Classification of biomedical texts

Introduction to Machine Learning for NLP

(Slides partially taken from Ulf Leser)

Mario Sänger (WBI, HU Berlin)

saengema@informatik.hu-berlin.de

We are (still) hiring!

Studentische Hilfskraft, Forschung und Lehre

In der Arbeitsgruppe "Wissensmanagement in der Bioinformatik" am Institut für Informatik der Humboldt-Universität zu Berlin ist ab 1.6.2020 eine studentische Hilfskraftstelle (40h/Monat, 2 Jahre) zu besetzen. Der/die Stelleninhaber*in unterstützt uns in der Lehre (als Korrektor*in bzw. Tutor*in) und arbeitet an Forschungsprojekten am Lehrstuhl mit. Diese beschäftigen sich mit angewandtem Maschinellem Lernen, biomedizinischen Text Minings, Informationsintegration, der skalierbaren verteilten Datenanalyse, und Bioinformatik für individualisierte Medizin.

Aufgaben

- Erstellung von Softwareprototypen
- Mitarbeit an Forschungsprojekten im Umfeld der biomedizinischen Datenanalyse
- Unterstützung in der Lehre

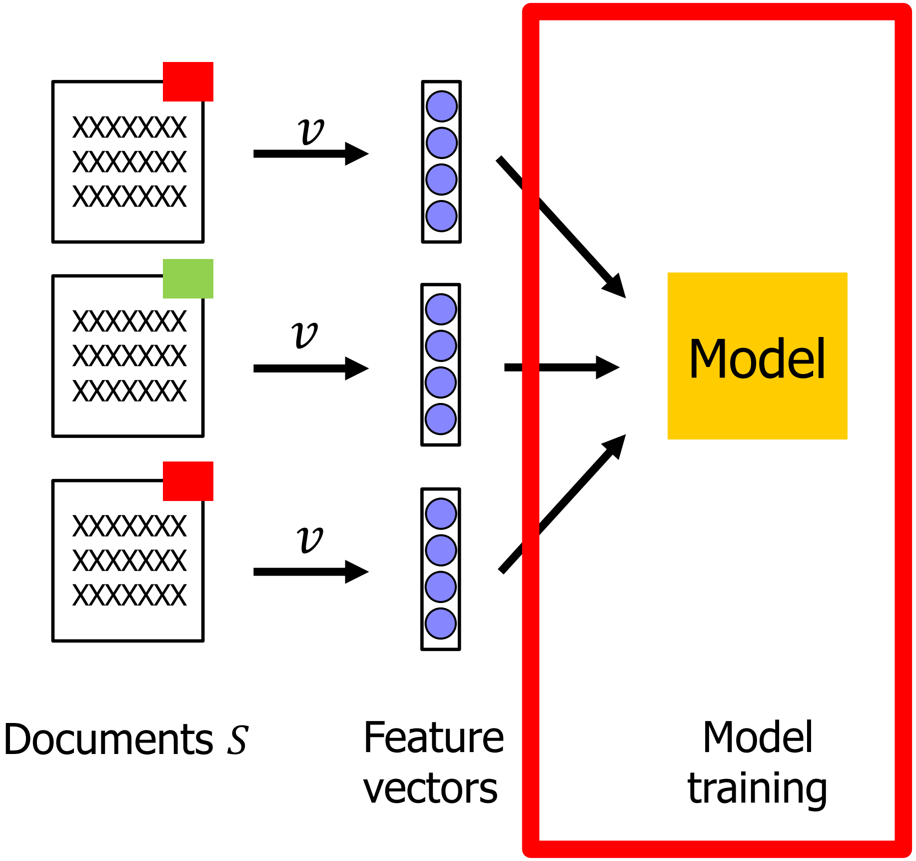
Voraussetzungen

- Studium der Informatik oder eines angrenzenden Fachs
- Vertiefte Erfahrung im Programmieren
- Erfahrung in der statistischen Datenanalyse und/oder der Bioinformatik
- Grosses Interesse an der angewandten Forschung
- Ein hohes Maß an Eigenmotivation und Kommunikationsfähigkeit / Teamfähigkeit
- Gutes Englisch

https://www.informatik.hu-berlin.de/de/forschung/gebiete/wbi/jobs/shk_haushalt_2004

Supervised text classification

- Given a set D of documents and a set of classes C. A classifier is a function $f: D \rightarrow C$



Problems

- Finding enough training data
- Finding the best pre-processing (tokenization, case, POS tag set …)
- Finding the best features
- Finding a good classifier (~ assigning as many docs as possible to their correct class)

Outline

- Machine Learning
  - Overview
  - Challenges and problems

- Classification methods
  - Nearest Neighbour
  - Linear classifiers

- Artificial neuronal networks
  - Motivation
  - Feed forward networks

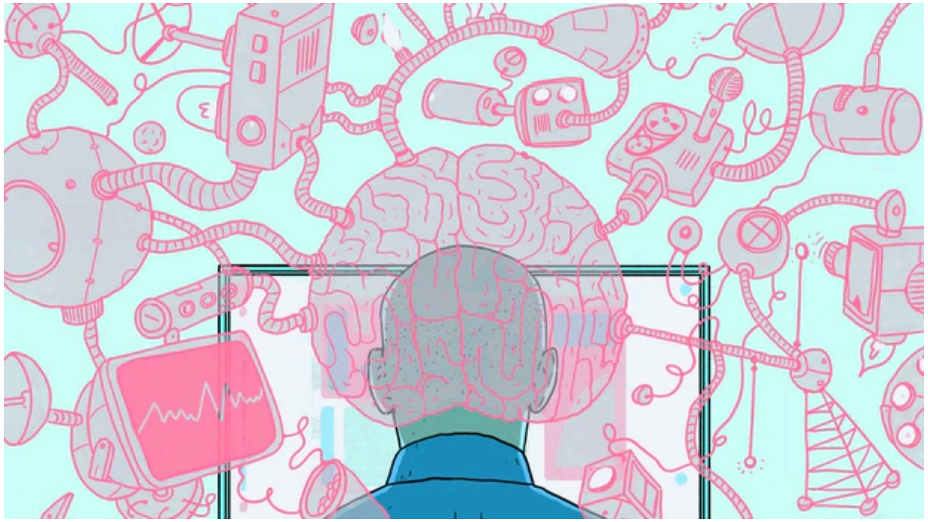
Machine Learning

Overview

Problems and Challenges

What is Machine Learning (ML)?

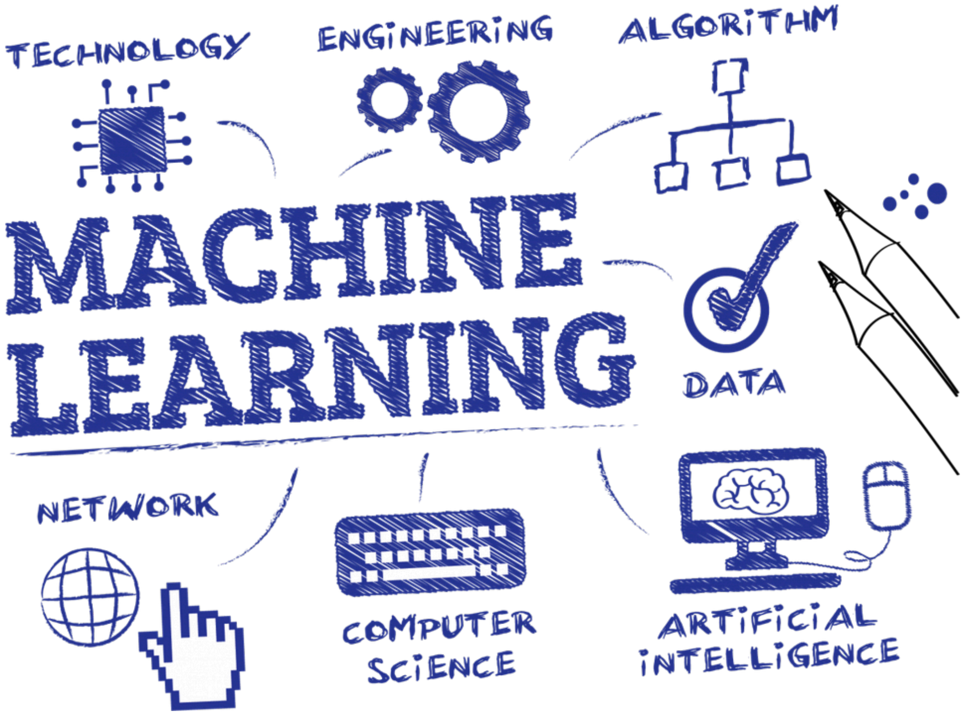
- Tom M. Mitchell (1997): "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E"



https://assets.t3n.sc/news/wp-content/uploads/2017/12/machine-learning-google-praesentation.jpeg?auto=format&fit=crop&h=348&ixlib=php-2.3.0&w=620

