount of data needed, complexity, assumptions made, prediction accuracy, computational complexity, over/under-fit.

A: Parametric methods

Advantages	Disadvantages
Simple to use and easy to understand	The function f is constrained to the specified form.
Requires little training data	The assumed function form of f will in general not match the true function, potentially giving a poor estimate.
Computationally cheap	Limited flexibility

A: Non-parametric methods

Advantages	Disadvantages
Flexible: a large number of functional forms can be fitted	Can overfit the data
No strong assumptions about the underlying function are made	Computationally more expensive as more parameters need to be estimated
Can often give good predictions	Much data is required to estimate (the complex) f.

Prediction accuracy vs. interpretability

(we are warming up to the bias–variance trade–off)

Inflexible methods:

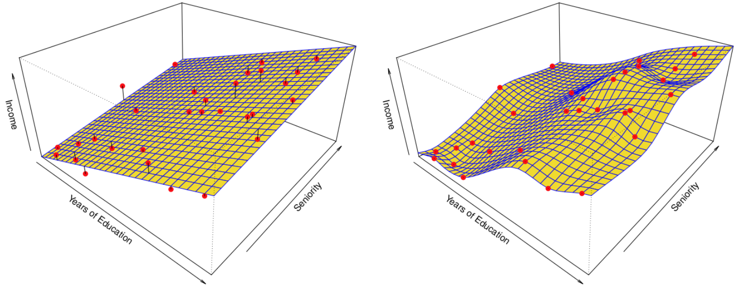
- Linear regression (M3)
- Linear discriminant analysis (M4)
- Subset selection and lasso (M6)

Flexible methods:

- KNN classification (M2, M4), KNN regression, Smoothing splines (M7)
- Bagging and boosting (M8), support vector machines (M9)
- Neural networks (M11)

Why would I ever prefer an inflexible method?

Example: Prediction of icome from “Years of Education” and “Seniority”



The choice of a flexible or inflexible method depends on the goal in mind.

- For **inference** an inflexible model is easier to interpret.
- For **prediction** a flexible model is more powerful.

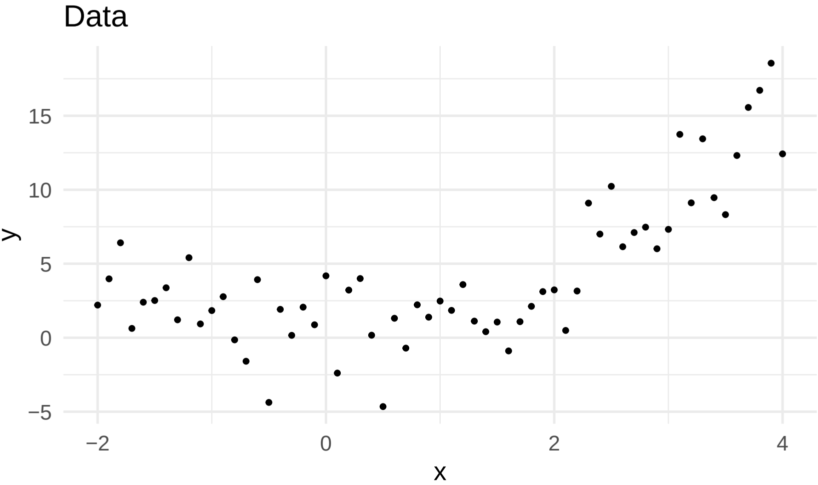
Overfitting occurs when the estimated function f is too closely fit to the observed data points.

Underfitting occurs when the estimated function f is too rigid to capture the underlying structure of the data.

We illustrate this by a toy example using polynomial regression.

Polynomial regression example (simulation)

Consider a covariate x observed between -2 to 4 and $n = 61$ observations.

