Lecture#2

Data and Learning

What is Machine Learning?

”… the construction and study of systems that can learn from data.”

- A system that can:
  - Take known data as input
  - Learn from the known data
  - Draw conclusions from unseen data

Machine Learning and Data Mining

- When talking about machine learning, you often come across the term Data Mining
- They are sometimes taken for meaning the same thing
- Data Mining is however a broader term
- It is about finding meaning in data
- It can be done with machine learning, but also with for example statistics and visualization
- Machine learning is about algorithms that can learn from data

Data and Data Representation

Example/Instance

- Data consists of inputs and outputs
- Each set of inputs and outputs is an independent example (instance) of the data
- In some cases the output is known
- Data can also be continuous streams, but that is out of scope of this course
- The inputs (called features or attributes) and outputs consists of one or more variables

Features/Attributes

- Features (attributes) are variables describing an example of the data
- The input typically consists of several features
- The output is often one or a few variables
- The variables can be of different types:
  - Numbers (integers or floats)
  - Nominal/categorical – a finite set of discrete categories

Common datasets

Weather dataset

- Learns if we want to go out and play or not based on weather conditions

- Four attributes, two nominal and two numerical
- Two categories
- 14 examples

Weather dataset

Outlook sunny, overcast, rainy	Temperature numeric	Humidity numeric	Windy true, false	Play yes,no
sunny	85	85	false	no
sunny	90	90	true	no
overcast	83	86	false	yes
rainy	70	96	false	yes
rainy	68	80	false	yes
rainy	65	70	true	no
overcast	74	65	true	yes
sunny	72	95	false	no
sunny	69	70	false	yes
rainy	75	80	false	yes
sunny	75	70	true	yes
overcast	72	90	true	yes
overcast	81	75	false	yes
rainy	71	91	true	no

Iris dataset

- Learns to distinct between three subspecies of the iris flower based on measurements on the flowers

- Four numerical attributes
- Three categories
- 150 examples

Iris dataset

Sepal Length	Sepal Width	Petal Length	Petal Width	Species
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.0	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
7.0	3.2	4.7	1.4	Iris-versicolor
6.4	3.2	4.5	1.5	Iris-versicolor
6.9	3.1	4.9	1.5	Iris-versicolor
6.3	3.3	6.0	2.5	Iris-virginica
5.8	2.7	5.1	1.9	Iris-virginica
7.1	3.0	5.9	2.1	Iris-virginica

Wikipedia dataset

- All words (tags and code removed) from 70 articles at Wikipedia
- 35 articles about Programming, 35 about Games (two categories)
- Learns how to distinct between articles about programming and about games
- In text classification datasets, we usually generate a list of all words from an article/blog post/tweet
- This is called a bag-of-words

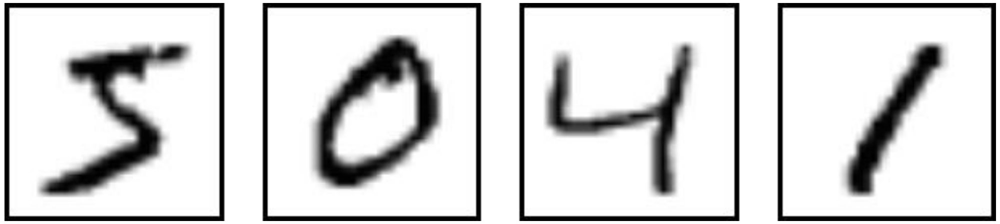
Wikipedia dataset

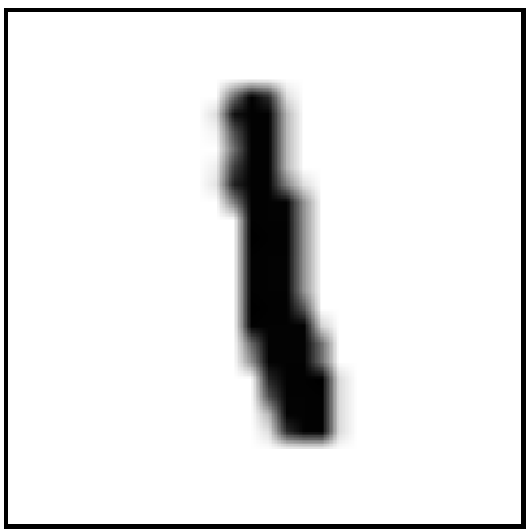
Bag-of-words	Category
perl from wikipedia free encyclopedia jump navigation search this article about programming language …	Programming
list best-selling video game franchises from wikipedia free encyclopedia jump navigation search this …	Games
video game development from wikipedia free encyclopedia jump navigation search game development …	Games
programming language from wikipedia free encyclopedia this latest accepted revision reviewed on …	Programming

MNIST dataset

- MNIST is a dataset containing images of handwritten digits
- It has a training set of 60000 examples and a test set of 10000 examples
- There are, of course, 10 categories (0, 1, … , 9)
- Each image is 28x28 pixels

MNIST dataset





$$\simeq \begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&.6&.8&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.5&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.7&0&0&0&0&0\\
0&0&0&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&0&0&.9&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&.3&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}$$