Machine Learning (ML)

- Perform a specific tasks without using explicit instructions
  - Build a mathematical model based on example data
  - ML models rely on patterns and inference methods in order to make predictions or decisions
- Integrates different disciplines



https://thumbor.forbes.com/thumbor/960x0/https%3A%2F%2Fblogs-images.forbes.com%2Fbernardmarr%2Ffiles%2F2018%2F03%2FAdobeStock_122936123-1200x891.jpg

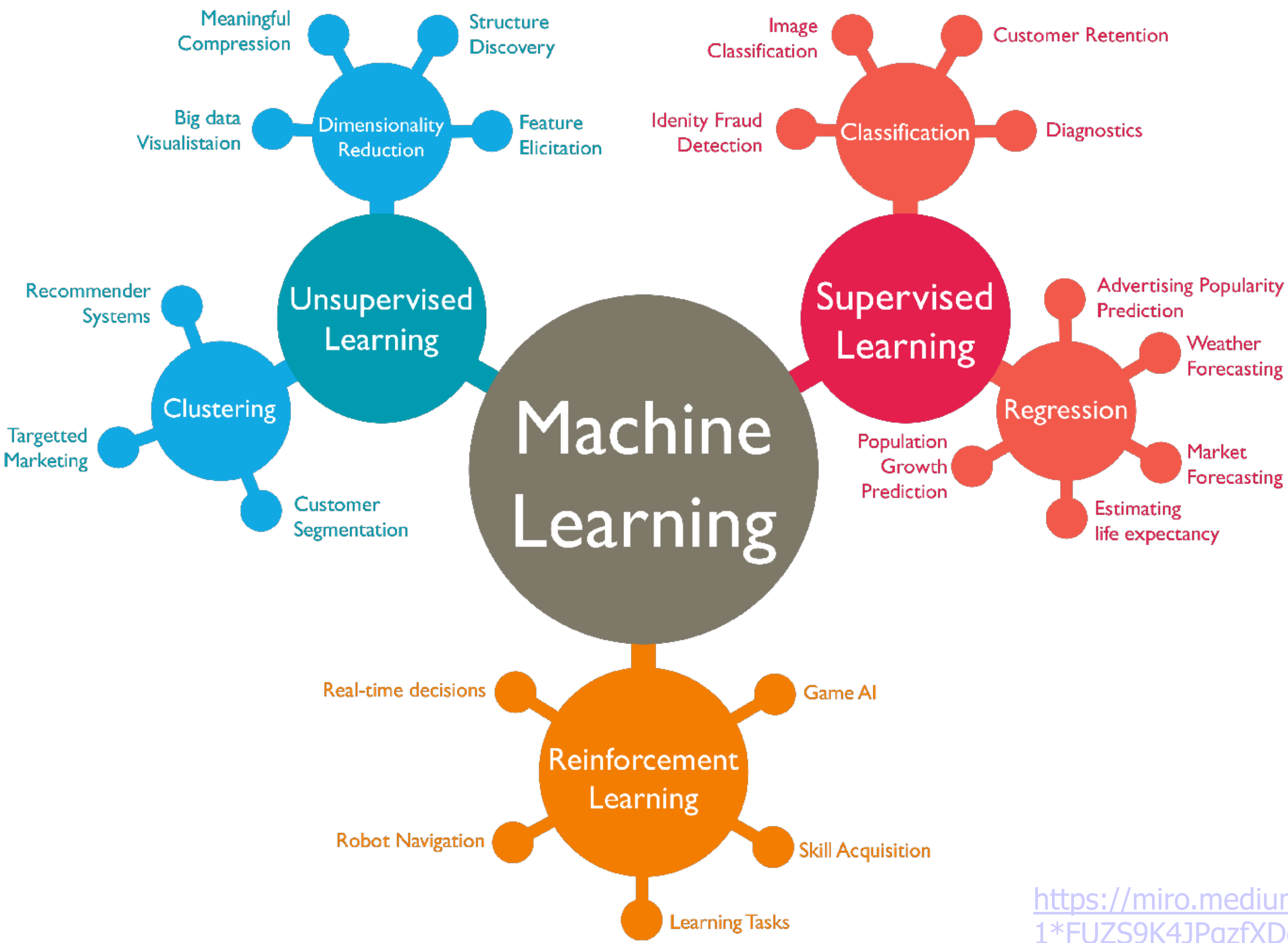
Types of Machine Learning (ML)

- Supervised learning: build a mathematical model based data sets that contain both the inputs and the desired outputs (labels)
  - Examples: classification and regression

- Unsupervised learning: find structures in unlabelled data
  - Examples: clustering and anomaly detection

- Reinforcement learning: software agents ought to take actions in an environment so as to maximize some (notion of) reward
  - Examples: game AIs and robot navigation

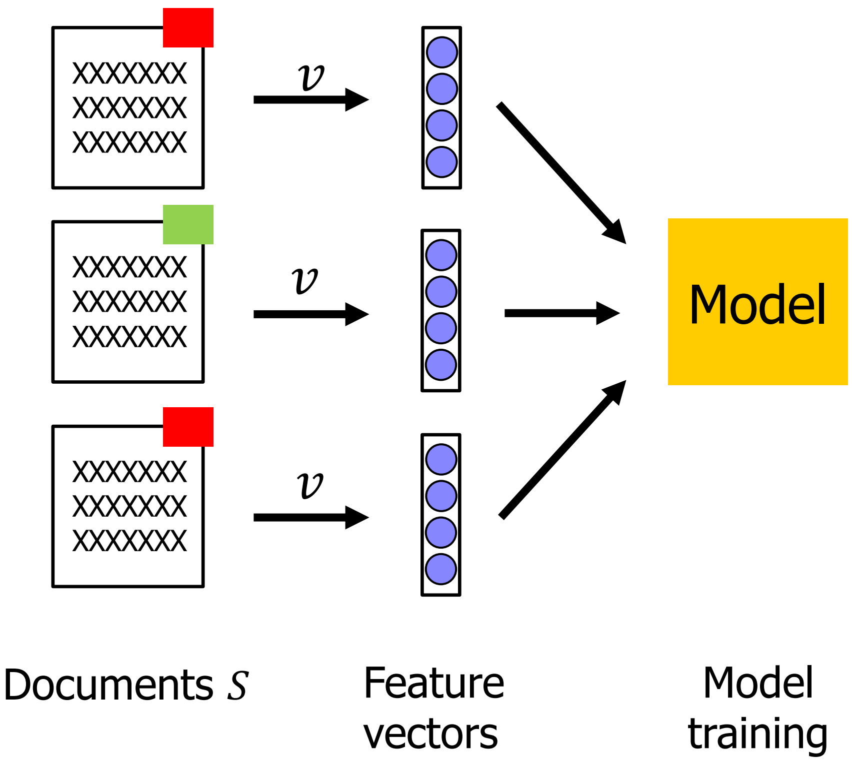
Types of machine learning



https://miro.medium.com/max/2796/1*FUZS9K4JPqzfXDcC83BQTw.png

Supervised Learning

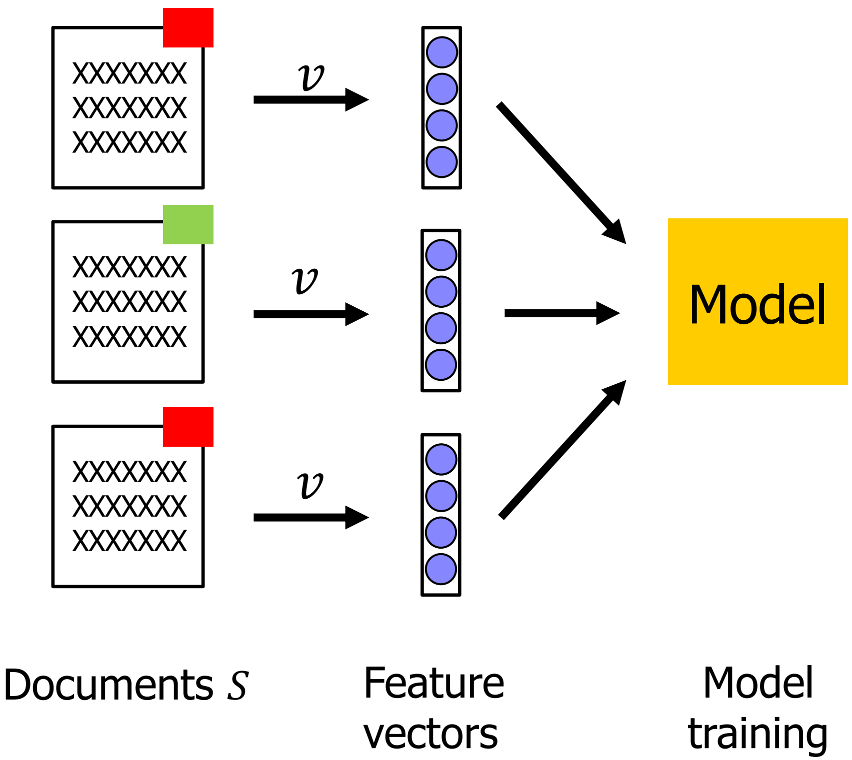
- Given a set D of documents and a set of classes C. A classifier is a function $f: D \rightarrow C$