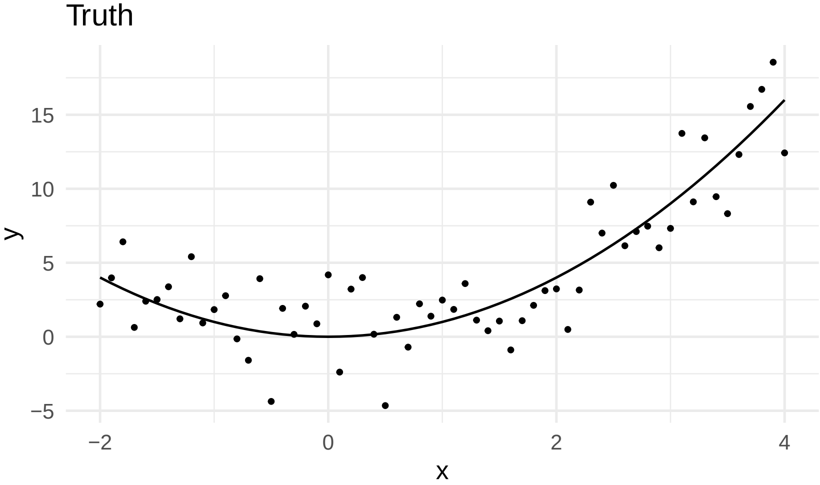


We impose a relationship between reponse Y and covariate x:

$$Y = x^2 + \varepsilon$$

with error (noise) term $\varepsilon \sim N(0, \sigma^2)$ with $\sigma = 2$. It is a substitue for all the unobserved variables that are not in our equation, but that might influence Y.

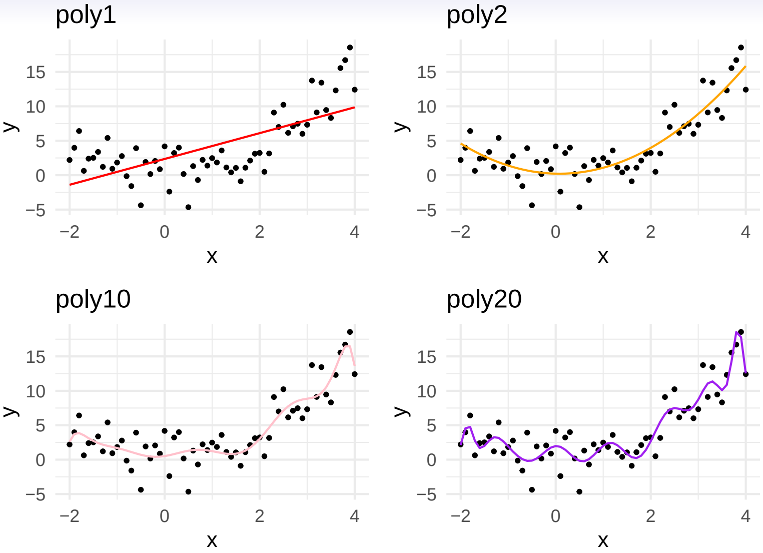
We call $Y = x^2$ the *truth*.


Truth
y
15
10
5
0
−5
−2
0
2
4
x

Next, we want to fit a function to the observations *without* knowing the true relationship, and we have tried different parametric polynomial functions.

- **poly1**: Simple linear model of the form $\beta_0 + \beta_1 x$ fitted to the observations. *Underfits* the data.
- **poly2**: Quadratic polynomial fit to the data, of the form $\beta_0 + \beta_1 x + \beta_2 x^2$. This fits well.
- **poly10**: Polynomial of degree 10 fit of the form $\beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_{10} x^{10}$ *Overfits* the data.
- **poly20**: Polynomial of degree 10 fit of the form $\beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_{20} x^{20}$ *Overfits* the data.

We will discuss polynomial regression in M7.



The degree of the polynomial is a *flexibility parameter*.

We can now ask:

- Which of these models performs “best”?
- Is there *one* method that dominates all others?

Assessing model accuracy

No method dominates all others over all possible data sets.

- That is why we need to learn about many different methods.
- For a given data set we need to know how to decide which method produces the *best* results.
- We need to understand what *best* means.
- How close is the predicted response to the true response value?