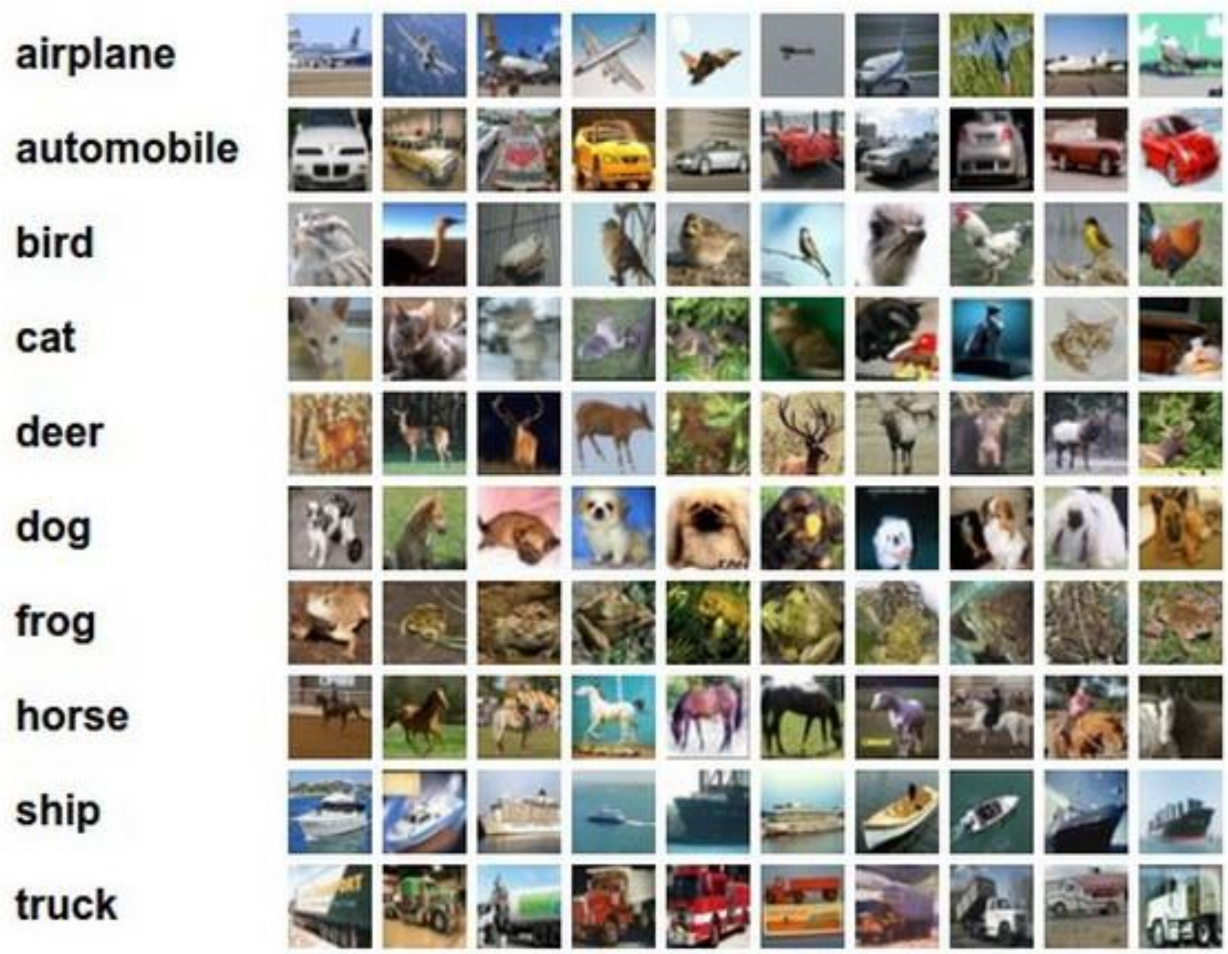
MNIST dataset

- Each image can be seen as a 28x28 matrix of float values
- Each value represents the darkness of a pixel:
  - 0.0: white
  - 1.0: black
- To use it we flatten the array to a 28x28 = 784 input vector:
  - [0.0, 0.0, 0.0, 0.0, 0.1, 0.1, 0.15, … , 0.0, **1**]
- MNIST is an image classification/recognition problem

CIFAR-10 dataset

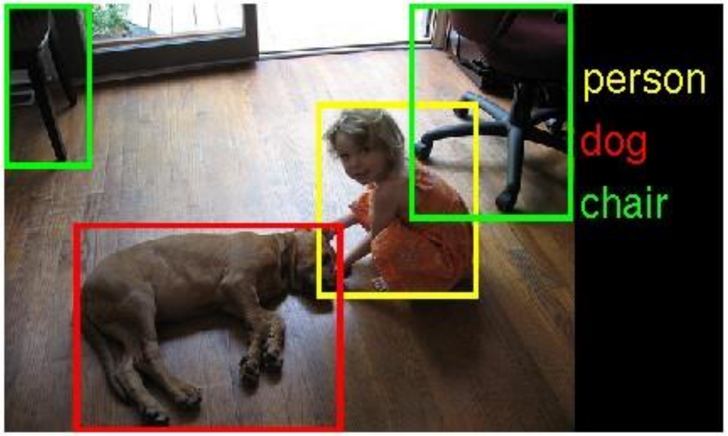
- The CIFAR-10 dataset is a much more complex image classification problem than MNIST
- It consists of 60000 images (50000 for training, 10000 for testing) of size 32x32 pixels
- It has 10 categories:
  - airplane, automobile, bird, …
- The input vector must be flattened to 32x32 pixels times 3 color channels (RGB):
  - 32x32x3 = 3072 input values

CIFAR-10 dataset



ImageNet challenge

- The ImageNet challenge is an annual contest for image classification and localization tasks
- The training dataset consists of 1.2 million images and 1000 possible categories
- The validation set for the challenge is a random subset of 50000 images
- Images can differ in size, but in average the resolution is 482x415 pixels



Types of learning problems

Types of learning problems

- In ML, data is feed to an algorithm which it can learn from
- Example: learn how to distinct between spam and no-spam emails
- Machine learning is divided into three broad categories:

Supervised learning

- Algorithms are presented with example inputs and known outputs:

Input 1	Input 2	Output
1	3	4
2	1	3
3	5	8

- The learning task is to map the inputs to the output
- The output can consist of categories (classification) or a continuous number (regression)

Unsupervised learning

- In contrast to supervised learning, no known output is given
- The algorithms are left on their own to find patterns or structures in the input data
- An example is to group news articles discussing similar topics together
- We will not cover unsupervised learning in this course

Reinforcement learning

- In reinforcement learning, systems learn from trial and error
- The system executes an action in its environment, and is given feedback on how well it worked out
- If the action was a success, a positive reward is given
- If the action was a failure, a negative reward (punishment) is given
- Over time, the system learns what actions are successful in its environment
- An example is creating a bot that can learn how to play a game
- We will not cover reinforcement learning in this course

Machine Learning

- Many machine learning algorithms are heavily based on mathematics or statistics
- We will try to minimize the mathematical background of algorithms
- And focus on applying algorithms on different tasks