- Problems
  - Finding enough training data
  - Finding the best pre-processing (tokenization, case, POS tag set …)
  - Finding the best features
  - Finding a good classifier (~ assigning as many docs as possible to their correct class)

Supervised Learning

- Given a set D of documents and a set of classes C. A classifier is a function $f: D \rightarrow C$



- How do we know?
- Use a (separate) gold standard data set
- Use training data in two roles (beware of overfitting)
  - Learning the model
  - Evaluating the model

Problem 1: Overfitting

- Let S be a set of texts with their classes (training data)
- We can easily build a perfect classifier for S
  - f(d) = {f(d’), if ∃d’∈S with d’=d; random otherwise)
  - f is perfect for any doc from S
  - But: produces random results for any new document
- Improvement:
  - f(d) = {f(d’), if ∃d’∈S with d’~d; random otherwise)
  - Improvement depends on |S| and definition of “~”
- Overfitting
  - If the model strongly depends on S, f overfits - it will only work well if all future docs are very similar to the docs in S
  - You cannot find overfitting when evaluation is performed on S only

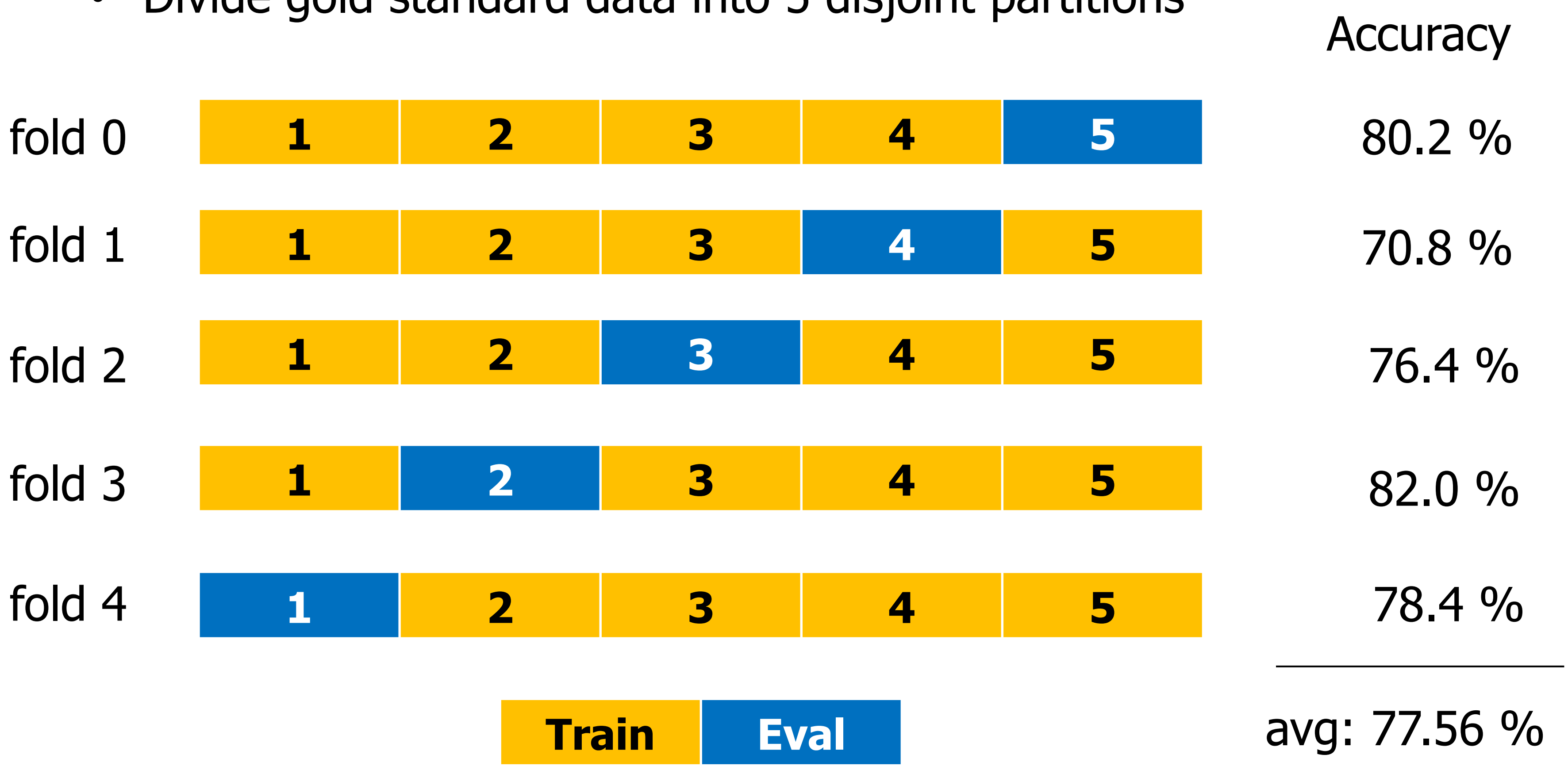
Against Overfitting

- f must generalize: Capture features that are typical for all docs in D, not only for the docs in S
- But usually we only have S for evaluation …
  - We need to extrapolate the quality of f to unknown docs

- Usual method: Cross-validation
  - Divide S into k disjoint partitions (typical: k=10)
  - Learn model on k-1 partitions and evaluate on the k'th
  - Perform k times, each time evaluating on another partition
  - Estimated quality on new docs = average performance over k runs

Cross-validation

- Example $k = 5$:
  - Divide gold standard data into 5 disjoint partitions



Cross-validation

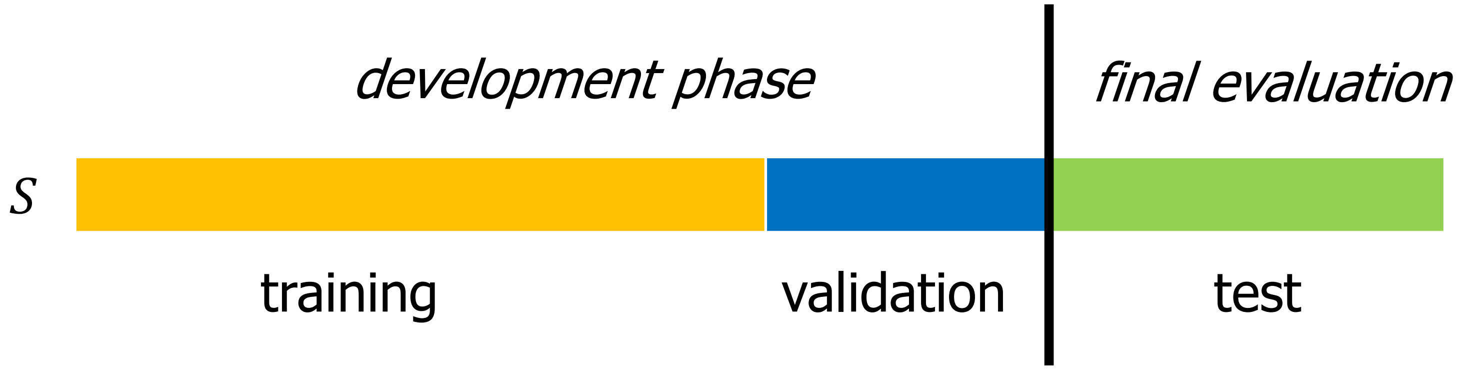
- For complex models cross-validation can be prohibitively expensive and time-consuming
  - We have to train and evaluate k models!
- Alternative: Split S into a (fixed) disjoint training and validation partition
  - Model selection will be performed based on the validation set performance
  - Both partitions should be "representative" for S
    - Same class and feature distribution (e.g. text length)

Problem 2: Information Leakage

- Developing a classifier is an iterative process
  - Define feature space
  - Evaluate performance using cross-validation
  - Perform error analysis, leading to others features / parameters
  - Iterate until satisfied
- In this process, you “sneak” into the data (during error analysis) you later will evaluate on
  - “Information leakage”: information on eval data is used in training
- Solution
  - Reserve a portion P of S for evaluation
  - Perform iterative process only on S\P
  - Final evaluation on P; no more iterations