Measuring the Quality of Fit

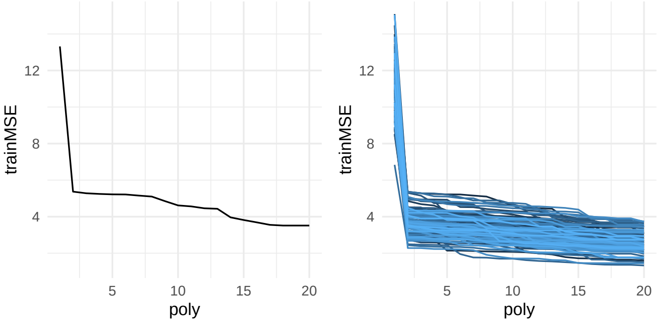
- A popular measure for the quality of fit: *Training MSE* (mean squared error).
- It is the mean of squared differences between prediction and truth for the training data (the same values that were used to estimate f):

$$\text{MSE}_{\text{train}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

- But: we are *not* interested in how the method works on the training data. We want to know how good the method is when we use it on *previously unseen test data* (e.g., future data).

Examples:

- We don’t want to predict last weeks stock price, we want to predict the stock price next week.
- We don’t want to predict if a patient in the training data has diabetes (because we already know this), we want to predict if a new patient has diabetes.



Q: Based on the training MSE - which model fits the data the best?

Polynomial example: fitted order 1-20 polynomial when the truth is order 2. Left: one repetition, right: 100 repetitions of the training set.

Test MSE

- Simple solution: estimate $\hat{f}$ using the training data (maybe my minimizing the training MSE), but choose the *best* model using a separate *test set*.
- *Test MSE* for a set of n_0 test observations (x_{0j}, y_{0j}):

$$\text{MSE}_{\text{test}} = \frac{1}{n_0} \sum_{j=1}^{n_0} (y_{0j} - \hat{f}(x_{0j}))^2$$

- Alternative notation:

$$\text{Ave}(y_0 - \hat{f}(x_0))^2$$

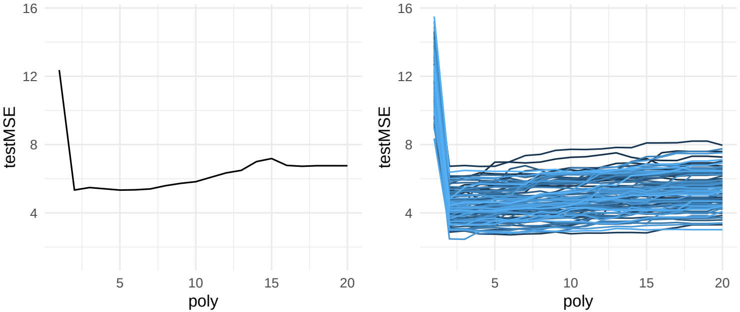
(taking the average over all available test observations).

Q: What if we do not have access to test data?

A: In Module 5 we will look into using *cross validation* to mimic the use of a test set.

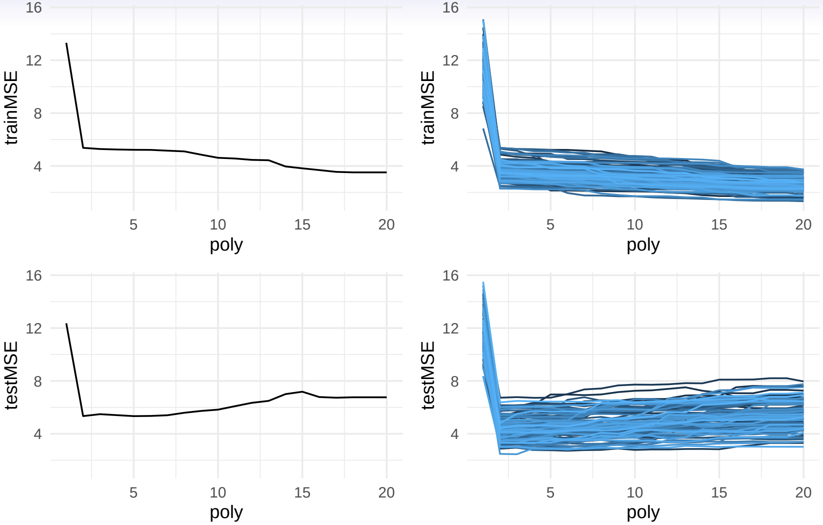
Q: But, can we instead just use the training data MSE to choose a model? A low training error should also give a low test error?

