
Course

# Foundations of Artificial Intelligence

## Lecture 09 – Supervised Machine Learning Algorithms

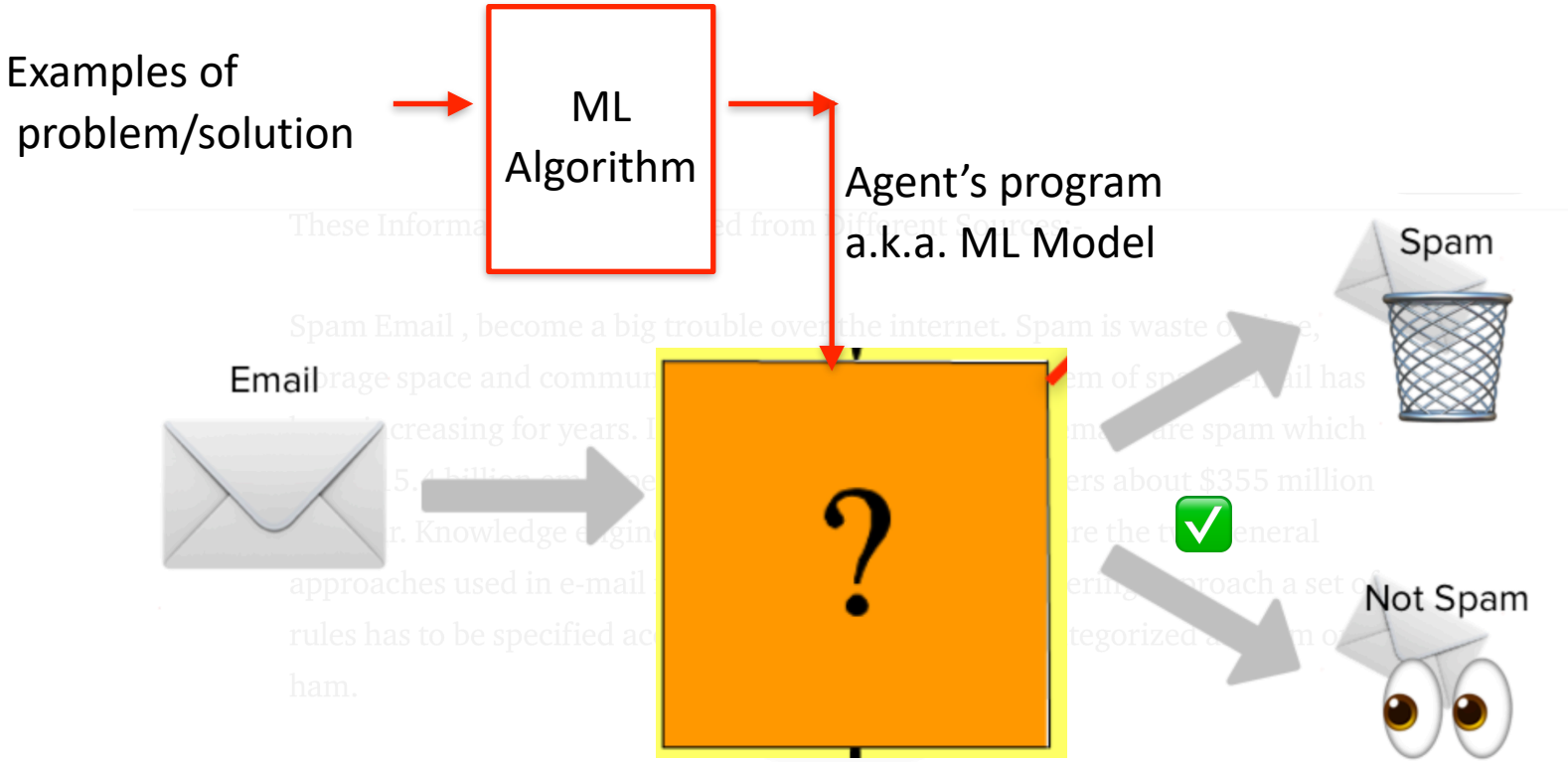
Dr. Mohsen Mesgar

Universität Duisburg-Essen

# Recall …

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken



# Any other open questions?

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

# In this lecture, you learn about …

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

- Basic ML algorithms
  - Decision Trees
  - kNN
  - Naive Bayes classifier
  - Support Vector Machines (SVMs)