Data organization

- In general the following data setup for S is used:
  - Training set: train different variants of classifiers (e.g. different methods, pre-processing, feature sets)
  - Validation set: validate performance of the different models and chose the best one
  - Test set: final evaluation of the model on hold-back data



Evaluation metrics (binary model)

- We can group the predictions of a classifier f according to the gold standard S into four categories:

	Truth: True	Truth: False
Classifier: True	True Positives (TP)	False Positives (FP)
Classifier: False	False Negatives (FN)	True Negatives (TN)

- Precision (P): TP/(TP+FP)
  - Fraction of truly true instances in the „answer“ of f
- Recall (R): TP / (TP+FN)
  - Fraction of the truly true instances of S found by f

Evaluation metrics (binary model)

- We can group the predictions of a classifier f according to the gold standard S into four categories:

	Truth: True	Truth: False
Classifier: True	True Positives (TP)	False Positives (FP)
Classifier: False	False Negatives (FN)	True Negatives (TN)

- What is more important – recall or precision?
  - Go to https://menti.com
  - Enter code 77 55 83
  - Submit your answer

Evaluation metrics (binary model)

- We can group the predictions of a classifier f according to the gold standard S into four categories:

	Truth: True	Truth: False
Classifier: True	True Positives (TP)	False Positives (FP)
Classifier: False	False Negatives (FN)	True Negatives (TN)

- F1-Measure: 2*P*R / (P+R)
  - Harmonic mean between precision and recall
  - Favours balanced precision / recall values

Evaluation metrics (binary model)

- We can group the predictions of a classifier f according to the gold standard S into four categories:

	Truth: True	Truth: False
Classifier: True	True Positives (TP)	False Positives (FP)
Classifier: False	False Negatives (FN)	True Negatives (TN)

- Accuracy: TP+TN / (TP+FP+FN+TN)
  - Fraction of correctly predicted instances

- Why not always use accuracy?

Evaluation metrics (binary model)

- We can group the predictions of a classifier f according to the gold standard S into four categories:

	Truth: True	Truth: False
Classifier: True	True Positives (TP)	False Positives (FP)
Classifier: False	False Negatives (FN)	True Negatives (TN)

- Accuracy: TP+TN / (TP+FP+FN+TN)
  - Fraction of correctly predicted instances

- Used in problems with balanced sets of TP+FN / FP+TN
  - Don't use accuracy, if FP+TN >>> TP+FN

Classification Methods

Nearest Neighbour

Support Vector Machine

Classification methods

- There are many classification methods
  - Bayesian Networks, Graphical models
  - Decision Trees and Random Forests
  - Linear / Logistic Regression
  - Perceptrons, Neural Networks [deep learning]
  - …

- Effectiveness of classification depends on problem, algorithm, feature selection method, sample, evaluation, …
- Differences when using different classification methods on the same data/representation are often astonishing small

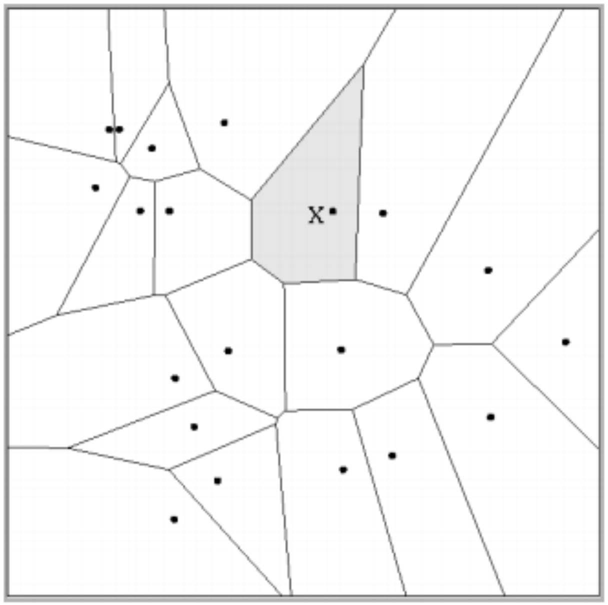
Nearest Neighbor Classifiers

- Definition:

  Let S be a set of classified documents, m a distance function between any two documents, and d an unclassified document

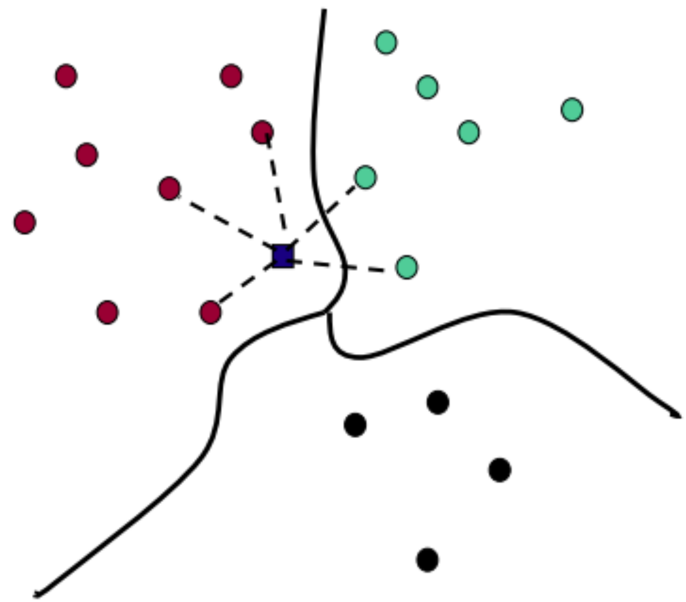
  - A nearest-neighbor (NN) classifier assigns to d the class of the nearest document in S wrt. m
  - A k-nearest-neighbor (kNN) classifier assigns to d the most frequent class among the k nearest documents in S wrt. m
- Remarks
  - Very simple and effective, but slow
  - We may weight the k nearest docs according to their distance to d
  - We need to take care of multiple docs with the same distance

Illustration – Separating Hyperplanes



Voronoi diagram in 2D-space

(for 1NN)



5NN

Properties

- Assumption: Similar docs (in feature space) have the same class; docs in one class are similar
  - Depends a lot on the text representation (bag of words)
  - Depends a lot on the distance function
  - These assumptions can be verified before using a kNN!
- kNN in general more robust than NN
- What do we learn during training of kNN classifier?

Properties

- Assumption: Similar docs (in feature space) have the same class; docs in one class are similar
  - Depends a lot on the text representation (bag of words)
  - Depends a lot on the distance function
  - These assumptions can be verified before using a kNN!
- kNN in general more robust than NN
- Example of lazy learning
  - Actually, there is no learning (only docs)
  - Actually, there is no model (only docs)

Linear Classifiers

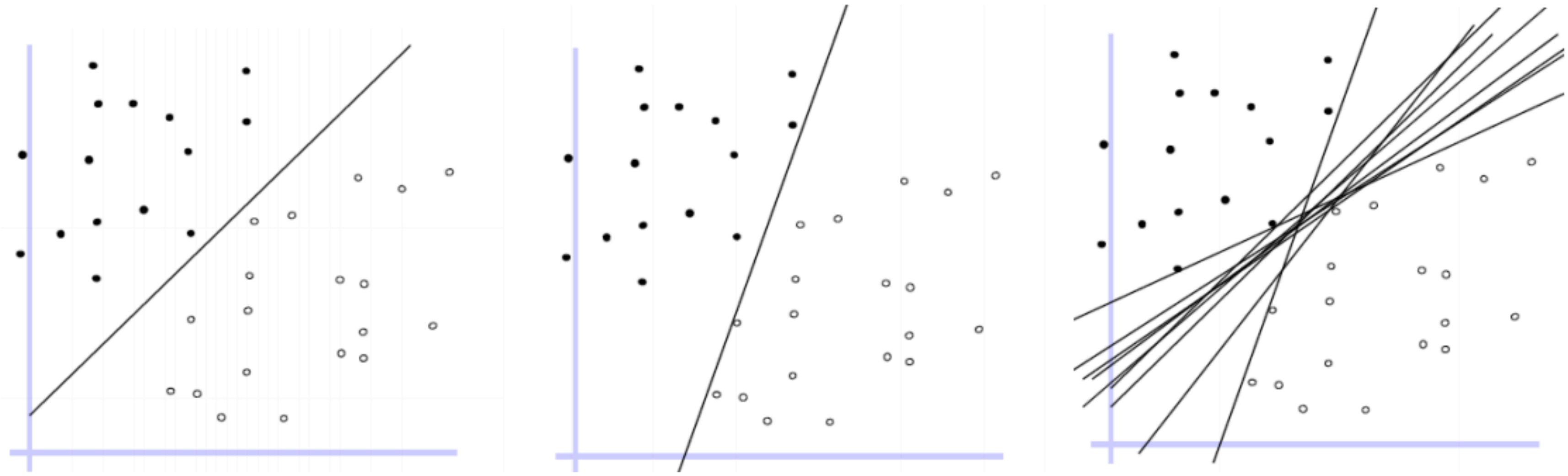
- Many common classifiers are (log-)linear classifiers
  - Naïve Bayes, Perceptron, Linear and Logistic Regression, Maximum Entropy, Support Vector Machines
- If applied on a binary classification problem, all these methods somehow compute a hyperplane which (hopefully) separates the two classes
  - Despite similarity, noticeable performance differences exist – Which feature space is used?
  - Which of the infinite number of possible hyperplanes is chosen?
  - How are non-linear-separable data sets handled?