A: Sadly no, if we use a flexible model we will look at several cases where a low training error is a sign of overfitting, and will give a high test error. So, the training error is not a good estimator for the test error because it does not properly account for model complexity.



Polynomial example: fitted order 1-20 when the truth is order 2. Left: one repetition, right: 100 repetitions for the testMSE.

Q: Based on the test MSE - which model fits the data the best?



A: If choosing flexibility based on training MSE=poly20 wins, if choose flexibility based on test MSE=poly 2 wins.

Test error vs. training error

Important observations:

- The test error seems to have a minimum (U-shape) in between the extremes.
- The training error keeps going down.

Why?

The Bias-Variance trade-off

- Assume we have fitted a *regression* curve $Y = f(x) + \varepsilon$ to our training data, which consist of independent observation pairs $\{x_i, y_i\}$ for $i = 1, .., n$. (Yes, only one covariate x.)
- Assume that ε is an unobserved random variable that adds noise to the relationship between the response variable and the covariates (random error), with mean zero and constant variance σ^2 for all values of x.
- ε is a substitute for all the unobserved variables that influence Y.
- The training data was used to estimate $\hat{f}$.

- We want to use $\hat{f}$ to obtain the predicted response value $\hat{f}(x_0)$ for an unseed test observation (x_0, y_0).
- The *expected test mean squared error (MSE) at* x_0 is defined as:

$$\mathrm{E}[y_0 - \hat{f}(x_0)]^2$$

- Compare this to the test MSE for the polynomical example ($\mathrm{MSE}_{\mathrm{test}}$): The average is simply replaced by the *theoretical version* (expected value).

Using that $y_0 = f(x_0) + \varepsilon$, this expected test MSE can be decomposed into three terms

$$\begin{aligned}
&\mathrm{E}[f(x_0) + \varepsilon - \hat{f}(x_0)]^2 \\
&= \mathrm{E}[f(x_0)^2] + \mathrm{E}[\varepsilon^2] + \underbrace{\mathrm{E}[\hat{f}(x_0)^2]}_{=f(x_0)^2} - 2\mathrm{E}[f(x_0)\hat{f}(x_0)] \\
&= \mathrm{Var}(f(x_0)) + \mathrm{E}[f(x_0)^2] + \mathrm{Var}(\varepsilon) + \mathrm{Var}(\hat{f}(x_0)) + \mathrm{E}[\hat{f}(x_0)]^2 \\
&\qquad - 2f(x_0)\mathrm{E}[\hat{f}(x_0)] \\
&= \underbrace{\mathrm{Var}(\varepsilon)}_{\text{Irreducible error}} + \underbrace{\mathrm{Var}(\hat{f}(x_0))}_{\text{Variance of prediction}} + \underbrace{\Big(f(x_0) - \mathrm{E}[\hat{f}(x_0)]\Big)^2}_{\text{Squared bias}}
\end{aligned}$$

Q: what assumptions have we made in the derivation above?

A: classnotes.

$$\mathrm{E}[(y_0 - \hat{f}(x_0))^2] = \cdots = \mathrm{Var}(\varepsilon) + \mathrm{Var}(\hat{f}(x_0)) + [\mathrm{Bias}(\hat{f}(x_0))]^2$$

- First term: irreducible error. This term cannot be reduced regardless how well our statistical model fits the data.
- Second term: variance of the prediction at x_0 or the expected deviation around the mean at x_0. If the variance is high, there is large uncertainty associated with the prediction.
- Third term: squared bias. The bias gives an estimate of how much the prediction differs from the true mean. If the bias is low the model gives a prediction which is close to the true value.