Training and Validation

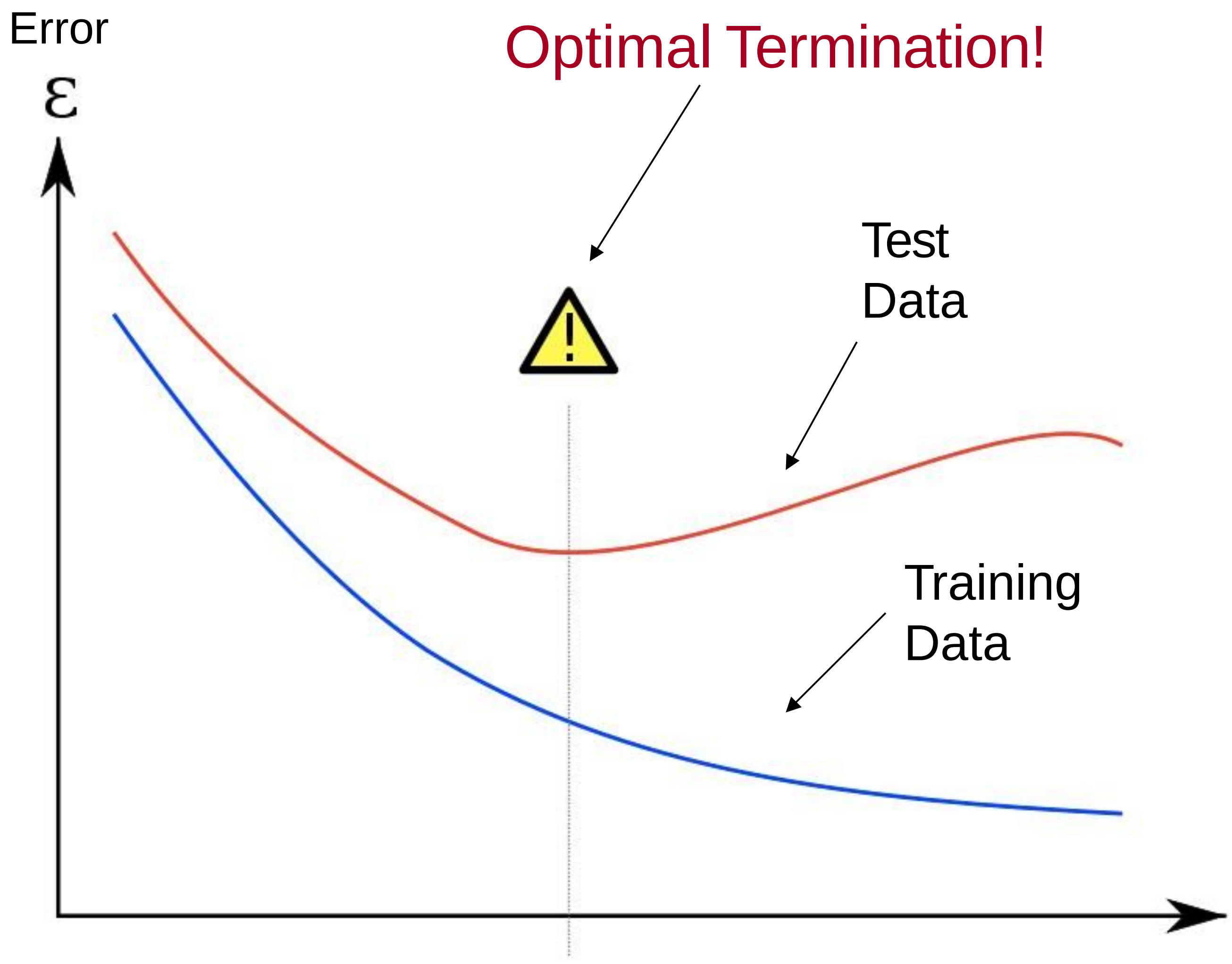
Training and Validation

- The machine learning algorithm is trained using a dataset
- The dataset consists of a number of *examples* (instances) with known output
- The trained algorithm is called a *model*
- The model is used to classify new instances
- We can check how good the model is by calculating the *accuracy*
- Accuracy means the percentage correctly classified instances in the test set

Training set and Test set

- If we use the same dataset for both training and testing, the model must loose some of its generalization abilities
- We learn the dataset too well, which can lead to worse performance on unseen examples
- This is called overfitting:

Overfitting



http://en.wikipedia.org/wiki/Overfitting_(machine_learning)

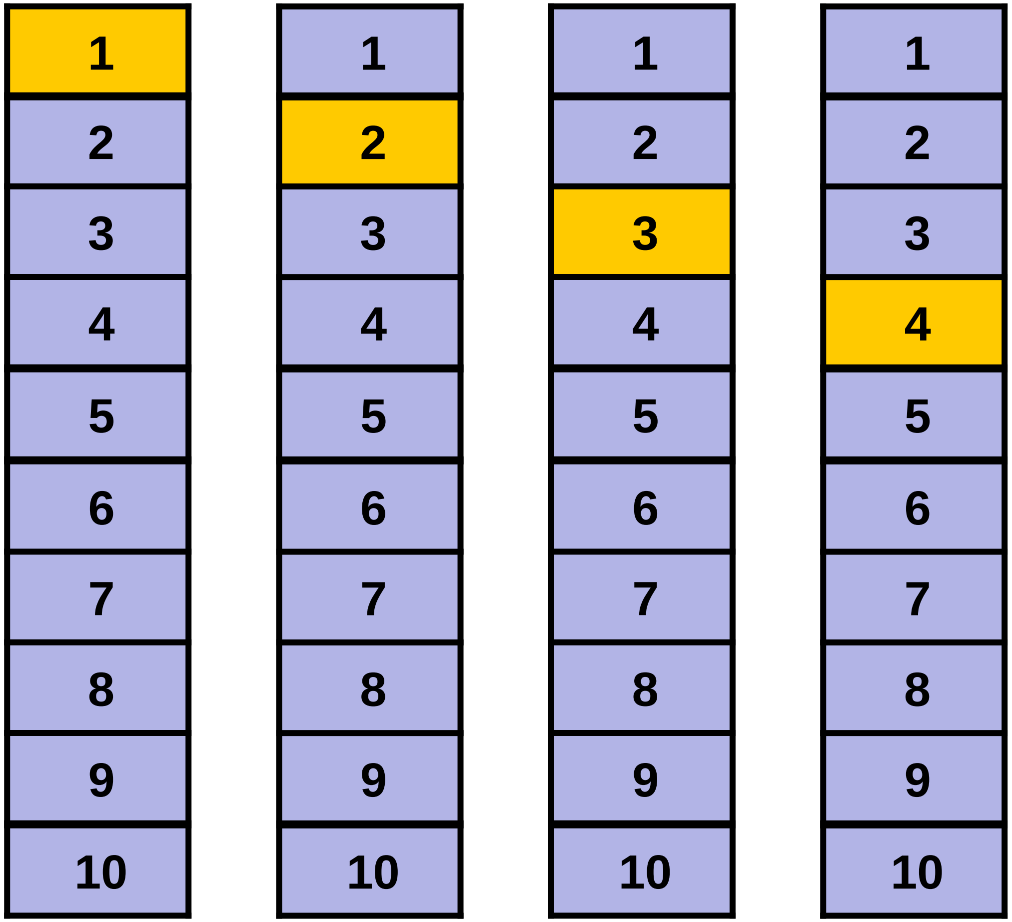
Separate datasets

- One way of improving the generalization compatibilities and reduce overfitting is to use two datasets
- The first set is used to train the model
- It typically contains around 90% of the examples
- The second set is used to test the model performance
- It contains around 10% of the examples
- We select the model candidate with the highest performance on the test dataset

