DATA11002 Introduction to Machine Learning

Kai Puolamäki

30 October 2020

Announcements

- *Exercise Set 0 (E0)* submission **DL today**!
    - graded pass/fail (if you fail, next DL will be on 6 November)
    - we are continuously grading the submissions so please don't wait until the DL!
- *Exercise Set 1 (E1)* submission DL will be on **8 November**
    - please use, e.g., R or Jupyter notebook for Exercise Set 1-3 answers! (you can easily combine include text, mathematical formulas, code, and plots with them)
- *Term project groups* of 1-3 students will be formed after **9 November**
    - you can **form groups of 1-3 students** on your own before 9 November (we'll give you details how to tell us about your group next week)
    - if you don't form a group on your own you will be assigned to groups of (mostly) 3 students in random after 9 November
    - you can be assigned to a term project group **only after passing E0**

Issues: parable of big data

Estimating prevalence of flu based on search engine usage.

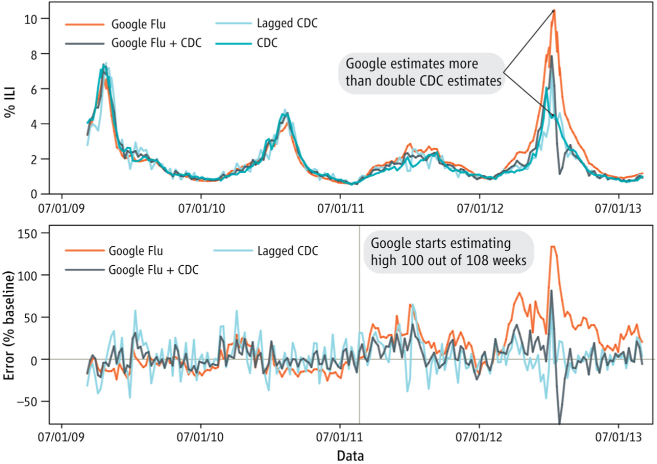


Figure from Lazer et al. (2014) Science

Managing data and tools

Data wrangling challenges

- Claim: 80% of the work on data mining project is about data understanding and data preparation
- See, e.g., Cross Industry Standard Process Data Mining (CRISP-DM)

Data wrangling challenges

- **DP** Data parsing, e.g., converting csv's or tables
- **DD** Obtaining (or inferring) a data dictionary: basic types + semantics
- **DI** Data integration: Combining data from multiple sources
- **ER** Entity resolution: Recognising that two distinct pieces of information in the data concern the same entity. Includes duplication and record linkage
- **FV** Format variability: e.g. for dates, but also for variability in names (e.g. IBM, I.B.M.).
- **SV** Coping with structural variability in the data, e.g. wide vs tall format. Also variation over time.
- **MD** Identifying and repairing missing data
- **AD** Anomaly detection and repair

List: Chris Williams

“Tools” for machine learning

- One of the major developments during last decades is introduction of software tools to implement AI methods
  - PhD no longer needed to run a complex algorithms (deep learning etc.)
  - PhD may still be needed to understand what happens
- Scientific publications often include open source libraries to implement the methods
- Take-away:
  - use these tools
  - contribute to these tools
  - recognise more the general computational problems and solve them instead of separately. tacking specific problems
- Examples: Keras for deep learning, Stan for probabilistic reasoning, R for statistical analysis etc.

Statistics on tools

- Top-3 data science tools (all used by over half of data scientists):
  - SQL
  - Python
  - R
  - Source: Data Science Salary Survey 2017
- The list might be different after 10 years
  - You should be able to switch between tools and recognize their strengths and weaknesses!

Looking at the data and communicating results

- Look at the data
- Understand what you try to do
  - First apply the dummy method
  - Apply the absolutely simplest non-trivial method
  - Only then try the advanced approaches!
- In data science communication is the key
  - Write a report of the above which includes figures and explanation of what you have done
  - Include your name on the report
  - It is not enough just to show random plots to confused audience!

Example

- See `aktia.zip` in Moodle!

Ingredients of machine learning

Ingredients of machine learning

- *Task* is what an end user actually wants to do.
- *Computational problem* is a (hopefully general) mathematical definition of the task
    - usually includes some *performance measure* (to be optimised somehow)
- *Model* is a (hopefully good) solution to the computational problem.
- *Data* consists of objects in the domain the user is interested in, with perhaps some additional information attached.
- *Machine learning algorithm* produces a model based on data.
- *Features* are how we represent the objects in the domain.

Task

- Task is an actual data processing problem some end user needs to solve.
- Examples were given in earlier (playing go, finding groups of mammals, estimating flu epidemics. . .)
- Typically, a task involves getting some input and then producing the appropriate output.
- For example, in hand-written digit recognition, the input is a pixel matrix representing an image of a digit, and the output is one of the labels '0', . . . , '9'.
- Machine learning is a way to find a solution for this data processing problem when it's too complicated or poorly understood for a programmer (or an application specialist) to figure out.