Characteristics of text data

- High dimensionality: 100k+ features
- Sparsity: Feature values are almost all zero
- Most documents are very far apart (i.e., not strictly orthogonal, but only share very common words)
- Consequence: Most document sets are well separable
  - This is part of why linear classifiers are quite successful in this domain
- The trick is more of finding the “right” separating hyperplane instead of just finding (any) one

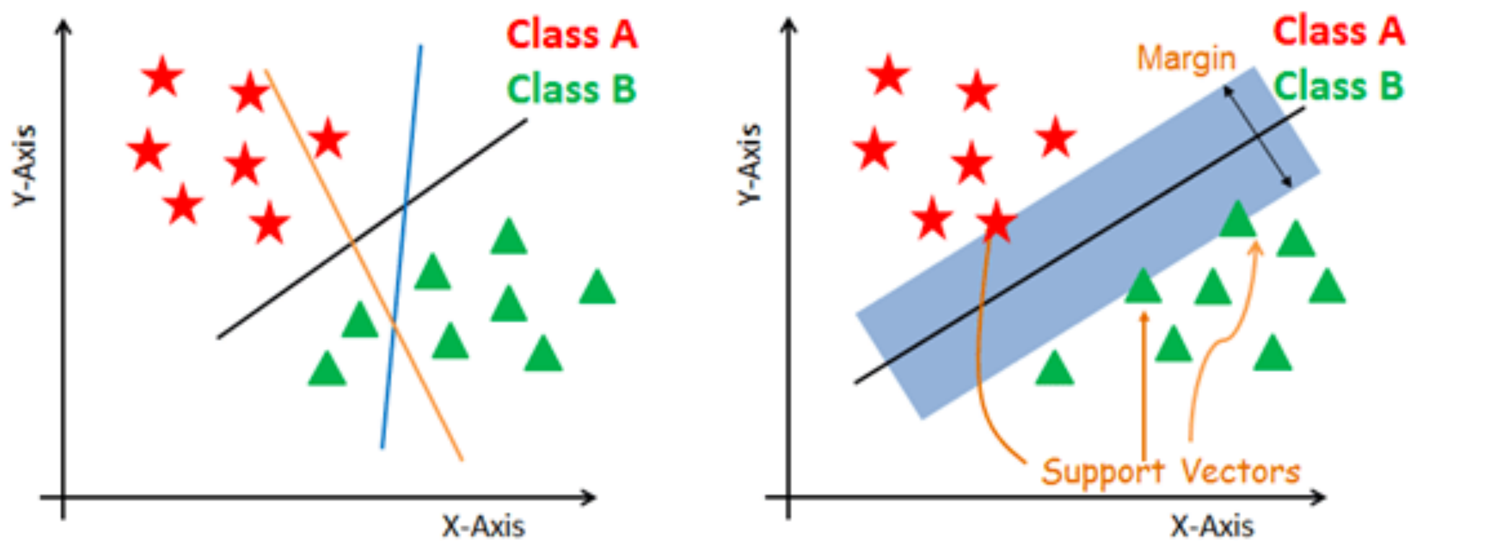
Example: Linear Classifiers – 2D

- Hyperplane separating classes in high dimensional space
- But which?



Support Vector Machine (SVM) - Idea

- SVMs: Hyperplane which maximizes the margin
  - I.e., is as far away from any data point as possible
  - Cast in a linear optimization problem and solved efficiently
  - Classification only depends on support vectors – efficient
    - Points most closest to hyperplane



http://res.cloudinary.com/dyd911kmh/image/upload/f_auto,q_auto:best/v1526288454/index2_ub1uzd.png

Artificial neural networks

Motivation

Feed forward networks

How to find good features?

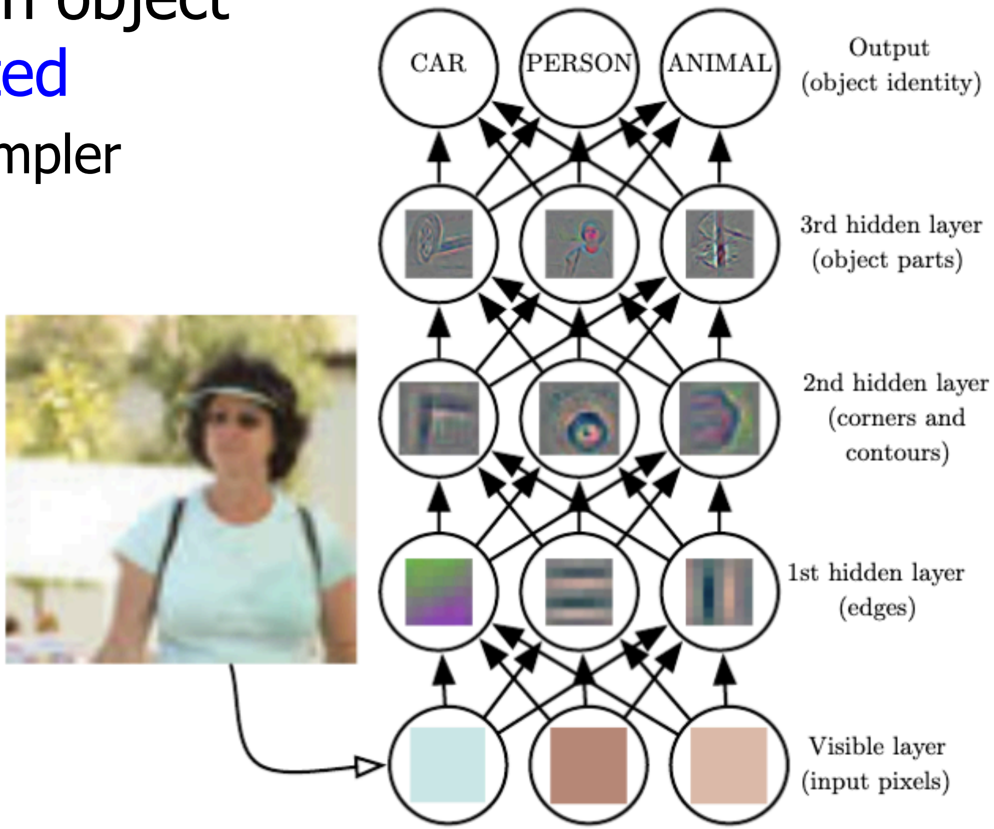
- Many AI tasks can be solved by designing the right set of features to extract and apply a (simple) ML approach
  - However, for many tasks is difficult to know what features should be extracted
  - Can take decades for an entire community of researchers for difficult problems
- Example: Identify cars in photographs
  - We know cars have wheels – presence of a wheel maybe a good feature
  - Unfortunately it is hard to describe a wheel in terms of pixels
  - Simple geometric shape – but it’s image may be complicated by shadows falling on it, the sun glaring off the metal parts, ...