Cross-validation

- In cross-validation, the dataset is divided into a number of buckets of equal size (10 is the most common)
- The system is trained on 9 buckets, and tested on the last bucket
- In the next iteration, another bucket is used for testing and the rest for training

Cross-validation



10-fold Cross Validation =
divide data into 10 parts

9 parts are used for training,
1 part for validation

Iterate until all parts have
been used for validation

CV and Test set

- Often we train the system on the training dataset using cross-validation
- The system performance is then validated using a test dataset

Good or bad result

- How ”good” the accuracy is depends on how many possible categories we have
- An accuracy of 50-60% on a binary classification problem (2 categories) is not much better than random chance!
- The same accuracy can however be rather good if we have 10 possible categories!

ZeroR

- We can use the ZeroR classifier as baseline when comparing the results for different classifiers
- ZeroR simply classifies all examples as the most frequent category in the dataset
- ZeroR has an accuracy of 33.3% on the iris dataset, since we have an equal amount of examples from the three categories

Performance Metrics

Accuracy

- Accuracy is the most common performance metric for machine learning algorithms
- It means the percentage correctly classified instances
- If we have 150 examples in the test dataset, and 138 of them is correctly classified
- … we calculate accuracy as 138/150 = 92%

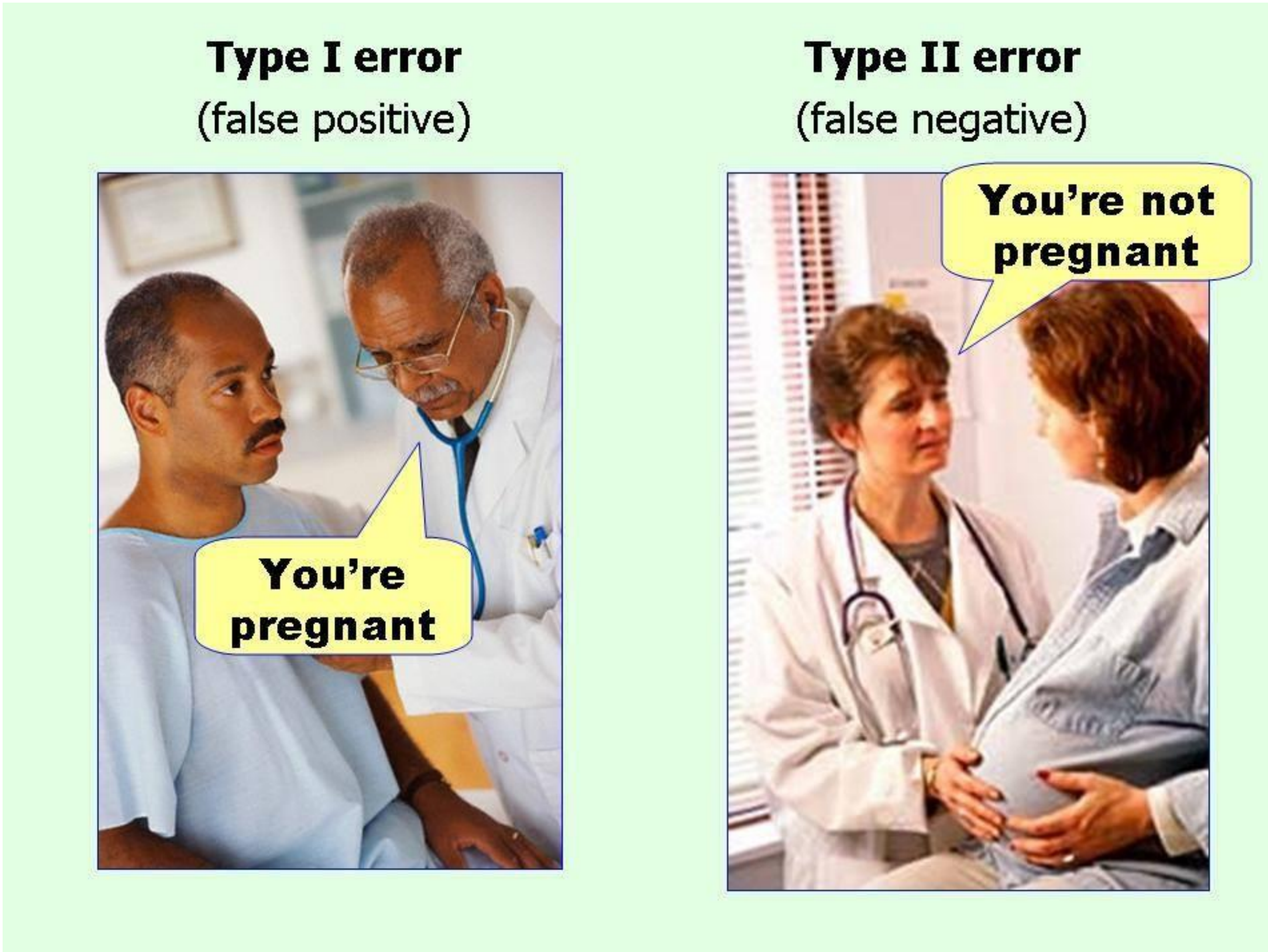
Is this a good metric?