# Decision Trees

# Name gender

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

Tom

Andrea

Dorchadas

Brigitte

Laraine

Lumen

Jamel

Diarmid

Simone

Dafne

Lea

Lumen

Wilhelm

Noor

Dietgard

Keisuke

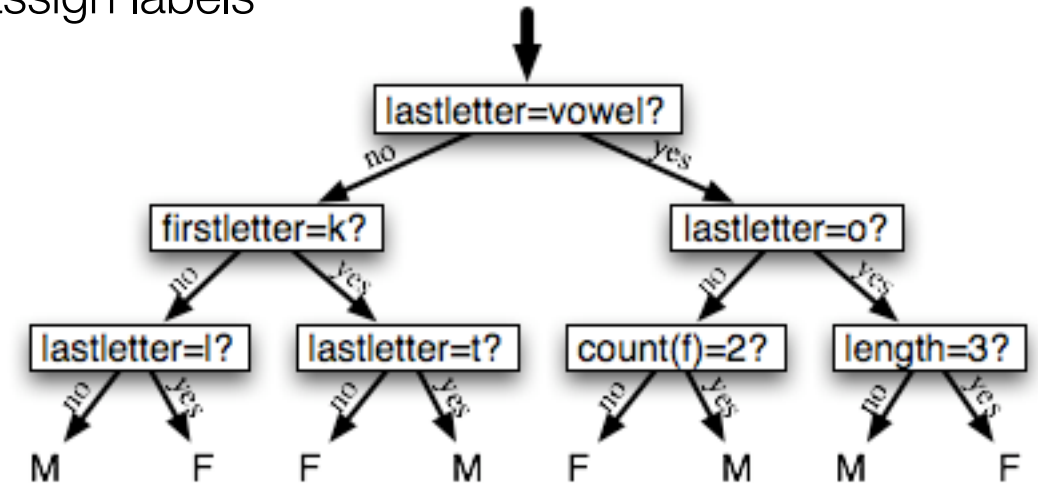
# Decision trees

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

Flowchart that selects labels for input values

- Decision nodes check feature values
- Leaf nodes assign labels




Start with the root node (decision stump)

- Use feature value that classifies all input best
- Continue growing using most informative features