Note:

$$\mathrm{E}[(y_0 - \hat{f}(x_0))^2]$$

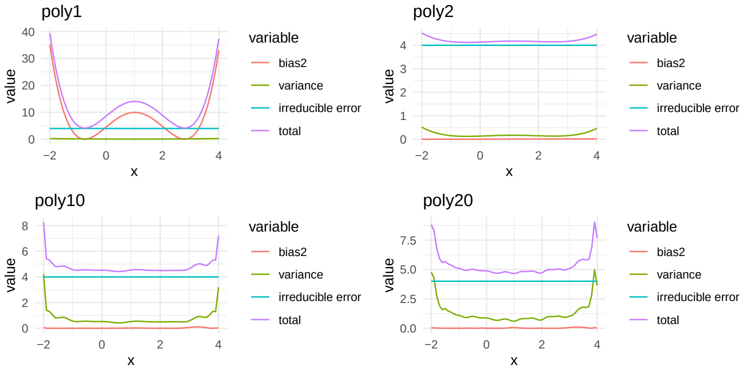
is the **expected test MSE**. We can think of this as the average test MSE we would obtain if we repeatedly estimated f using many training sets (as we did in our example), and then tested this estimate at x_0.

However, if we also assume that X is a random variable, this is really $\mathrm{E}[(Y - \hat{f}(x_0))^2 \mid X = x_0]$

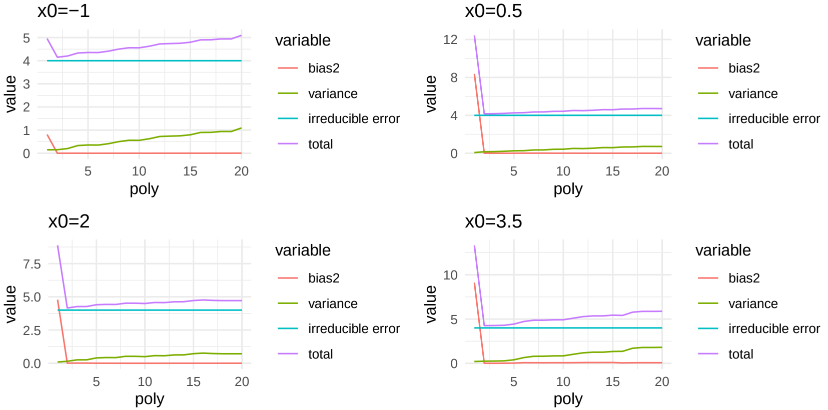
The **overall expected test MSE** can we then compute by averaging the expected test MSE over all possible values of x_0 (averaging with respect to frequency in test set), or mathematically by the law of total expectation $\mathrm{E}\{\mathrm{E}[(Y - \hat{f}(X))^2 \mid X]\}$ (also sometimes referred to as the law of double expectations).

Polynomial example (cont.)

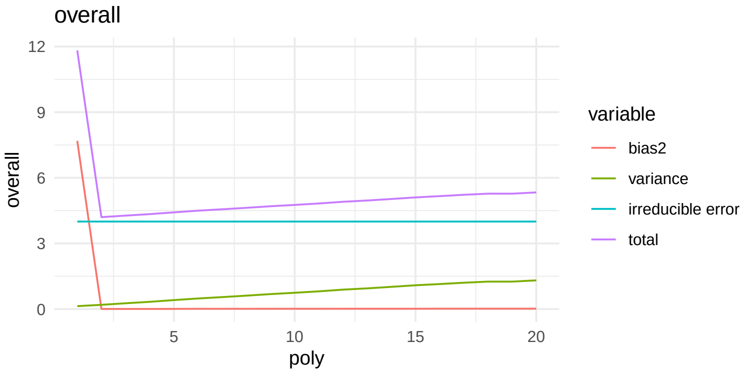
$Y = x^2$ is still the *truth*.