- Accuracy gives an estimate of how well the model performs on a test dataset
- It is simple to calculate and it is easy to compare performance with other systems that uses the same dataset
- It is however often overly optimistic
- There are other things that are more or less important to know depending on the task:

True or false classifications

- TP = true positives
  - we classify a correct example as correct
- FP = false positives (type 1 error)
  - we classify an incorrect example as correct
- TN = true negatives
  - we classify an incorrect example as incorrect
- FN = false negatives (type 2 error)
  - we classify a correct example as incorrect

True or false classifications



True or false classifications

- Depending on the task, knowing if an error is of type 1 or 2 can be important
- In for example earthquake detection systems we really don’t want to alert the alarm if there are no earthquake, spreading fear among people (type 1 error)
- It is better that we miss a sign, and possibly detects the earthquake later (type 2 error)

True or false classifications

- In an email spam detection system, we want to avoid having legitimate emails ending up in the spam folder (type 1 error)
- It doesn’t matter that much if some spam end up in the Inbox (type 2 error)

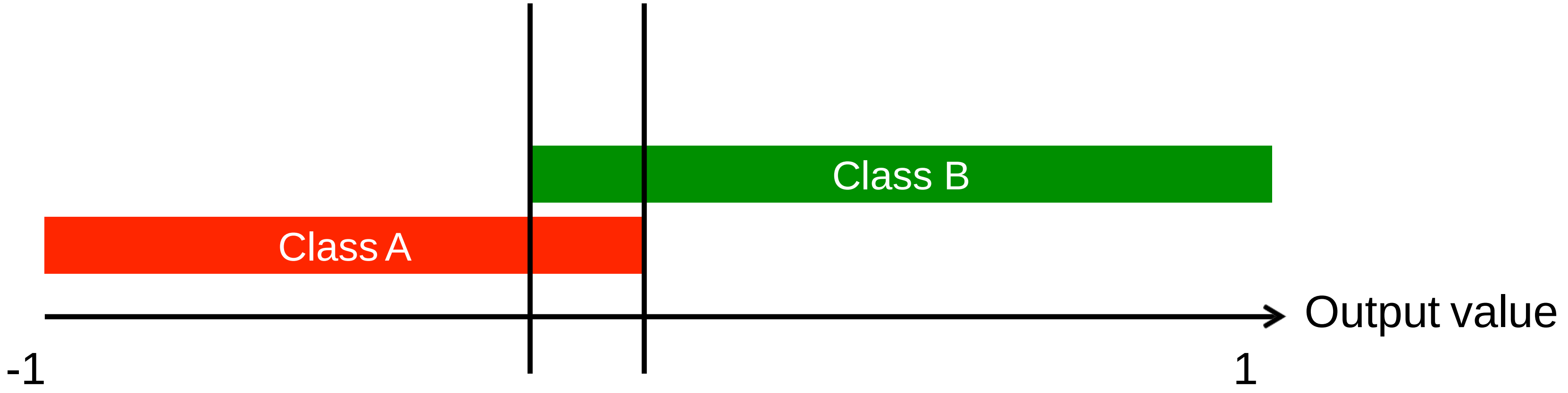
ROC Analysis

- TPR = TP / (TP + FN) Sensitivity
- FPR = FP / (FP + TN) Specificity

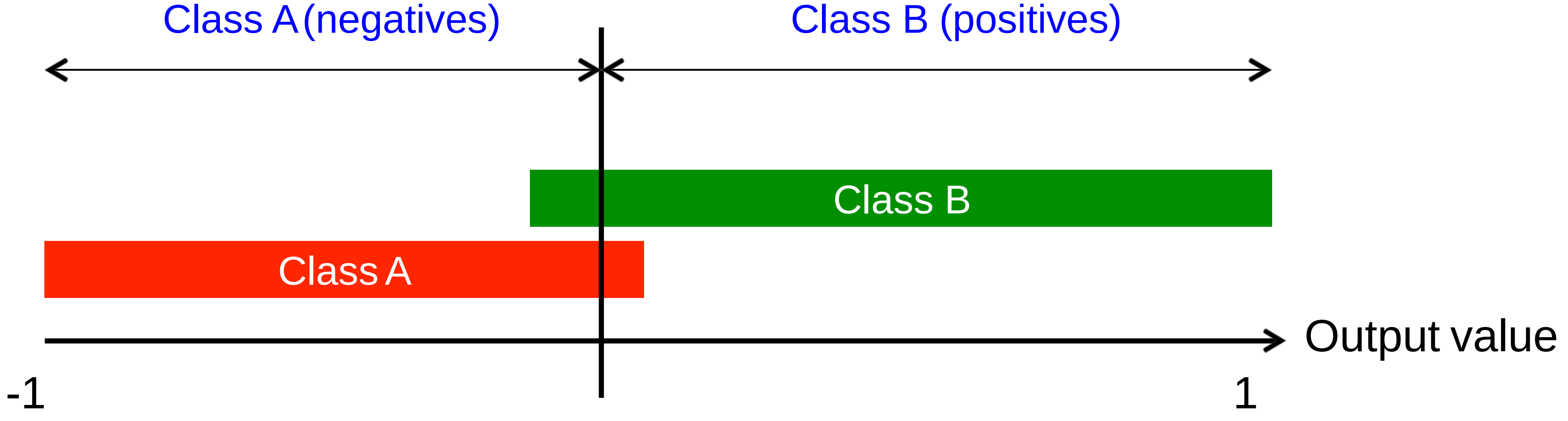
- Plot TPR vs. FPR as the discrimination threshold is varied
- This is where we place the line that divides two classes
- In many cases, classes overlap