# Play golf example

UNIVERSITÄT DUISBURG ESSEN

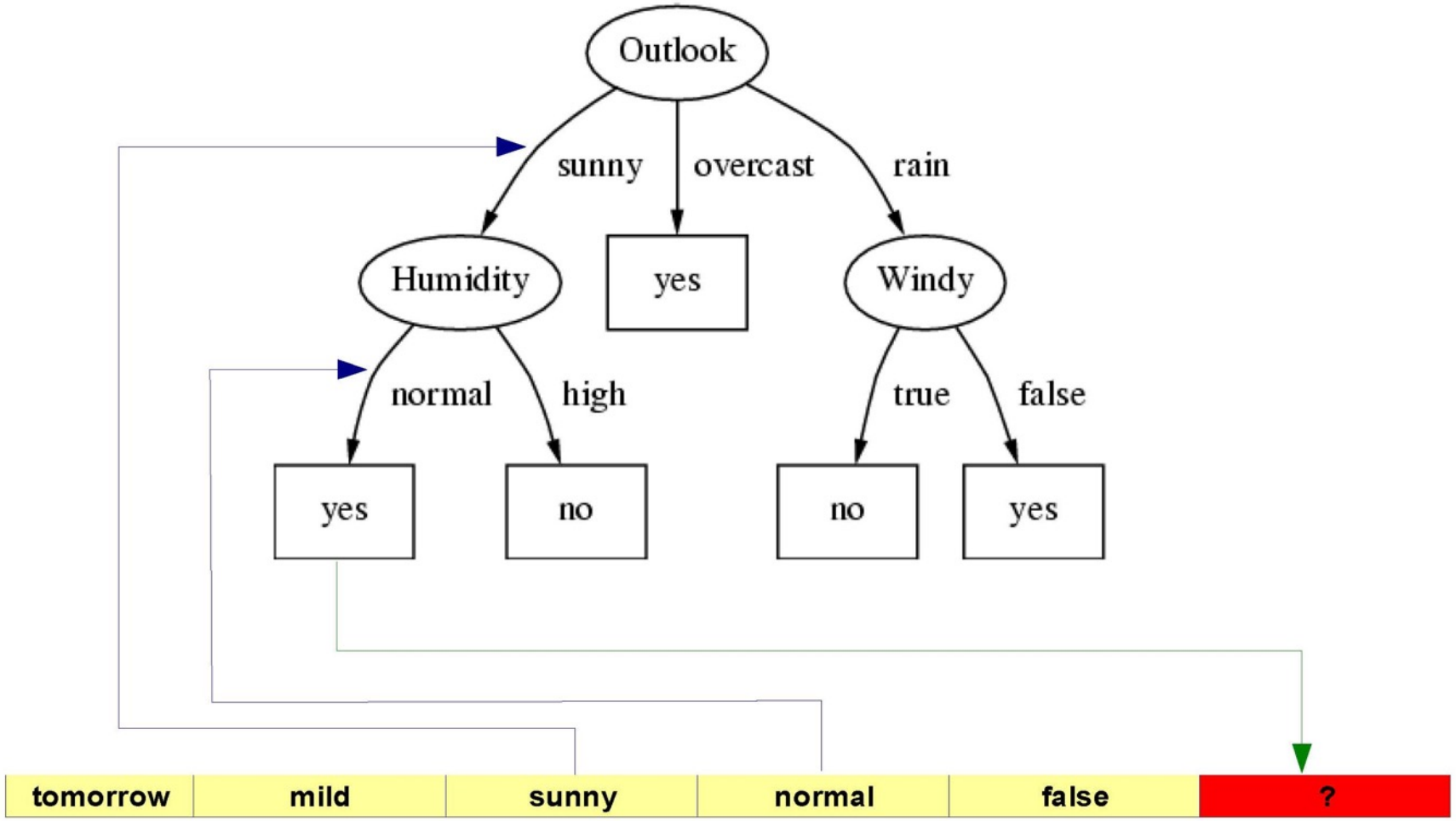
*Offen im Denken*

| Day | Temperature | Outlook | Humidity | Windy | Play Golf? |
|---|---|---|---|---|---|
| 07-05 | hot | sunny | high | false | no |
| 07-06 | hot | sunny | high | true | no |
| 07-07 | hot | overcast | high | false | yes |
| 07-09 | cool | rain | normal | false | yes |
| 07-10 | cool | overcast | normal | true | yes |
| 07-12 | mild | sunny | high | false | no |
| 07-14 | cool | sunny | normal | false | yes |
| 07-15 | mild | rain | normal | false | yes |
| 07-20 | mild | sunny | normal | true | yes |
| 07-21 | mild | overcast | high | true | yes |
| 07-22 | hot | overcast | normal | false | yes |
| 07-23 | mild | rain | high | true | no |
| 07-26 | cool | rain | normal | true | no |
| 07-30 | mild | rain | high | false | yes |

| | | | | | |
|---|---|---|---|---|---|
| today | cool | sunny | normal | false | ? |
| tomorrow | mild | sunny | normal | false | ? |