Computational problem

- Usually, task can be expressed as a instance of a **computational problem**
- Computational problems are more generic than tasks
- Often, the computational problem is some kind of a optimisation problem
- In this course we focus on a couple of common and generic computational problems
- When faced with a practical ML problem, always think the corresponding computational problem!
  - Then choose a model and make the algorithm to solve the computational problem (not practical task!)
- Wrong way:
  - Start hacking the model and algorithm and hope the best!

Supervised learning

Many common tasks belong to the area of *supervised learning* where we neeed to produce some target value (often called *label*):

- binary classification: divide inputs into two categories
  - *Example:* classify e-mail messages into spam and non-spam
- multiclass classification: more than two categories
- *Example:* classify an image of a digit into one of the classes '0', . . . , '9'

Supervised learning

- multilabel classification: multiple classes, of which more than one may match simultaneously
  - *Example:* classify news stories based on their topics
- regression: output is a real-valued
  - *Example:* predict the value of a house based on its size, location etc.

Supervised learning as a computational problem

Task: predict a value y given some *covariates* x given a (training) data which contains pairs of (x, y).

- *Training set*, $D_{tr} = \{(x_i, y_i)\}_{i=1}^{n}$ of n points drawn i.i.d. from fixed but unknown distribution F.
- *Hypothesis class*, the set of models (here functions) $\hat{y} = f(x)$ to predict y, given x.
- *Loss function*, $L(\hat{y}, y)$ difference between the output of model $\hat{y}$ and the desired output y.

Computational problem: given the training set D_{tr}, the hypothesis class, and a loss function, find a function $\hat{f}$ such that the expected loss on the data drawn from F is minimized, i.e.,

$$\hat{f} = \arg\min_f E_{(x,y)\sim F}(L(f(x), y))).$$

NB: "Learning is not possible without some assumptions". No free lunch teorems. E.g., Wolpert et al. 1997.

Unsupervised learning

In unsupervised learning, there is no specific target value of interest.

Examples of unsupervised learning tasks:

- *clustering:* partition the given set of data into *clusters* so that elements that belong to same cluster are similar (in terms of some given similarity measure)
- *dimensionality reduction:* if the data is high-dimensional (each data point is described by a large number of variables), find an alternative lower-dimensional representation that retains as much of the structure as possible
- *association rules:* given shopping cart contents of different customers, find product combinations often bought together

Unsupervised learning as a computational problem

- *clustering (k-means):* Given a constant k and data $D = \{X_i\}_{i=1}^n$, where $X_i \in \mathbb{R}^d$, find k *cluster centroids* $\overline{X}_j$, where $j \in \{1, \ldots, k\}$, such that the loss

  $$\sum\nolimits_{i=1}^n \min_{j \in \{1,\ldots,k\}} \left| X_i - \overline{X}_j \right|^2 / n$$

  is minimised.
  - Each the data item X_i will be assigned to cluster indexed by $\arg\min_{j \in \{1,\ldots,k\}} \left| X_i - \overline{X}_j \right|^2$.

Unsupervised learning as a computational problem

- *dimensionality reduction (PCA):* Find a unit vector $V \in \mathbb{R}^d$ such that the variance

  $$\sum_{i=1}^{n} \left(X_i^T V \right)^2 / n$$

  is maximised.
  - Data item $X_i \in \mathbb{R}^d$ is projected into $P_i \in \mathbb{R}$ by $P_i = X_i^T V$.

Reinforcement learning

In reinforcement learning, the learning algorithm **can interact with its environment** and learn from rewards it receives.

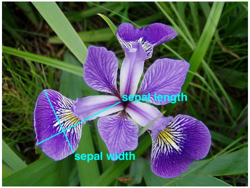
- Main idea: A learning *agent* is in some state s, and carries out action a, which takes the agent to state s'. (e.g., carries out some specific move in Chess)
- After taking some number of steps the agent receives a *reward*. (e.g., takes the queen of the opponent)
- The agent aims to *learn a policy* that maps states to actions so that it's total reward is maximised.
- Can be thought of as a type of supervised learning, where the target value is specified indirectly through the rewards, possibly with a temporal delay.
- Interesting stuff, but not covered in this course.

Predictive vs. descriptive model

- **Predictive model:** our goal is generalization, i.e., predict outcomes in future data
- **Descriptive model:** no guarantees about generalizaton to future data, we want to understand the data (a.k.a. exploratory analysis, data mining)
- Distinction between supervised vs. unsupervised learning is whether the target values are available to the learning algorithm: both can be either predictive or descriptive
- Keep in mind: **Look at the data first!** It is very easy to screw things up by blindly fitting models to data without slowing down and taking a closer look.