Discrimination Threshold

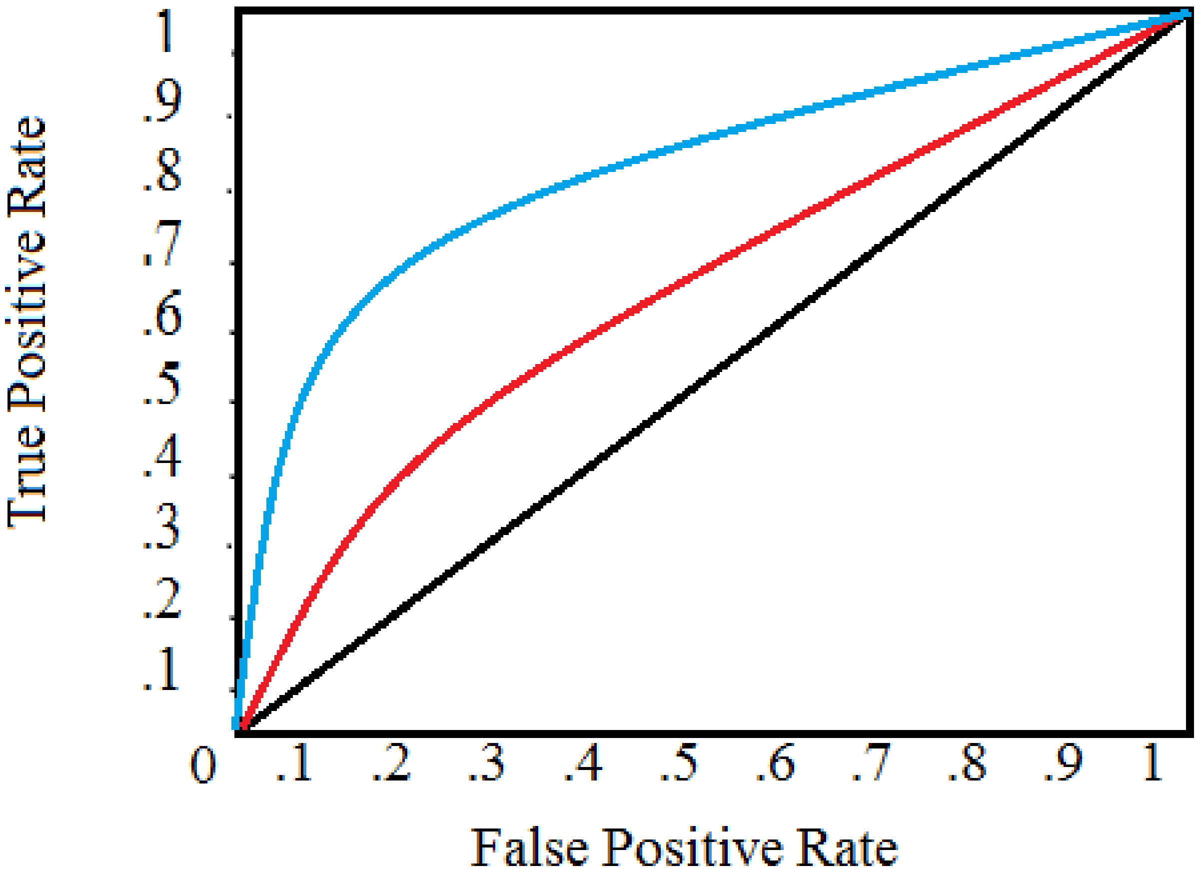
Depending on where we put the discrimination threshold, the TPR and FPR will vary.



Discrimination Threshold



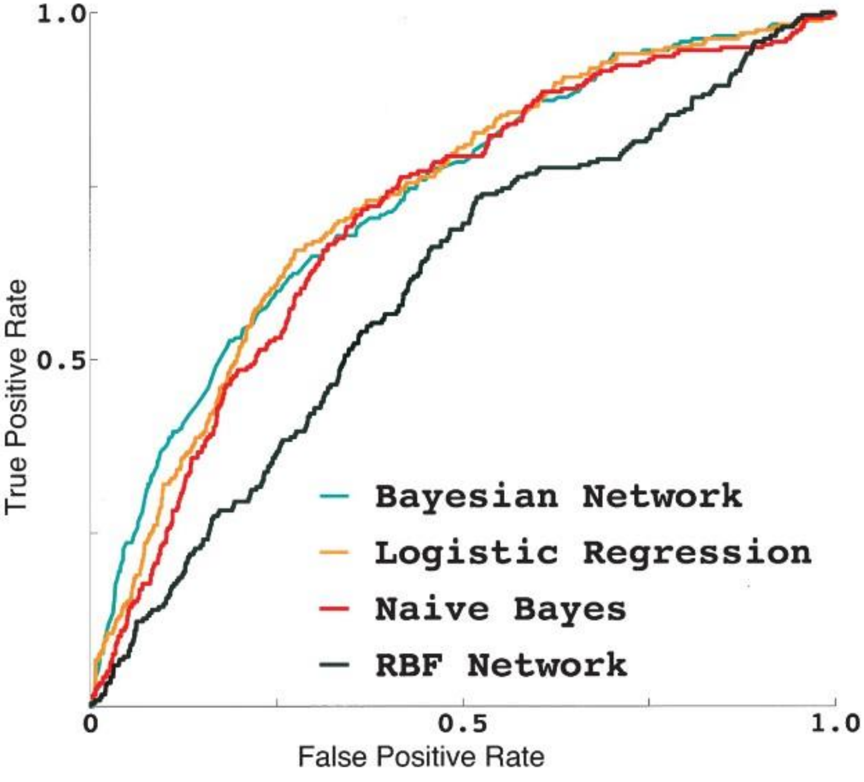
ROC curve



True Positive Rate
1
.9
.8
.7
.6
.5
.4
.3
.2
.1
0
.1
.2
.3
.4
.5
.6
.7
.8
.9
1
False Positive Rate

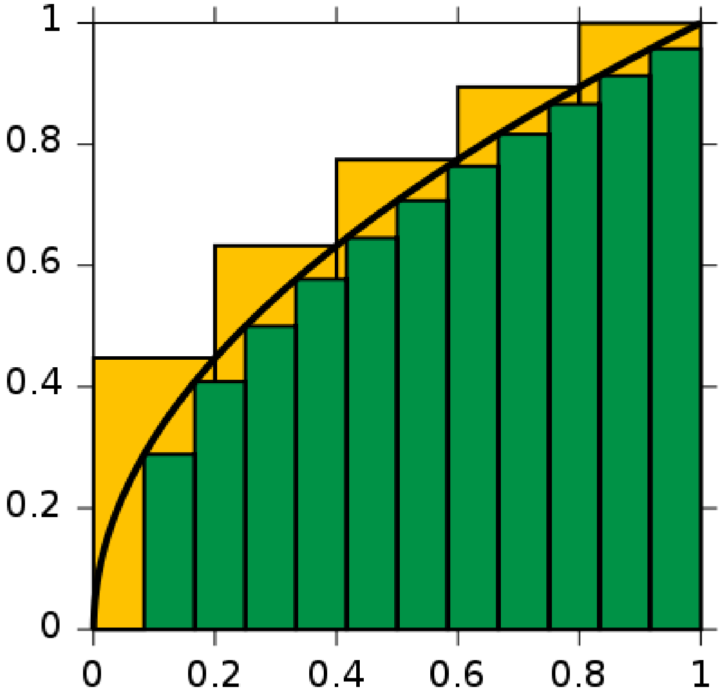
ROC curve

- The diagonal represents a pure guess
- The closer the curve is to the upper left corner, the more accurate it is



ROC area

- A single measure instead of a curve
- Calculated as the area (integral) under the ROC curve



F-score

- Another single measure that takes FN and FP in consideration:

$$F = \frac{2 * TP}{2 * TP + FP + FN}$$

Confusion Matrix

- A confusion matrix plots the correct and incorrect classifications for each category:

Confusion Matrix

A	B		
48	2	A	= Category 1
4	46	B	= Category 2

Example: Iris dataset

Correctly classified examples	142	94.67%
Incorrectly classified examples	8	5.33%
TP rate	0.947	
FP rate	0.027	
F-score	0.947	
ROC area	0.996	

Confusion Matrix

A	B	C		
50	0	0	A	= Iris-setosa
0	46	4	B	= Iris-versicolor
0	4	46	C	= Iris-virginica

Other important characteristics

- There are other things we need to take into consideration when selecting an algorithm for a problem:
  - Training and classification time
  - Space consumption of the trained model
  - Explainability – can we understand what the model has learned?
  - Possibility of online learning – can we continue training a model with new examples without having access to all data?