# Final decision tree





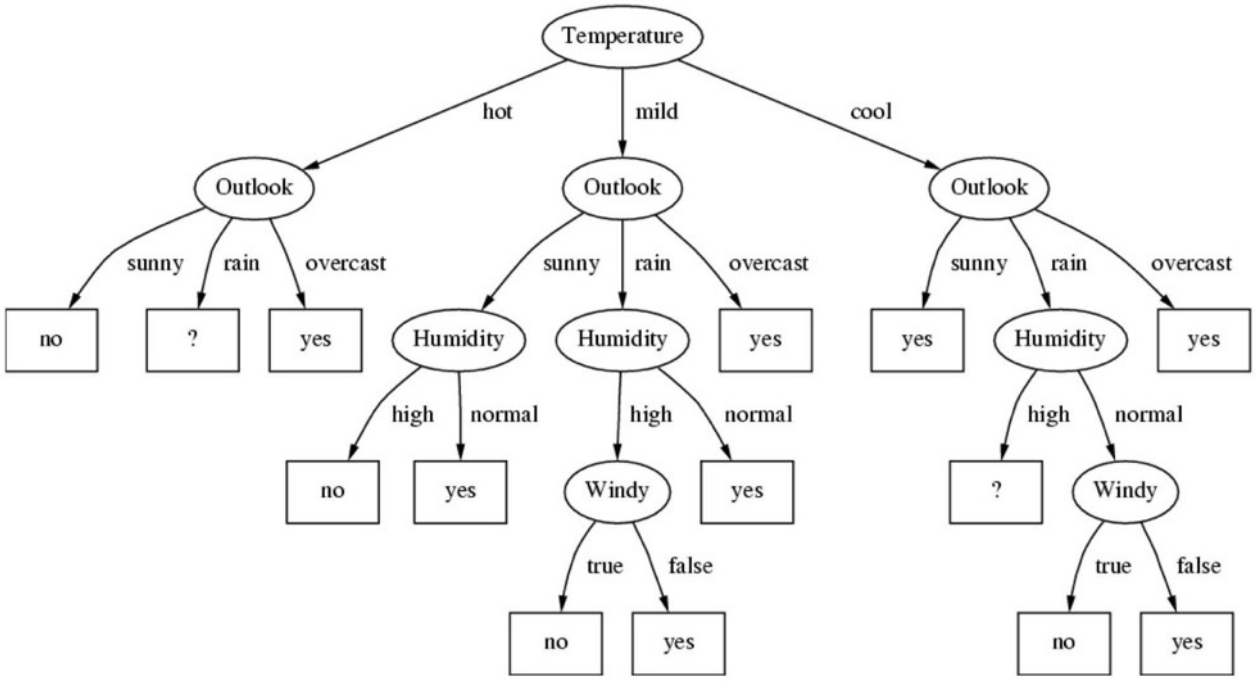
# How do we find the best decision tree?

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

- Search through all possible trees? → Too expensive.
- This tree also explains all the training data.
  Will it generalize well to new data?

→ Determine good decision nodes based on information gain.



# Entropy (for two classes)

UNIVERSITÄT DUISBURG ESSEN

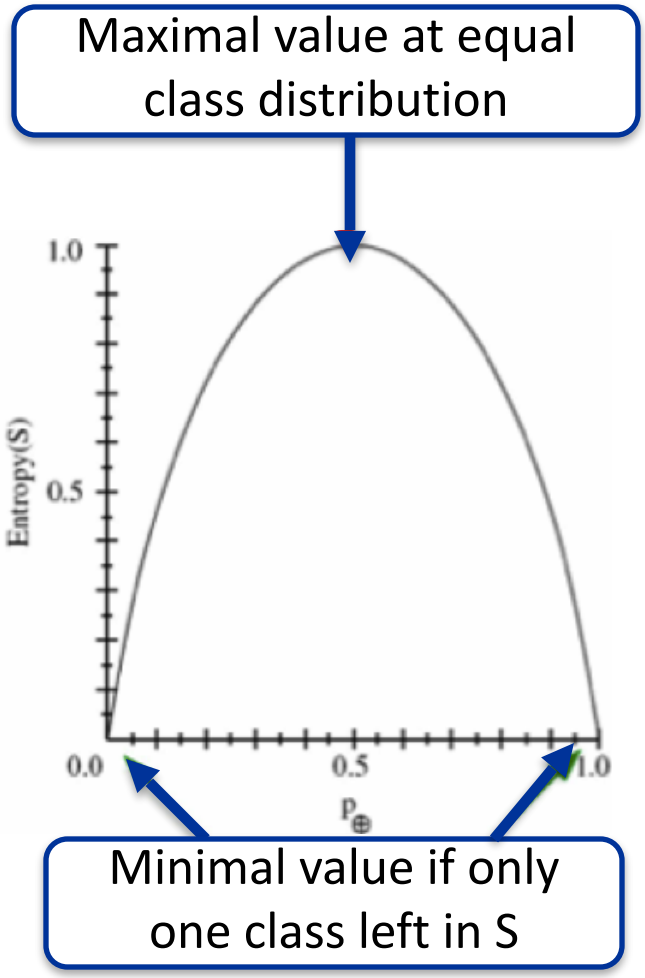
*Offen* im Denken

- S is a set of examples
- $P_i$ is the proportion of examples in class i
- $P_j = 1 - p_i$ is the proportion of examples in the other class j

$$E(S) = -p_i \log_2 p_i - p_j \log_2 p_j$$

Interpretation:

- Amount of unorderedness in the class distribution of S



# Average entropy / information

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

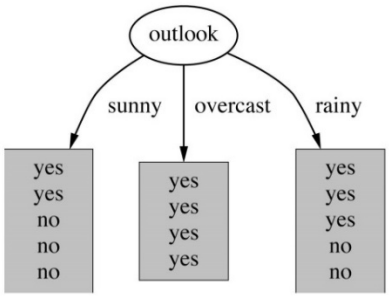
Problem:

- Entropy only computes the quality of a single (sub-)set of examples
  - Corresponds to a single value
- How can we compute the quality of the entire split?
  - Corresponds to an entire attribute

Solution:

- Compute the weighted average over all sets resulting from the split
  - Weighted by their size

$$I(S,A)=\sum_i \frac{|S_i|}{|S|}\cdot E(S_i)$$




Example:

- Average entropy for attribute Outlook:

$$I(\text{Outlook})=\frac{5}{14}\cdot 0.971+\frac{4}{14}\cdot 0+\frac{5}{14}\cdot 0.971=0.693$$

# Decision tree example

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

outlook=sunny (3 yes/2 no)

$$\mathbf{Entropy(sunny)} = -\frac{3}{5} * log\frac{3}{5} - \frac{2}{5} * log\frac{2}{5} = \mathbf{0.97}$$

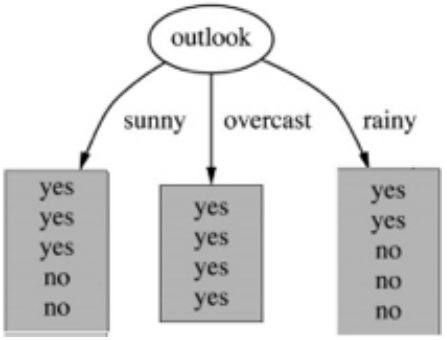
log(0) is undefined

outlook=overcast (4 yes/0 no)

$$\mathbf{Entropy(overcast)} = -1 * \log 1 - 0 * \log 0 = 0$$

outlook=rainy (2 yes/3 no)

$$\mathbf{Entropy(rainy)} = -\frac{2}{5} * log\frac{2}{5} - \frac{3}{5} * log\frac{3}{5} = \mathbf{0.97}$$




Information(outlook) =

$$\frac{5}{14} * 0.97 + \frac{4}{14} * 0 + \frac{5}{14} * 0.97 = 0.69$$

# Information of features

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

Feature humidity

- Entropy(high) = 0.99 (3 yes/4 no)
- Entropy(low) = 0.59 (6 yes/1 no)
- $\text{Information(humidity)} = \frac{7}{14} * 0.99 + \frac{7}{14} * 0.59 = 0.79$

Feature wind

- Entropy(weak) = 0.81 (6 yes/2 no)
- Entropy(strong) = 1.0 (3 yes/3 no)
- $\text{Information(wind)} = \frac{8}{14} * 0.81 + \frac{6}{14} * 1 = 0.89$

Feature temperature

- Entropy(hot) = 1.0 (2 yes/2 no)
- Entropy(mild) = 0.92 (4 yes/2 no)
- Entropy(cool) = 0.81 (3 yes/1 no)
- $\text{Information(temperature)} = \frac{4}{14} * 1 + \frac{6}{14} * 0.92 + \frac{4}{14} * 0.81 = 0.91$