Supervised learning

Example of supervised learning: morphology of two species of Iris flowers

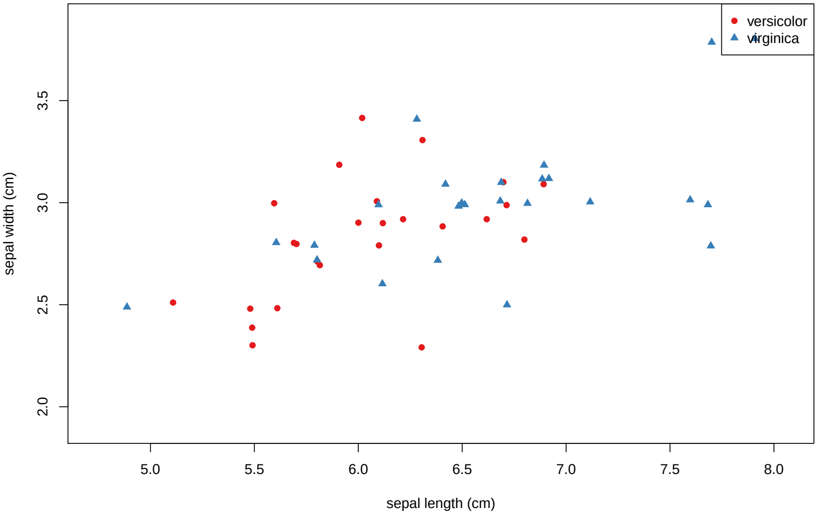


Iris versicolor (left) and iris virginica (right).

```
##   sepal_length sepal_width    species
## 1     6.304882    2.291057 versicolor
## 2     5.789014    2.791309  virginica
## 3     6.097653    2.989641  virginica
## 4     6.713670    2.988024 versicolor
## 5     6.099598    2.790570 versicolor
```

[45 rows omitted]

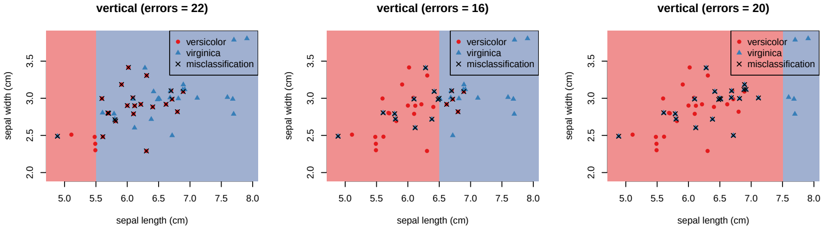
Example of supervised learning: morphology of two species of Iris flowers



Hypothesis class “vertical”

$$\hat{f}(x \mid \theta) = \begin{cases} \text{versicolor} & , \quad \text{sepal length} < \theta \\ \text{virginica} & , \quad \text{sepal length} \geq \theta \end{cases}$$

- Set of models/functions for all values of θ defines a hypothesis class.
- Let the error be number of misclassifications (*0/1 loss*) and find the model with smallest error.



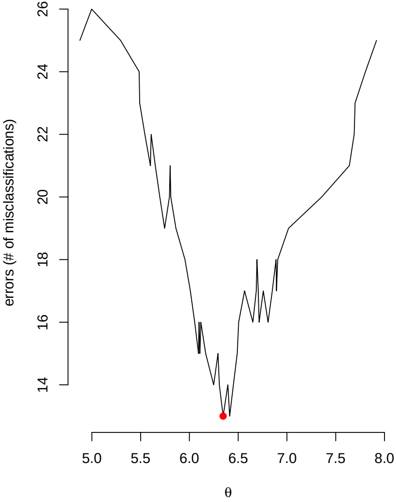
vertical (errors = 22)

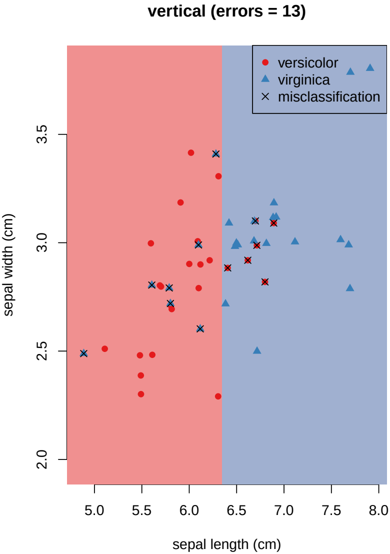
vertical (errors = 16)

vertical (errors = 20)

Hypothesis class “vertical”