Tools and Libraries

Tools and Libraries

- There are a wide range of different tools and libraries for machine learning
- Some are free, some costs a lot of money
- Some can be called from code using an API, others cannot
- In this course we will take a look at four tools/libraries:
  - Weka
  - R
  - TensorFlow
  - Scikit

Weka

WEKA
The University
of Waikato

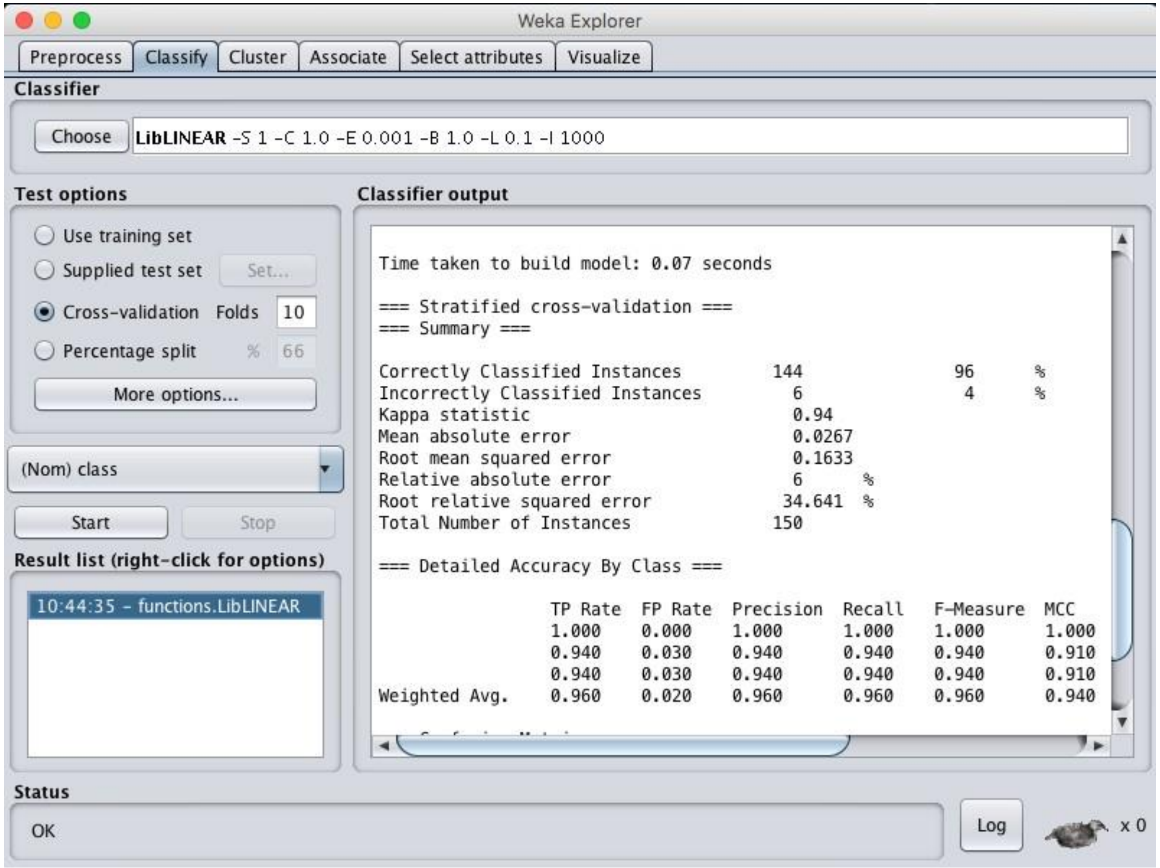
Weka 3: Data Mining Software in Java

Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes.