# Information Gain

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

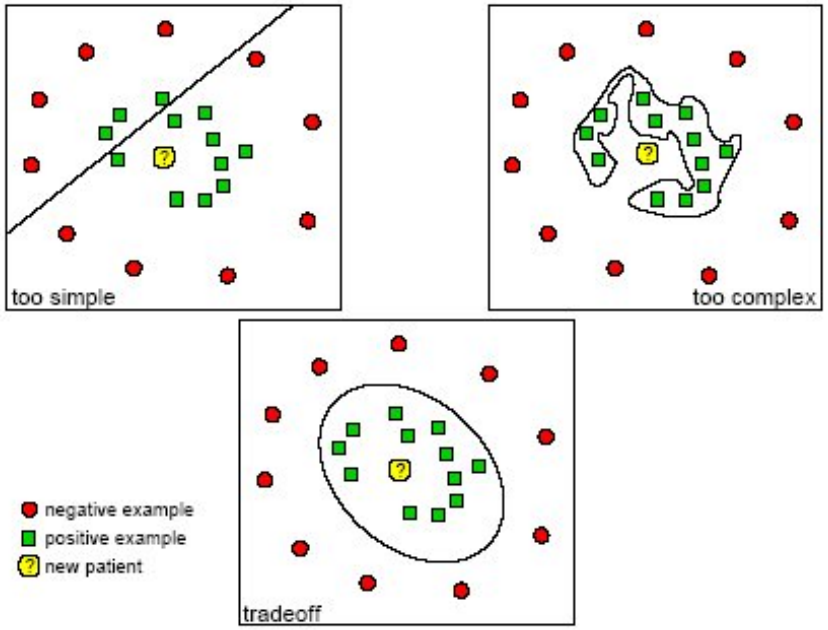
- Entropy(before) = 0.94 (9 yes/5 no)

- $\text{InformationGain(outlook)} = 0.94 - 0.69 = 0.25$
- $\text{InformationGain(humidity)} = 0.94 - 0.79 = 0.15$
- $\text{InformationGain(wind)} = 0.94 - 0.89 = 0.05$
- $\text{InformationGain(temperature)} = 0.94 - 0.91 = 0.03$

- Select feature outlook as root node

# Underfitting & overfitting






## Underfitting

- the model is not capable of learning the (complex) patterns in the training set

→ use better features or classifier

## Overfitting

- The model perfectly learns to classify the training data, but has bad generalization ability → high test error
- results in useless model for unseen data
- typical for small sample sizes combined with powerful models

# Overfitting and pruning

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

The smaller the complexity of a concept, the less it overfits the data

- a polynomial of degree *n* can always fit *n*+1 points

Thus, learning algorithms try to keep the learned concepts simple

- note a "perfect" fit on the training data can always be found for a decision tree! (except when data is contradictory)

Pre-pruning

- stop growing a branch when information becomes unreliable

Post-pruning

- grow a decision tree that correctly classifies all training data
- simplify it later by replacing some nodes with leaves

Post-pruning preferred in practice, because pre-pruning can "stop early"