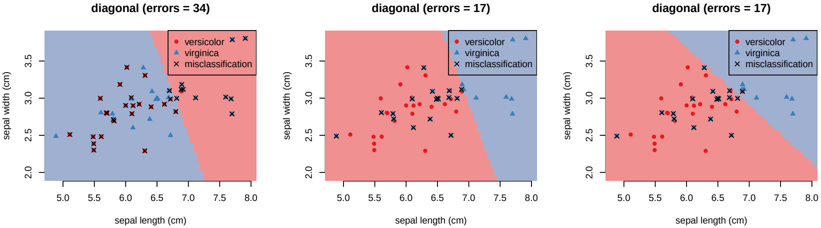
Find the model with smallest error.




vertical (errors = 13)

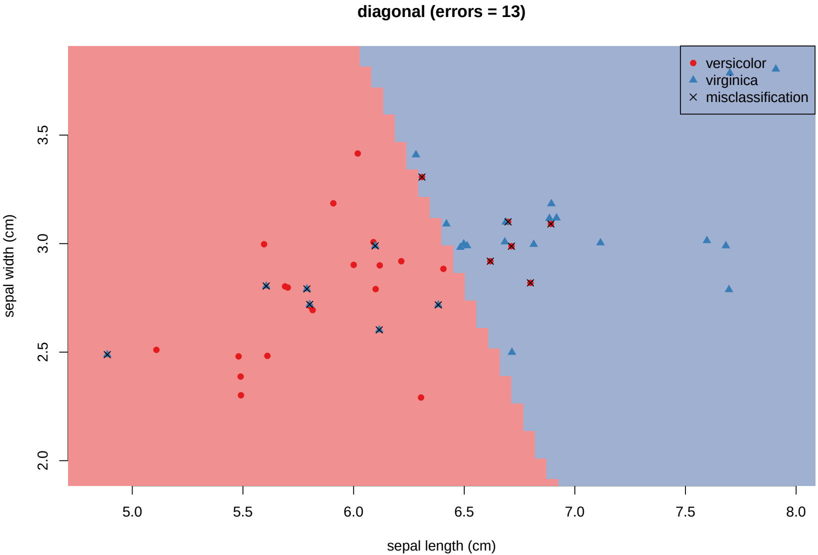
Hypothesis class "diagonal"

Hypothesis class with a *diagonal* class boundary:



Hypothesis class “diagonal”

Out of all possible models the following gives the smallest error:

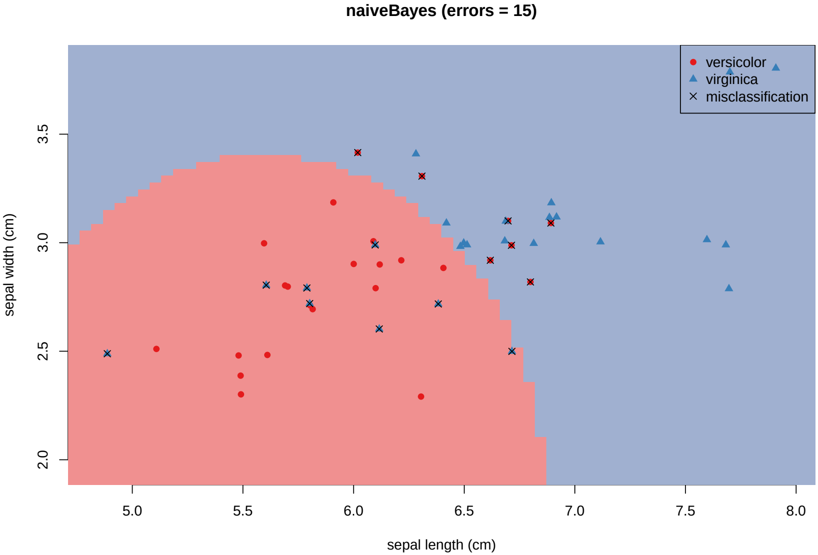


State-of-the-art classifiers and Iris data

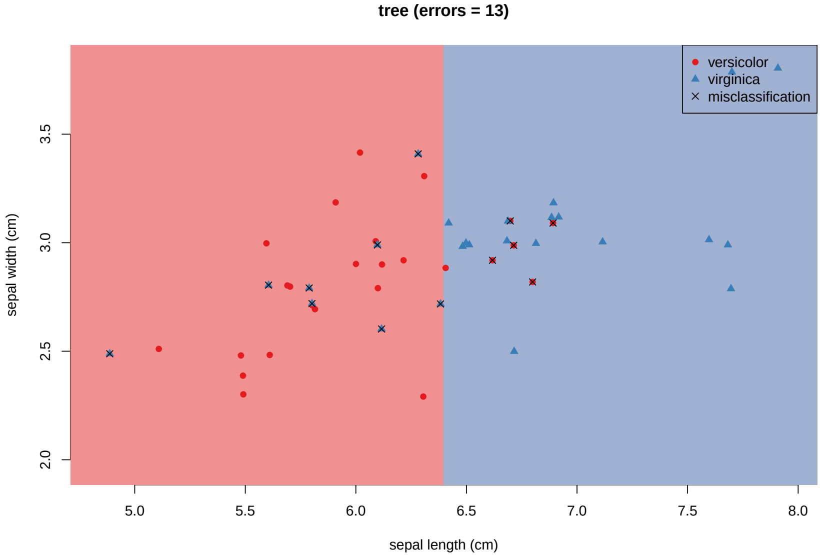
- Naive Bayes classifier (naiveBayes)
- Classification tree (tree)
- Support vector machine with radial basis kernel (SVM)
- 1-nearest neighbour classifier (1NN)

These will be discussed later in the course!

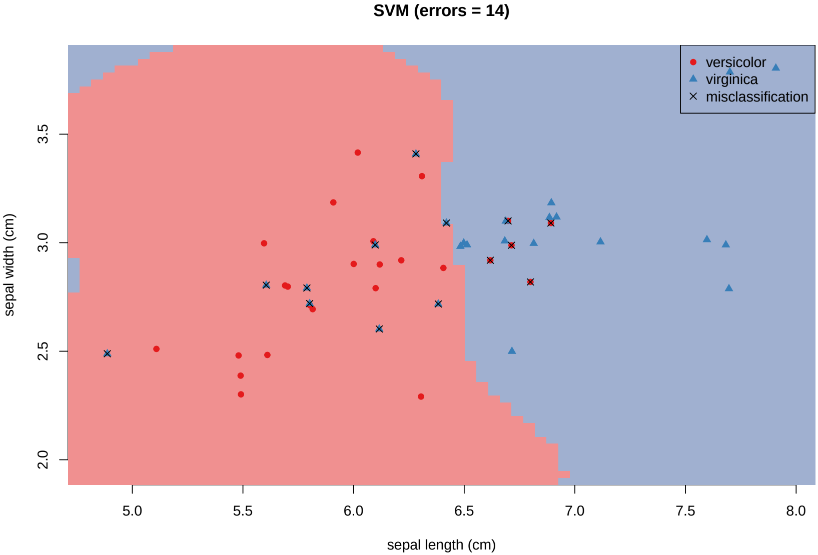
Naive Bayes classifier (naiveBayes)



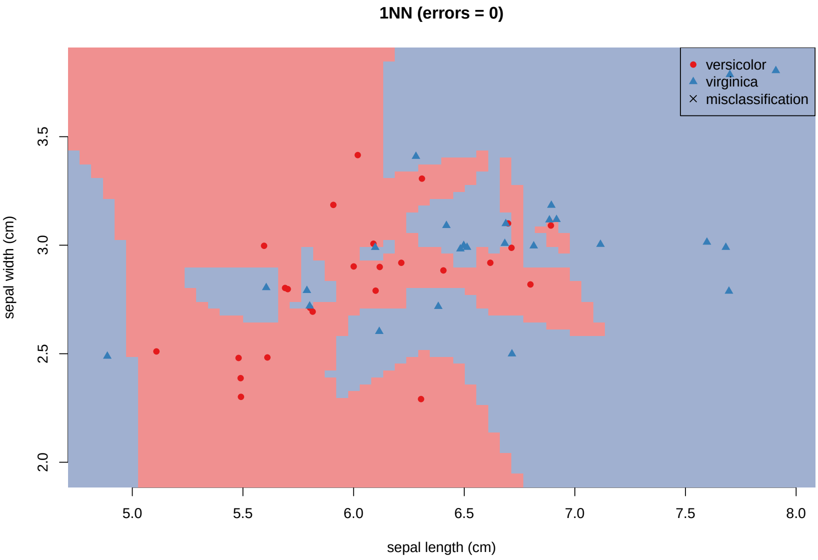
Classification tree (tree)



Support vector machine with radial basis kernel (SVM)



1-nearest neighbour classifier (1NN)



Summary of performance

classifier	errors
vertical	13
diagonal	13
naiveBayes	15
tree	13
SVM	14
1NN	0

Question: Which is the best algorithm / model?

Evaluating performance on a task (Sec. 2.2)

- When we apply machine learning techniques, we have available some *training data* which we use to come up with a good model
- However, often we care more about *generalisation*: performance on future *unseen* data, *not* the training set
- Lets take 50 flowers that are not in training set (*test set*) and evaluate the performance of the classifiers!