For 4 different polynomial models (poly1,2,10 and 20), the squared bias, variance, irreducible error and the total sum. Plots based on 100 simulations for the polynomial example.



At 4 different values for x_0, the squared bias, variance, irreducible error and the total sum. Plots based on 100 simulations for the polynomial example.

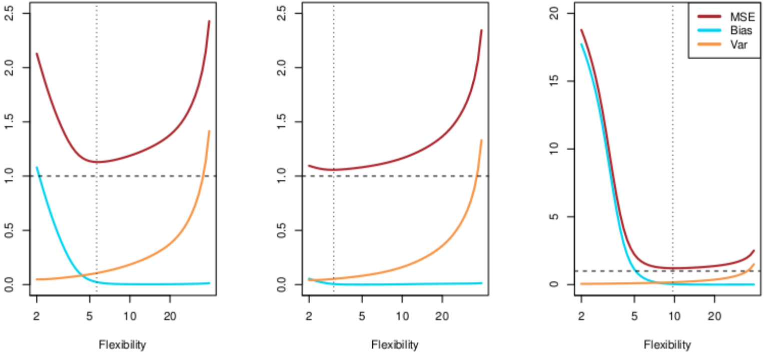


Overall version (averaging over 61 gridpoints of x).

Choosing the best model

When fitting a statistical model the aim is often to obtain **the most predictive model**.

- **Training set**: The observations used to fit the statistical model → Training error
- **Test sample**: new observations which were not used when fitting the model → Test error
- Training error decreases for more complex models, but the test error has an **optimum**.
- This trade-off in selecting a model with the right amount of complexity/flexibility is the **Bias-Variance trade-off**.



- Inflexible models may lead to a poor fit (high bias)
- Flexible (complex) models may provide more unbiased fits but may overfit the data (high variance)
- The aim is to find the optimum.

Classification (ev to be removed, with reference to see Module 4!!)

How about model accuracy in classification?

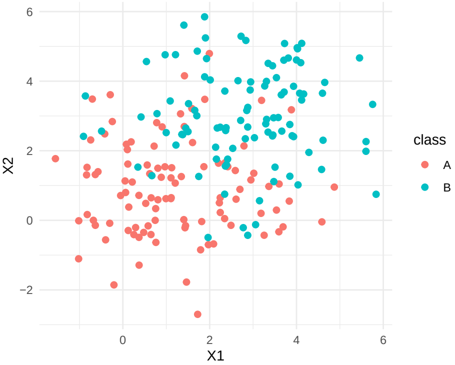
Set-up: Training observations $\{(x_1, y_1), ..., (x_n, y_n)\}$ where the response variable Y is categorical, e.g $Y \in \mathcal{C} = \{0, 1, ..., 9\}$ or $Y \in \mathcal{C} = \{dog, cat, ..., horse\}$.

Aim: To *build* a classifier $f(x)$ that assigns a class label from $\mathcal{C}$ to a future unlabelled observation x and to asses the *uncertainty* in this classification.

Performance measure: Most popular is the misclassification error rate (training and test version).

Synthetic example

- Simulate 2×100 observations from a bivariate normal distribution with mean vectors $\mu_A = (1,1)^T$, $\mu_B = (3,3)^T$, and covariance matrix $\Sigma_A = \Sigma_B = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$.
- Aim: Find a rule to classify a new observation to A or B.



- **Training error rate**: The proportion of mistakes that are made if we apply our estimator $\hat{f}$ to the training observations, i.e. $\hat{y}_i = \hat{f}(x_i)$

$$\frac{1}{n}\sum_{i=1}^{n} \mathrm{I}(y_i \neq \hat{y}_i) \ ,$$

with indicator function I, which is defined as:

$$\mathrm{I}(a \neq \hat{a}) = \begin{cases} 1 \text{ if } a \neq \hat{a} \ , \\ 0 \text{ else.} \end{cases}$$

- The indicator function counts the number of times our model has made a wrong classification. The training error rate is the fraction of misclassifications made on our training set.
- A very low training error rate may imply overfitting.