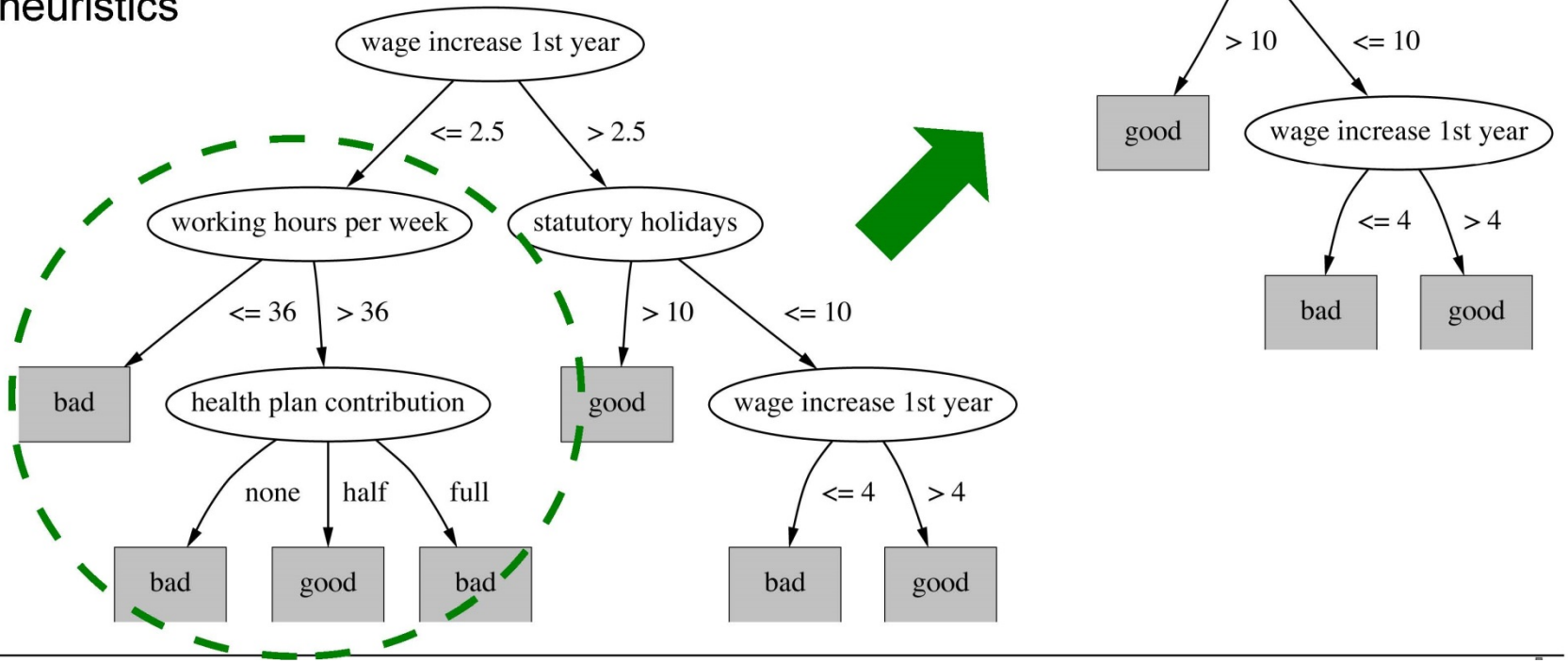
# Sub-tree replacement

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

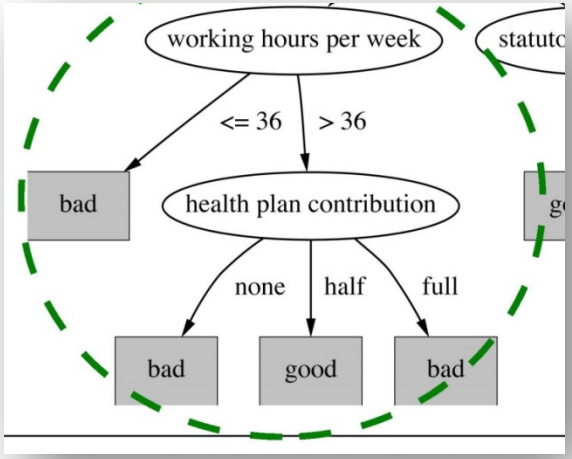
Proceeds Bottom-up:

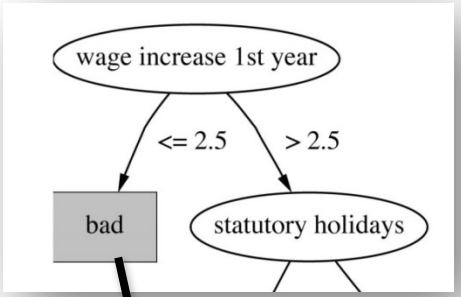
- consider replacing a tree only after considering all its subtrees
- may make a difference for complexity-based heuristics