Evaluating performance on a task (Sec. 2.2)

- Performance on training data can be much better than performance on future *test* data:
  - choosing a model that performs best on the training data can favor a model that was good by chance (it got "lucky")
  - the more models there are to choose from, and the less the training data, the worse this gets
  - performance on unseen test data can be much worse
- To get an unbiased estimate of performance on unseen data, one can withhold part of the training data and use it as *test set* – but only after choosing a model; why?

Multiple testing (multiple comparison)

Example:

- A number of investment advisors who are predicting whether the stock market will rise or fall in the next day.
  - No advisor is actually any better than a coin-flip!
- We have the record of 50 advisors for 10 consecutive days.
- Probability that a specific advisor will be correct at least 8 days out of 10:

$$\frac{\binom{10}{8} + \binom{10}{9} + \binom{10}{10}}{2^{10}} \approx 0.0547$$

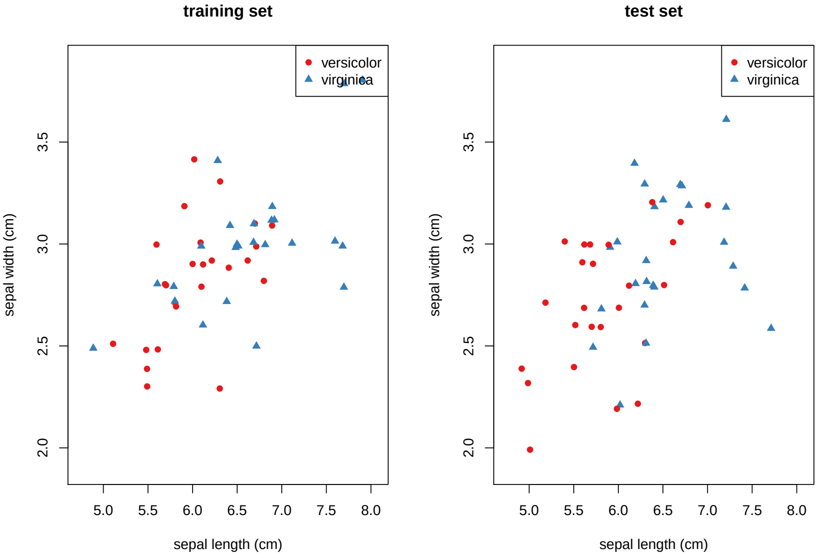
- Probability that at least one of them gets at least 8 correct guesses:

$$1 - (1 - 0.0547)^{50} \approx 0.9399$$

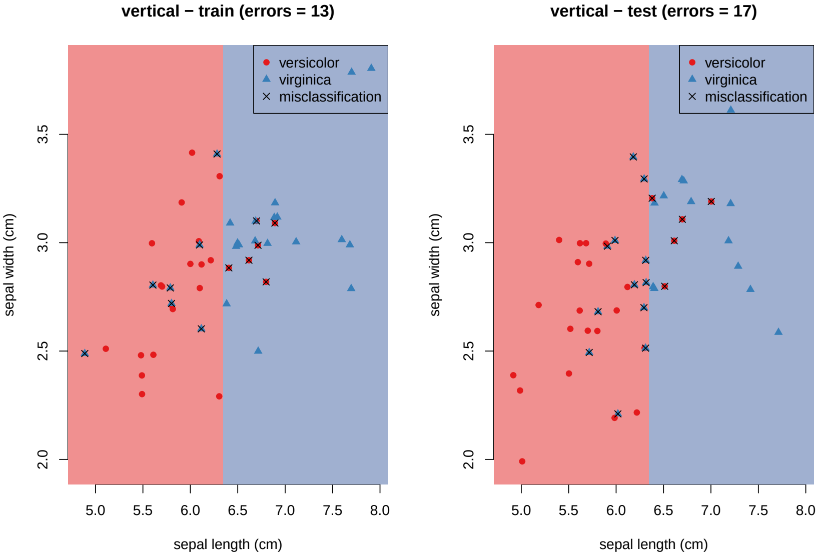
Multiple testing (multiple comparison)

- The moral of the story: If you are comparing a number of random predictors, it is likely that *some* will have very good empirical performance *even if they a re all quite random.*
- While the training set performance is related to generalization, one should not expect similar test set performance unless one tests the model on a fresh data set *after selection*
- **The bigger the set of models to choose from, the worse it gets.**
- This issue is fundamental in machine learning:
  1. A tempting approach is to evaluate the performance of each model on the training set and to choose the best (a.k.a. empiricial risk minimization).
  2. However, this is usually not the best approach. Can you think of a better one? (It's *very* tough.)