Representation Learning / Deep Learning

- Of course, it is very difficult to extract such high-level features / factors from raw data
  - Need very sophisticated (nearly human-level) understanding of the raw data
- One solution to the problem: representation learning (RL)
  - Use machine learning to discover not only the mapping from representation to output but also the representation itself
  - Representation learning ~ feature learning
- Deep Learning: A RL technique that learns represent-tations that are expressed by simpler representations
  - Build more complex concepts out of simpler concepts

Example: image classification

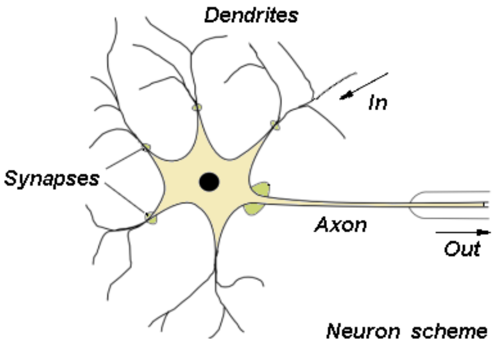
- Mapping from pixels to an object identity is very complicated
  - Instead, use a series of simpler nested mappings
- Every layer builds a higher abstractions based on the former layer's output
- Final layer uses most abstract representations to make the prediction



http://www.deeplearningbook.org/contents/intro.html

Artificial neural networks (~ Deep Learning)

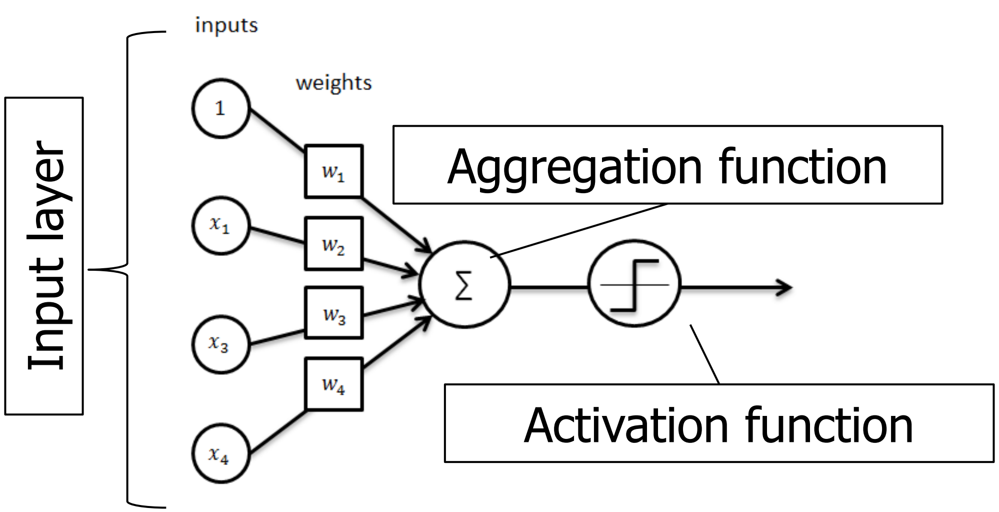
- A method for non-linear classification
  - Long history - but also forgotten for a long time
    - First works range back to the 1950s / 60s
    - Extremely hyped since about 2005
  - Basic concepts inspired by biological networks
    - But, it isn't the goal to simulate / model these networks

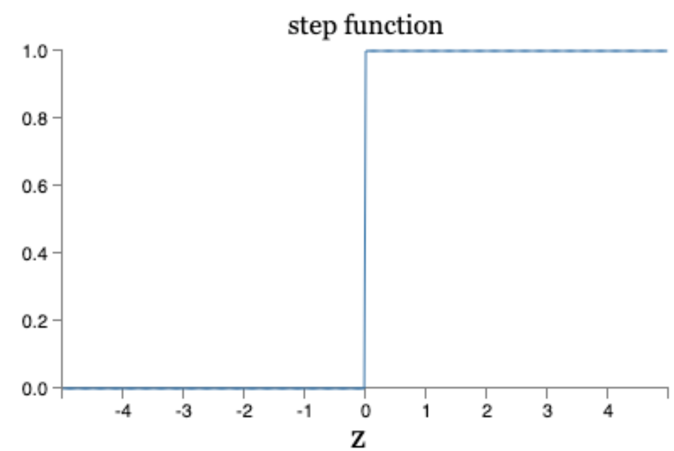


- Today: state-of-the-art in machine translation, image recognition, gaming, machine reading, ...

Concepts

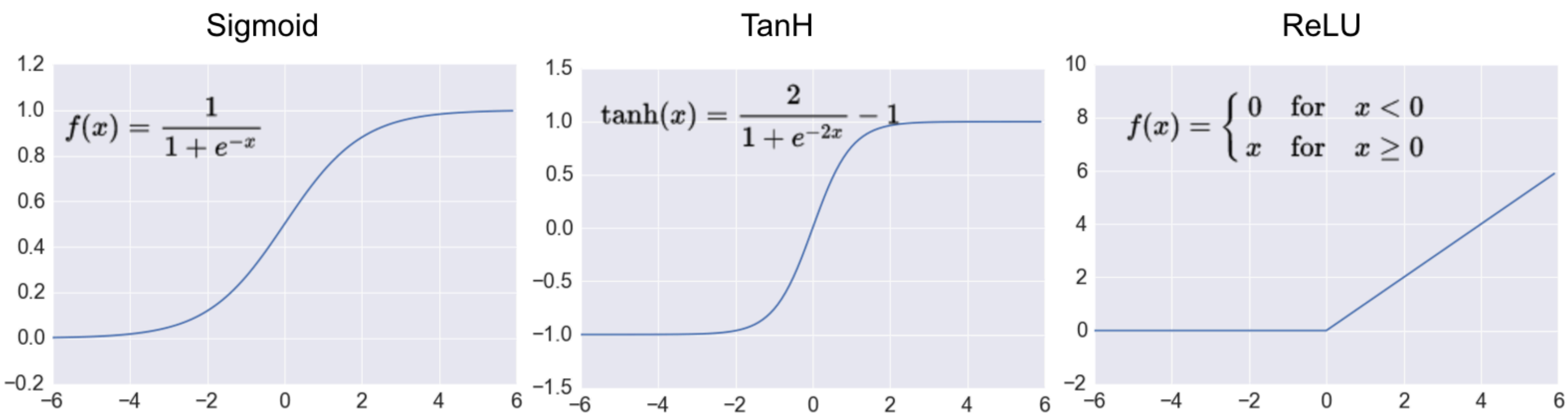
- ANNs are composed of artificial neurons (~ basic unit)
  - Neurons receive input signals from input data or other neurons
  - Input signals will be weighted and aggregated to a scalar value through an aggregation function (e.g. weighted sum)
  - Neuron's output is determined by an activation function (e.g. 1 if weighted sum is > 0 or else 0)





Activation functions

- Use activation functions with a continuous value range
  - Small changes in the weights and biases cause only a small change in their output
  - Often: activations saturate for very large and/or small values

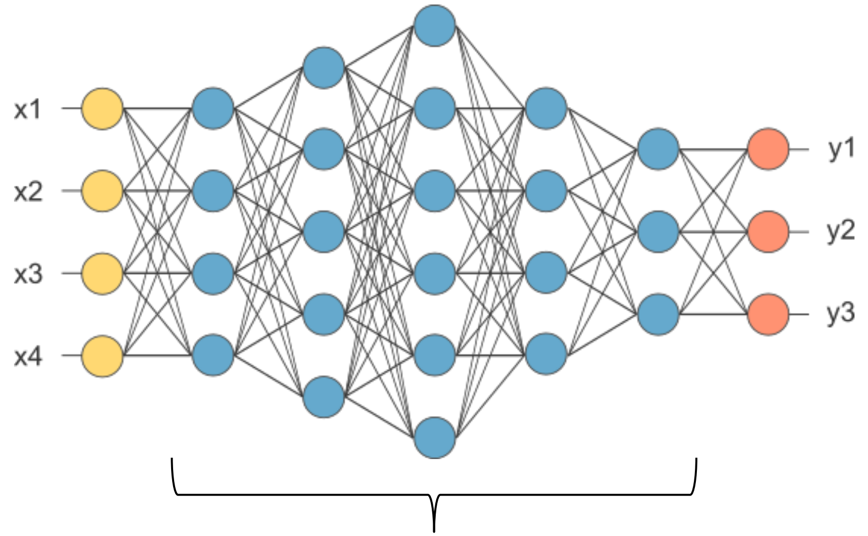


http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/

Structure of neural networks

- Neurons are organized and stacked in layers
  - The neurons of each layer work on the activations from the former layer
  - Each layer learns more complex abstractions (~ decisions) of the input based on the former layer’s abstractions