- **Test error rate**: The fraction of misclassifications when our model is applied on a test set

$$\mathrm{Ave}(I(y_0 \neq \hat{y}_0)) \ ,$$

  where the average is over all the test observations (x_0, y_0).
- Again, this gives a better indication of the true performance of the classifier (than the training error).
- We assume that a *good* classifier is a classifier that has a *low* test error.

Bayes classifier

- The *Bayes classifier assigns an observation to the most likely class*, given its predictor values.
- Suppose we have a quantitative response value that can be a member in one of K classes $\mathcal{C} = \{c_1, c_2, ..., c_k, ..., c_K\}$. Further, suppose these elements are numbered $1, 2, ..., K$. The probability of that a new observation x_0 belongs to class k is

  $$p_k(x_0) = \Pr(Y = k | X = x_0), \quad k = 1, 2, ...K.$$

  This is the conditional class probability: the probability that $Y = k$ given the observation x_0.
- Two-class example for two groups $\{A, B\}$. A new observation x_0 will be classified to A if $\Pr(Y = A | X = x_0) > 0.5$ and to class B otherwise.

Properties of the Bayes classifier

- It has the *smallest test error rate*.
- The class boundaries using the Bayes classifier is called the *Bayes decision boundary*.
- The overall Bayes error rate is given as

  $$1 - \mathrm{E}(\max \Pr(Y = j \mid X))$$

  where the expectation is over X.
- The Bayes error rate is comparable to the *irreducible error* in the regression setting.
- Caveat: we never (or very seldom) know the conditional distribution of Y given X for real data → The K-nearest neighbor classifier estimates this conditional distribution and then classifies a new observation based on this estimated probability.

K-nearest neighbour classifier

The K-nearest neighbour classifier (KNN) works in the following way:

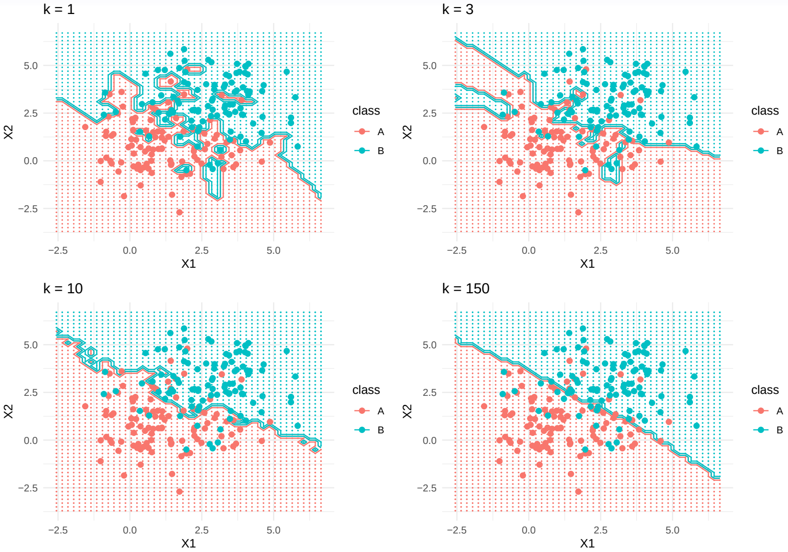
- Given a new observation x_0 it searches for the K points in our training data that are closest to it (Euclidean distance).
- These points make up the neighborhood of x_0, $\mathcal{N}_0$.
- The point x_0 is classified by taking a majority vote of the neighbors.
- That means that x_0 is classified to the most occurring class among its neighbors

$$\Pr(Y = j | X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j).$$

We return to our synthetic data with X_1 and X_2 and two classes A and B:

- Assume we have a new observation $X_0 = (x_{01}, x_{02})^T$ which we want to classify as belonging to the class A or B.
- To illustrate this problem we fit the K-nearest neighbor classifier to our simulated data set with $K = 1, 3, 10$ and 150.