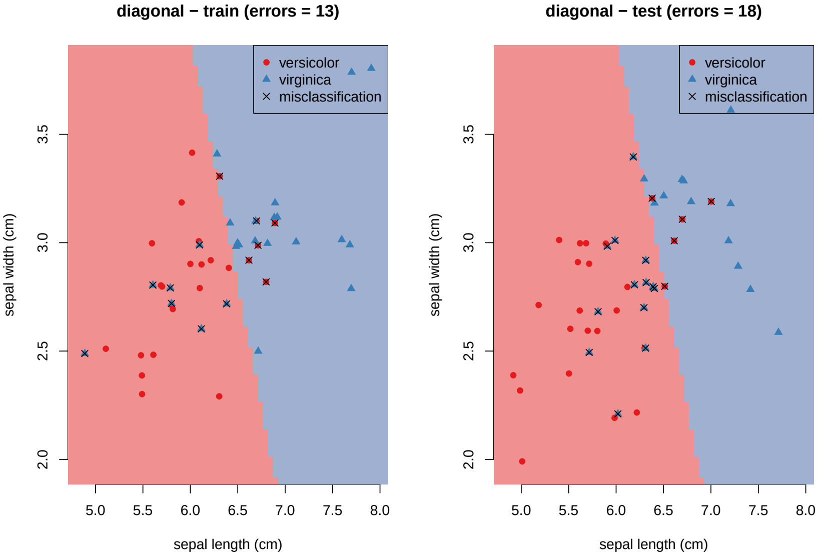
Training set and test set



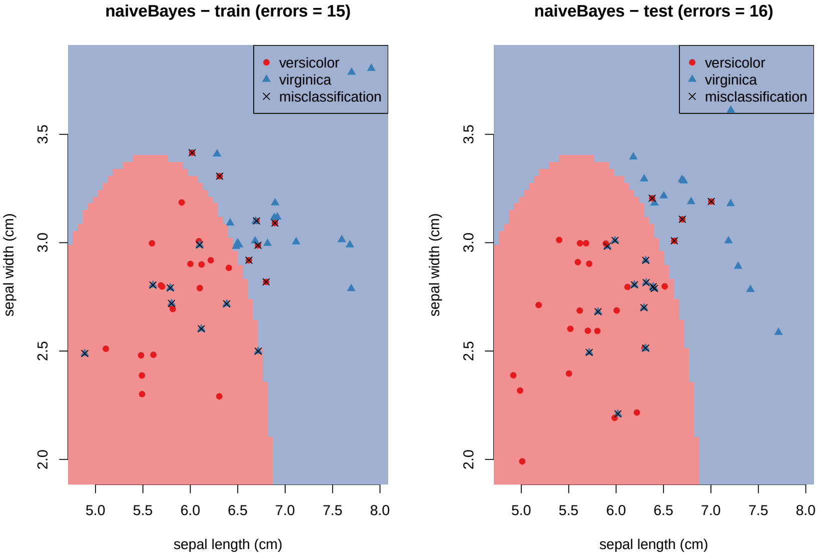
vertical on test set



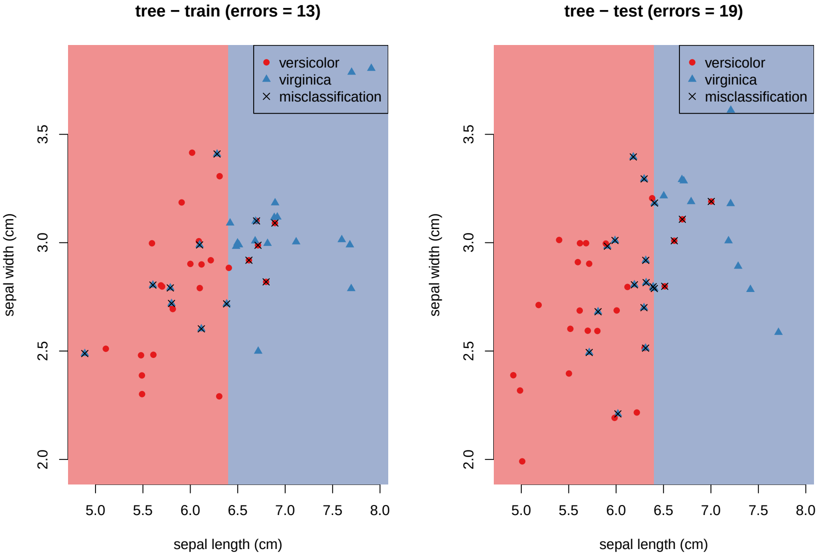
diagonal on test set



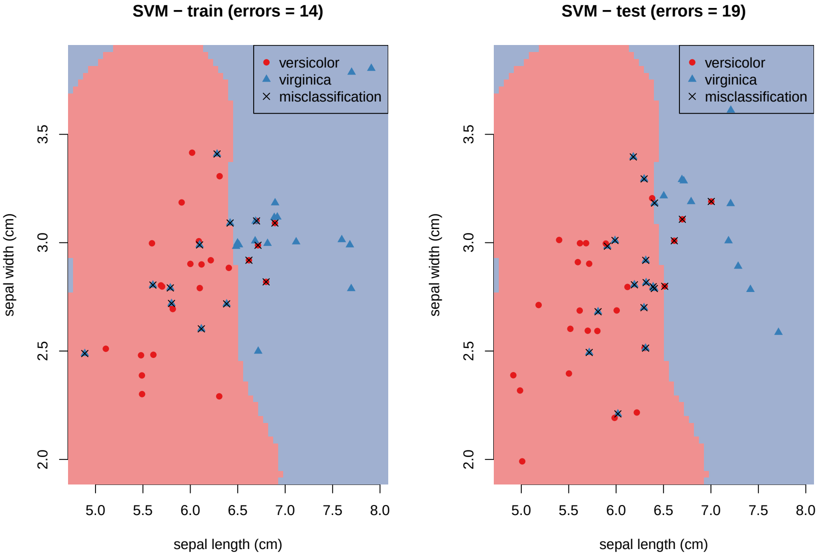
naiveBayes on test set



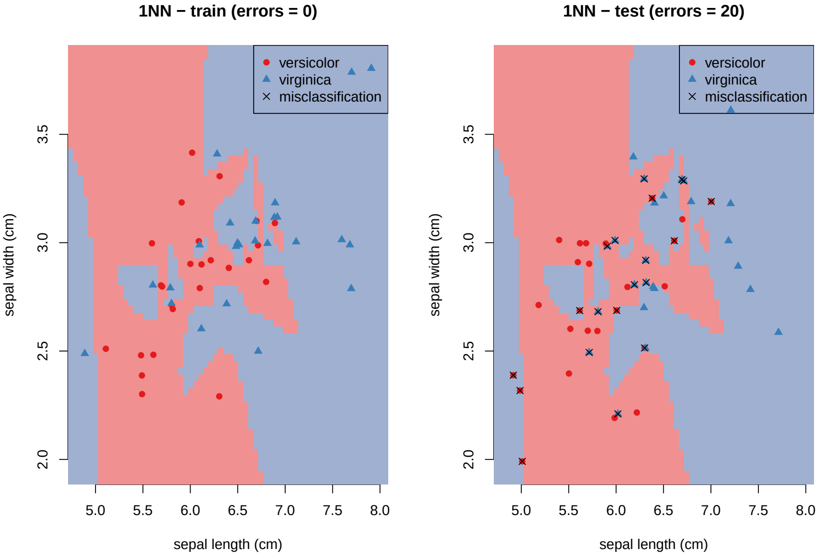
tree on test set



SVM on test set



1NN on test set



Summary of performance on test set

classifier	errors on training set	errors on test set
vertical	13	17
diagonal	13	18
naiveBayes	15	16
tree	13	19
SVM	14	19
1NN	0	20

- *Underfitting*: model too simple (poor generalization)
- *Overfitting*: model too complex (poor generalization)
- *Inductive bias*: choice of learning algorithm incorporates assumptions (generalization is impossible without any assumptions!)

Imagenet

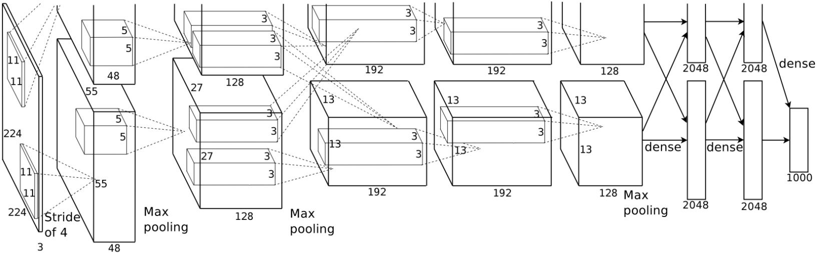


Figure 1: Imagenet

Krizhevsky et al. (2012) ImageNet Classification with Deep Convolutional Neural Networks. In Proc NIPS 2012.

Estimating generalization performance

- To get an unbiased estimate of the generalization performance, test it on data that *has not been used in any way to guide the search for the best model*. (The test data should be **“locked in a vault”**. No peeking!)
- But again, if you are testing several models, beware that the best/worst results may not be good estimates of the generalization error of those models. (Don’t fall into that trap, again!)

Where we are, and what's happening next

- We defined some basic components of a machine learning scenario, such as task, data, model, algorithm.
- We defined different types of machine learning problems: supervised, unsupervised, reinforcement learning.
- We took some first steps on our way to understand generalisation performance, **the most important topic of this course**!