https://www.cs.waikato.ac.nz/ml/weka/

Weka

- Weka is both a stand-alone application with a GUI, and a Java API

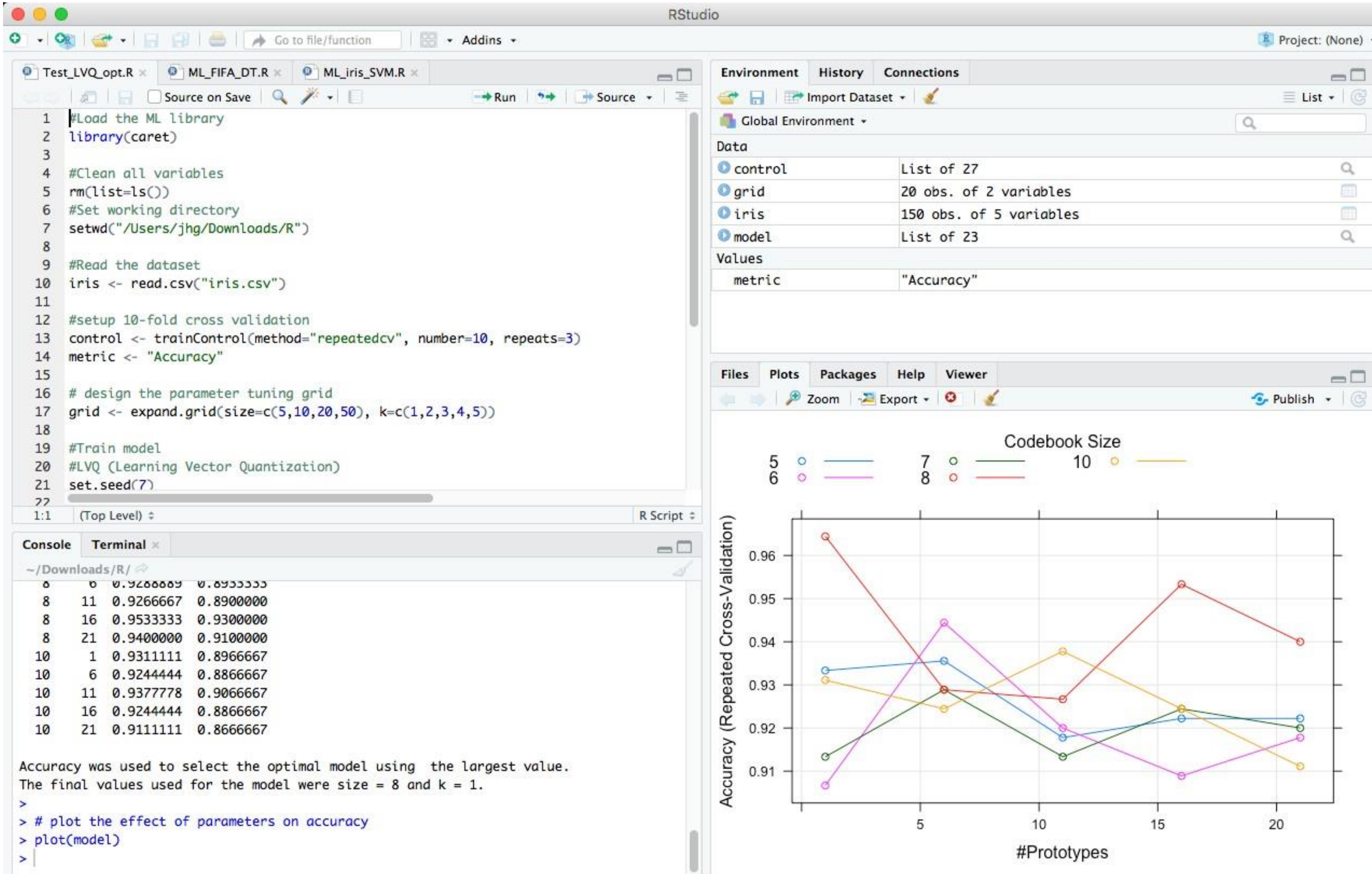


R

- R is a mathematical and statistical tool that contains several machine learning algorithms
- R is a free alternative to Matlab
- It has no API, but is useful for experimenting on datasets since it has many features for visualizing data and classifiers
- R has a somewhat unconventional language which can take some time to learn

https://www.r-project.org
https://www.rstudio.com

R


RStudio
Test_LVQ_opt.R
ML_FIFA_DT.R
ML_iris_SVM.R
#Load the ML library
library(caret)
#Clean all variables
rm(list=ls())
#Set working directory
setwd("/Users/jhg/Downloads/R")
#Read the dataset
iris <- read.csv("iris.csv")
#setup 10-fold cross validation
control <- trainControl(method="repeatedcv", number=10, repeats=3)
metric <- "Accuracy"
# design the parameter tuning grid
grid <- expand.grid(size=c(5,10,20,50), k=c(1,2,3,4,5))
#Train model
#LVQ (Learning Vector Quantization)
set.seed(7)
Environment
History
Connections
Global Environment
Data
control List of 27
grid 20 obs. of 2 variables
iris 150 obs. of 5 variables
model List of 23
Values
metric "Accuracy"
Console
Terminal
8 11 0.9266667 0.8900000
8 16 0.9533333 0.9300000
8 21 0.9400000 0.9100000
10 1 0.9311111 0.8966667
10 6 0.9244444 0.8866667
10 11 0.9377778 0.9066667
10 16 0.9244444 0.8866667
10 21 0.9111111 0.8666667
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were size = 8 and k = 1.
> # plot the effect of parameters on accuracy
> plot(model)
Files
Plots
Packages
Help
Viewer
Codebook Size
5
6
7
8
10
Accuracy (Repeated Cross-Validation)
0.96
0.95
0.94
0.93
0.92
0.91
5
10
15
20
#Prototypes

TensorFlow

- Google's TensorFlow is a library for machine learning
- It is most well known for its Deep Learning implementations
- It can use GPUs and multiple CPUs to speed up training and testing
- There is also a version that runs on mobile devices
- The API is for Python, but there are also Java and C++ versions available

https://www.tensorflow.org

TensorFlow

```
"""A very simple MNIST classifier.
See extensive documentation at
https://www.tensorflow.org/get_started/mnist/beginners
"""
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import argparse
import sys

from tensorflow.examples.tutorials.mnist import input_data

import tensorflow as tf

FLAGS = None


def main(_):
  # Import data
  mnist = input_data.read_data_sets(FLAGS.data_dir, one_hot=True)

  # Create the model
  x = tf.placeholder(tf.float32, [None, 784])
  W = tf.Variable(tf.zeros([784, 10]))
  b = tf.Variable(tf.zeros([10]))
  y = tf.matmul(x, W) + b

  # Define loss and optimizer
  y_ = tf.placeholder(tf.float32, [None, 10])

  # The raw formulation of cross-entropy,
```

Scikit

- Scikit is a very popular machine learning library for Python
- It has many features for visualizing data and classifiers

http://scikit-learn.org/

Scikit

```
print(__doc__)

# Author: Gael Varoquaux <gael dot varoquaux at normalesup dot org>
# License: BSD 3 clause

# Standard scientific Python imports
import matplotlib.pyplot as plt

# Import datasets, classifiers and performance metrics
from sklearn import datasets, svm, metrics

# The digits dataset
digits = datasets.load_digits()

# The data that we are interested in is made of 8x8 images of digits, let's
# have a look at the first 4 images, stored in the `images` attribute of the
# dataset.  If we were working from image files, we could load them using
# matplotlib.pyplot.imread.  Note that each image must have the same size. For these
# images, we know which digit they represent: it is given in the 'target' of
# the dataset.
images_and_labels = list(zip(digits.images, digits.target))
for index, (image, label) in enumerate(images_and_labels[:4]):
    plt.subplot(2, 4, index + 1)
    plt.axis('off')
    plt.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
    plt.title('Training: %i' % label)

# To apply a classifier on this data, we need to flatten the image, to
# turn the data in a (samples, feature) matrix:
n_samples = len(digits.images)
data = digits.images.reshape((n_samples, -1))

# Create a classifier: a support vector classifier
classifier = svm.SVC(gamma=0.001)

# We learn the digits on the first half of the digits
classifier.fit(data[:n_samples // 2], digits.target[:n_samples // 2])

# Now predict the value of the digit on the second half:
expected = digits.target[n_samples // 2:]
predicted = classifier.predict(data[n_samples // 2:])

print("Classification report for classifier %s:\n%s\n"
      % (classifier, metrics.classification_report(expected, predicted)))
print("Confusion matrix:\n%s" % metrics.confusion_matrix(expected, predicted))

images_and_predictions = list(zip(digits.images[n_samples // 2:], predicted))
for index, (image, prediction) in enumerate(images_and_predictions[:4]):
```