The small colored dots show the predicted classes for an evenly-spaced grid. The lines show the decision boundaries. If our new observation falls into the region within the red decision boundary, it will be classified as A. If it falls into the region within the green decision boundary, it will be classified as B.



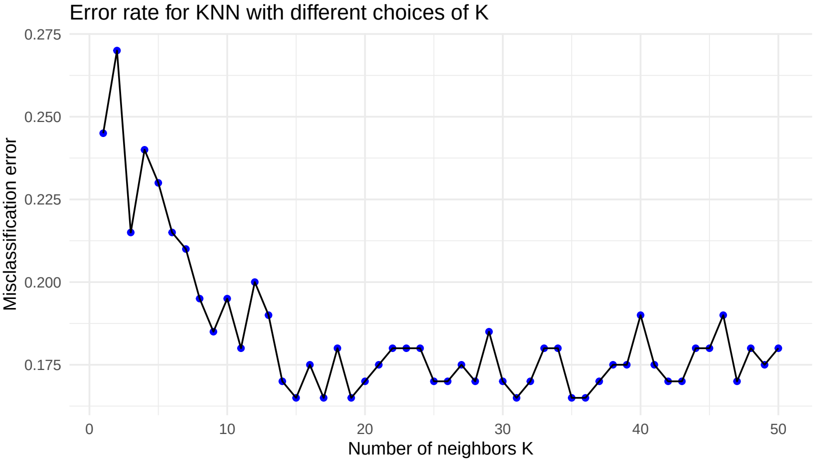
k = 1

k = 3

k = 10

k = 150

- The choice of K has a big influence on the result of our classification. For $K = 1$ the classification is made to the same class as the one nearest neighbor. As K gets very large, the decision boundary tends towards a straight line (which is the Bayes boundary in this set-up).
- To find the optimal value of K the typical procedure is to try different values of K and then test the predictive power of the different classifiers, for example by cross-validation, which will be discussed in M5.
- After trying all choices for K between 1 and 50, a few choices of K gave the smallest misclassification error rate, estimating by leave-one out cross-validation (see M5). The smallest error rate is equal to 0.165 (16.5% misclassification).


Error rate for KNN with different choices of K
Misclassification error
0.275
0.250
0.225
0.200
0.175
0
10
20
30
40
50
Number of neighbors K

→ Bias-variance trade-off in a classification setting:

- A too low value of K will give a very flexible classifier (with high variance and low bias) which will fit the training set too well (it will overfit) and make poor predictions for new observations.
- Choosing a high value for K makes the classifier loose its flexibility and the classifier will have low variance but high bias.

The curse of dimensionality

- The nearest neighbor classifier can be quite good if the number of predictor p is small and the number of observations n is large. We need enough close neighbors to make a good classification.
- The effectiveness of the KNN classifier falls quickly when the dimension of the preditor space is high.
- Why? Because the nearest neighbors tend to be far away in high dimensions and the method no longer is local. This is referred to as the *curse of dimensionality*.

What was important in Part A?

- prediction vs. interpretation (inference)
- supervised vs. unsupervised methods
- classification vs. regression
- parametric vs. non-parametric methods
- flexibility vs. interpretation
- under- and overfitting
- quadratic and 0/1 loss functions
- training and test MSE and misclassification error
- bias-variance trade off
- Bayes classifier and KNN-classifier

R packages

If you want to look at the .Rmd file and `knit` it, you need to first install the following packages (only once).

```r
install.packages("knitr")
install.packages("kableExtra")
install.packages("rmarkdown")
install.packages("devtools")
install.packages("ggplot2")
install.packages("ggpubr")
install.packages("dplyr")
install.packages("reshape2")
install.packages("ElemStatLearn")
install.packages("GGally")
install.packages("class")
install.packages("mvtnorm")
install.packages("MASS")
install.packages("car")
install.packages("faraway")
```

Acknowledgements

Thanks to Julia Debik for contributing to this module page.