Input layer



Output layer

Hidden layers

http://www.opennn.net/

Example: Text classification

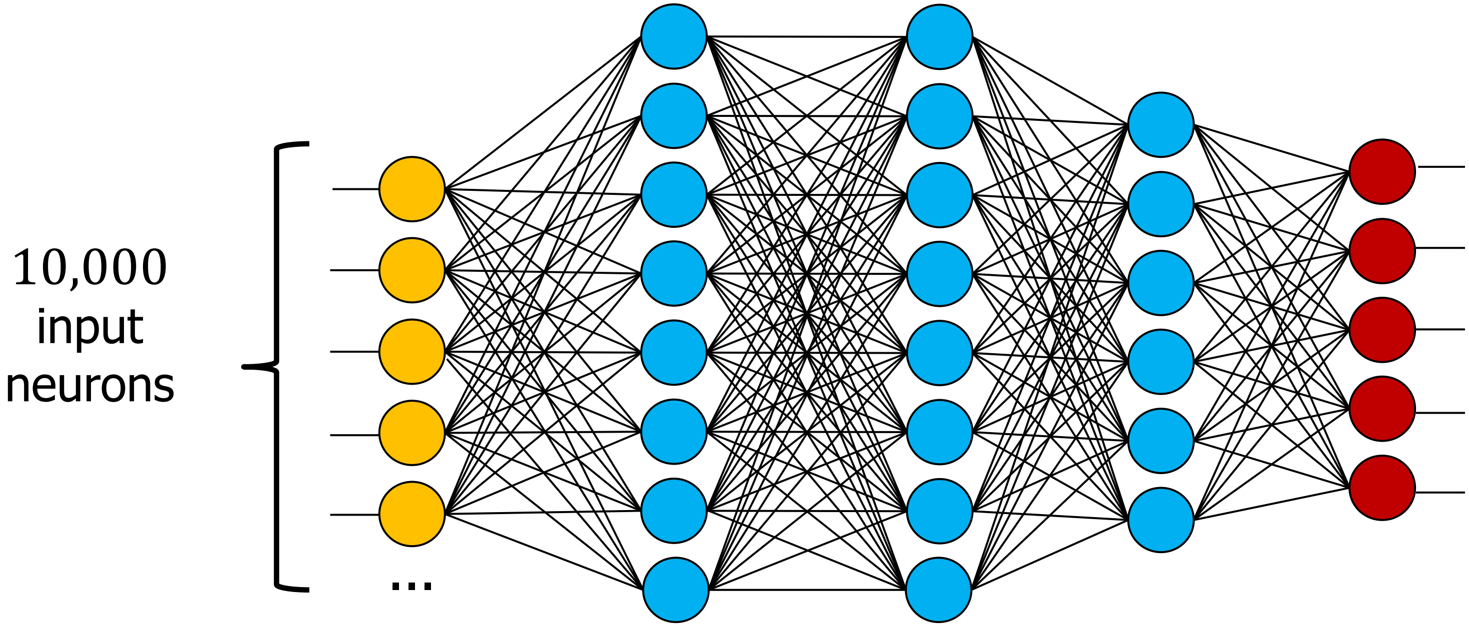
- Let's suppose, we have a corpus of news articles and we want to perform automatic categorization of these articles
  - We want to distinguish articles from five different categories: politics, economy, culture, lifestyle, sport

- Assume we have a set S of labelled examples (x_i, y_i)
  - x_i : TF-IDF vector of text from article i (details next slide)
  - y_i : The gold standard label for article i
    - In following we will often refer to the label as one-hot encoded vector

	politics	economy	culture	lifestyle	sport
$y(x_1) = economy$	0	1	0	0	0
$y(x_2) = lifestyle$	0	0	0	1	0

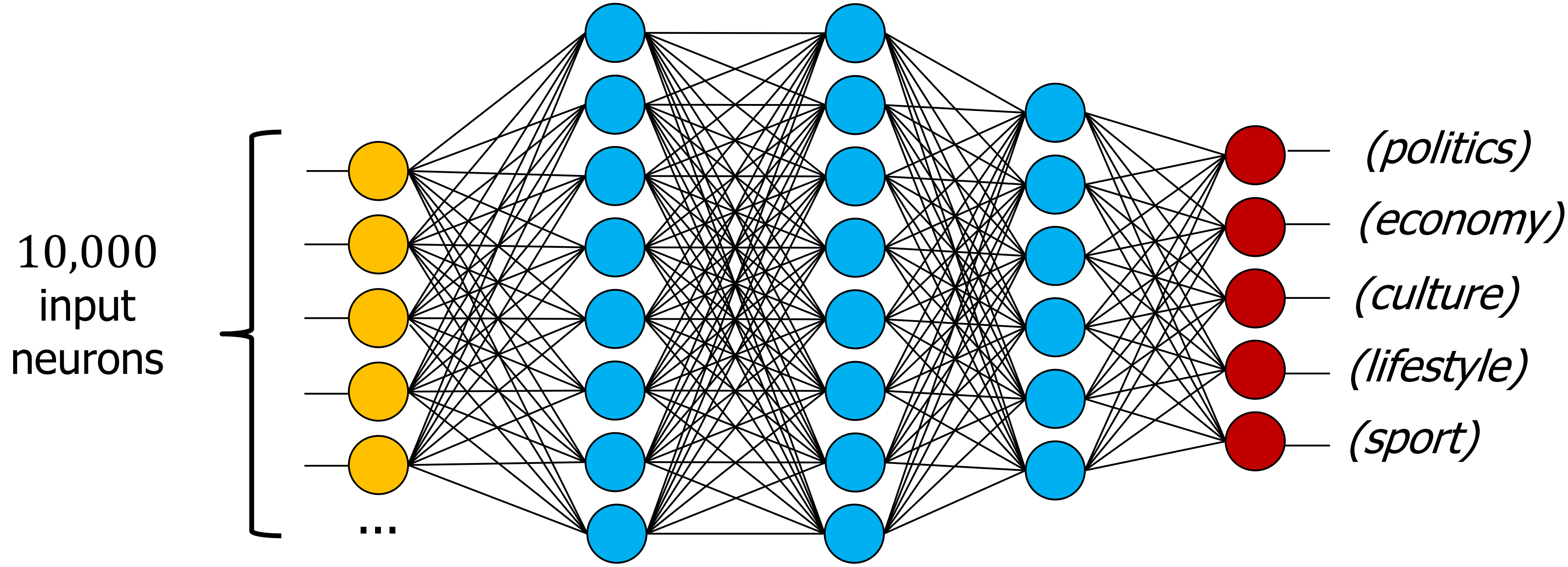
Example: Text classification

- Input: TF-IDF vectors of the articles
  - Let's say we have a vocabulary with 10.000 distinct token
  - Each component of the vector is modelled as separate input neuron



Example: Text classification

- Output: One of the five classes politics, economy, culture, lifestyle, sport
  - Each class gets one dedicated neuron in the output layer
  - We select the output neuron which fires resp. has the highest activation as prediction



Learning a ANN

- Feedforward (and many other) ANN can be efficiently learned using backpropagation
- Idea
  - Initialize weights at random
  - Compute loss function for training samples
  - Adjust weights level wise along the gradient of the loss function
  - Repeat until convergence
  - Trick: Fast and repeated computation of the gradients
- Variation of stochastic gradient descent (SGD)

Example: cost function

- Quadratic cost function (mean squared error)

$$C(w,b) = \frac{1}{|S|}\sum_i \|y_i - a_i\|^2$$

  - w and b all weights and biases of the network
  - $n = |S|$ is number of training examples
  - a_i activation of the neurons in the output layer

Example: cost function

- Quadratic cost function (mean squared error)

$$C(w,b) = \frac{1}{|S|}\sum_i \|y_i - a_i\|^2$$

	politics	economy	culture	lifestyle	sport
y_1	0	1	0	0	0
a_1	0.2	0.4	0.2	0.1	0.1
$y_1 - a_1$	-0.2	0.6	-0.2	-0.1	-0.1
$\|y_1 - a_1\|^2$	0.04	0.36	0.04	0.01	0.01
			0.46		

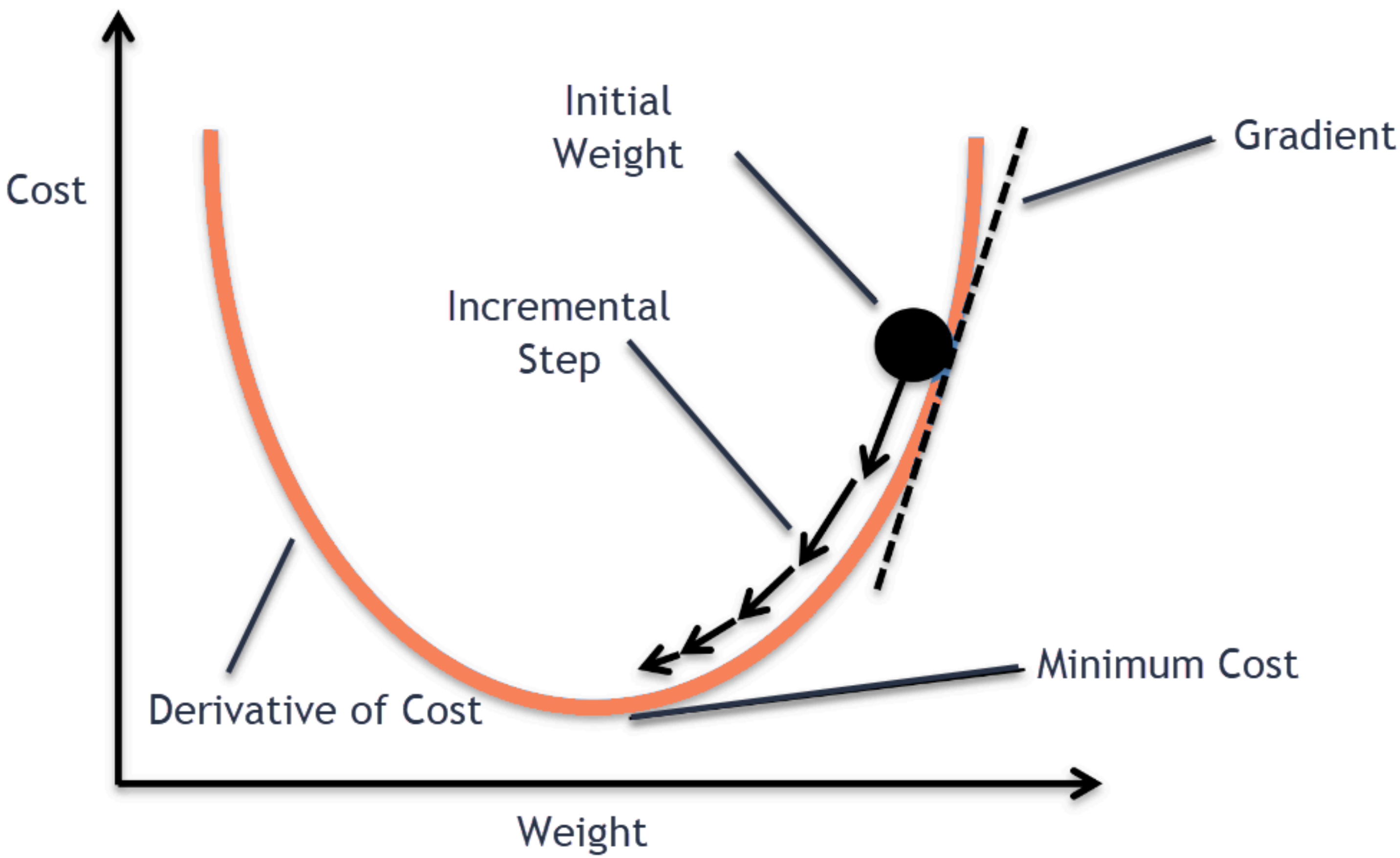
Example: cost function

- Quadratic cost function (mean squared error)

$$C(w,b) = \frac{1}{|S|}\sum_{i} \|y_i - a_i\|^2$$

  - w and b all weights and biases of the network
  - $n = |S|$ is number of training examples
  - a_i activation of the neurons in the output layer
- Properties
  - $C(w,b)$ becomes small $C(w,b) \approx 0$ when y_i is approximately equal to the network output a_i for all instances
  - In contrast, a large $C(w,b)$ means that output a_i is not close to y_i for many instances

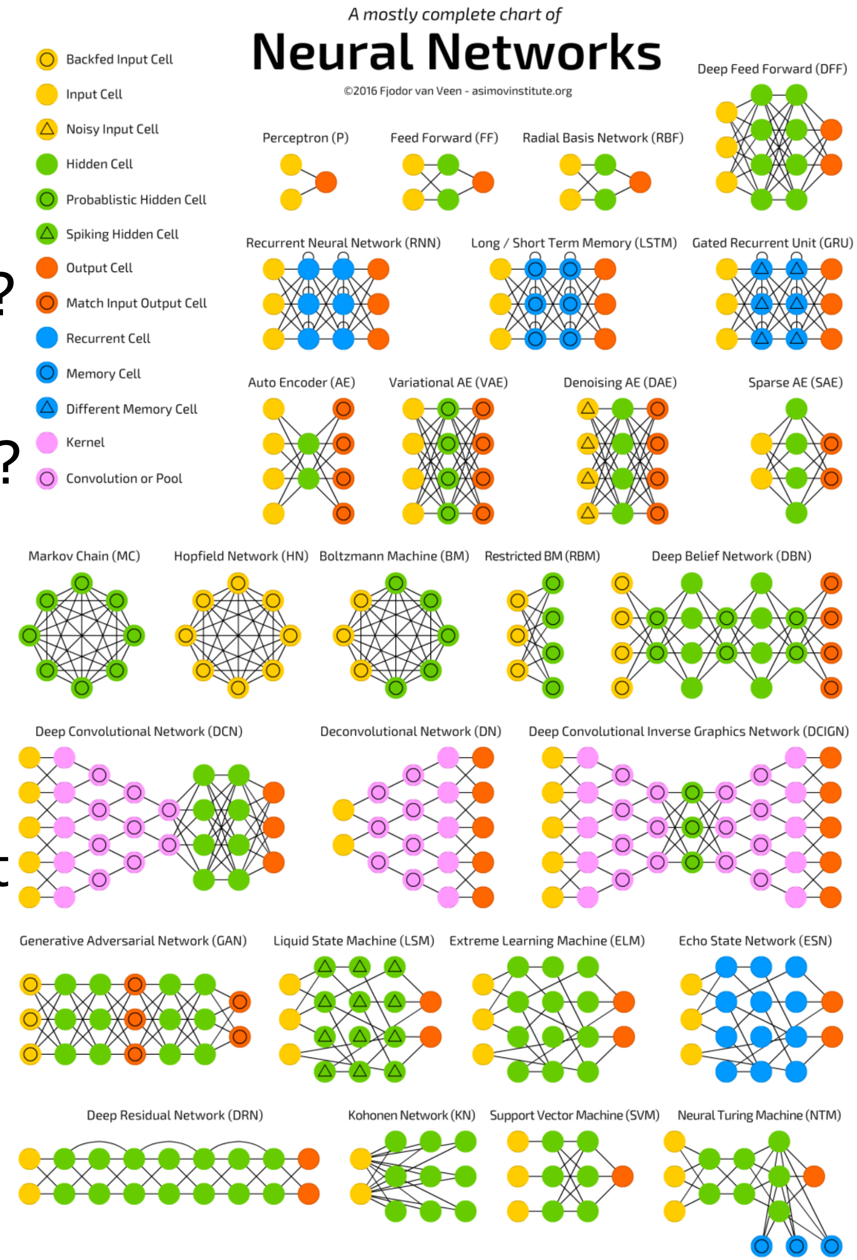
Gradient Descent (sketch)



https://miro.medium.com/max/1005/1*_6TVU8yGpXNYDkkpOfnJ6Q.png

Many design choices

- Activation (aggregation) function?
- Number of hidden layers?
- Number of units per hidden layer?
- Connections only between adjacent layers?
- Only "forward" connections?
- Central issue: "Learnability"
  - Different choices lead to different problems
  - Especially back-links increase complexity (and expressiveness)



Organization

Frameworks & Courses

Next steps

NLP frameworks - Python

- scikit-learn (ML)
  - https://scikit-learn.org/stable/
- PyTorch (DL)
  - https://pytorch.org/
- Tensorflow (DL)
  - https://www.tensorflow.org/
- Keras (DL)
  - https://keras.io/

NLP frameworks - Java

- Weka 3 Workbench (ML)
  - https://www.cs.waikato.ac.nz/ml/weka/
- LibSVM (ML)
  - https://www.csie.ntu.edu.tw/~cjlin/libsvm/
- DeepLearning4J (DL)
  - https://deeplearning4j.org/

General literature

- Text books:
  - Manning et al.: *Foundations of statistical natural language processing* (Online)
  - Manning et al.: *Introduction to Information Retrieval* (Online)
  - Bishop: *Pattern recognition and machine learning* (Online)
  - Hastie et al.: *The Elements of Statistical Learning* (Online)
  - Goodfellow et al.: *Deep learning* (Online)

Online courses

- US San Diego – Machine Learning:
  - https://www.youtube.com/playlist?list=PL_onPhFCkVQhUzcTVgQiC8W2ShZKWlm0s

- Fast.ai – Introduction to machine learning:
  - https://www.fast.ai/2018/09/26/ml-launch/

- Coursera – Machine learning:
  - https://de.coursera.org/learn/machine-learning

- Stanford - Natural Language processing with deep learning:
  - https://www.youtube.com/playlist?list=PLoROMvodv4rOhcuXMZkNm7j3fVwBBY42z

Next steps

- I will ....
  - ... send you literature hints and recommendations
  - ... release the training data until end of next week

- You have to ...
  - ... get familiar with your topic
  - ... communicate with your group members
  - ... become acquainted with the framework you want to work
  - ... start to implement your classification pipeline

- Please contact me until 29.05. to discuss your approach

General recommendations

- Experiment with different variants of your approach
  - Investigate different data pre- and post-processing steps
  - Try different feature selection strategies, perform hyperparameter-search, use additional information, ...

- There are a plethora of tutorials and blog posts in internet
  - Do not blindly copy source code – understand what you are do

- Don’t be afraid to ask questions
  - Get in touch with me instead of being stuck with a problem for weeks

Thank you for your attention!

Questions?