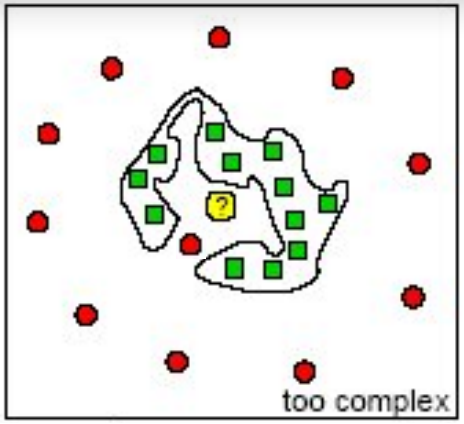


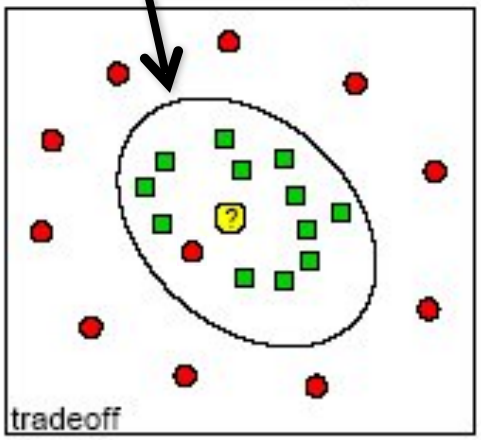
# Sub-tree replacement





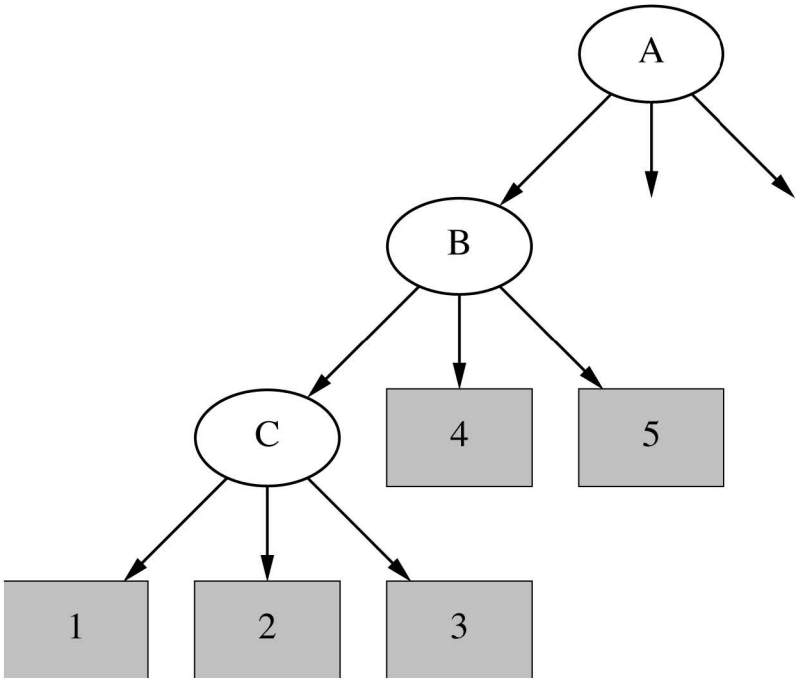




Overfitted

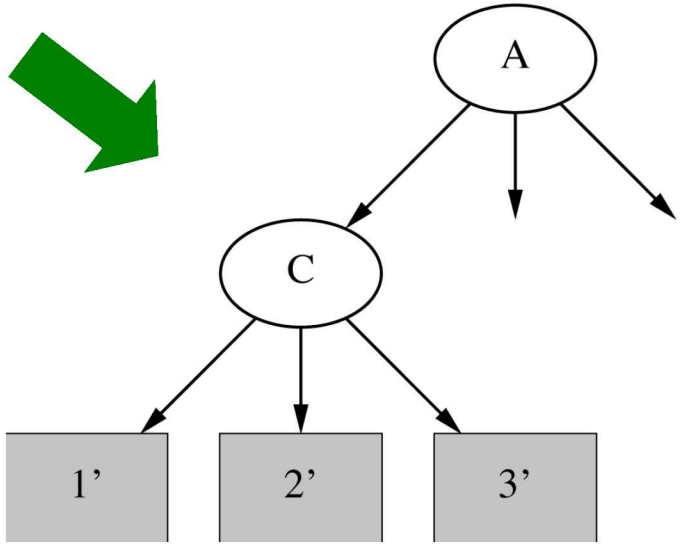
Pruned

# Sub-tree raising






- Delete node B
- Redistribute instances of leaves 4 and 5 into C



# Decision Trees

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

## Advantages

- efficient to create (training is fast)
- simple to interpret → liked by users, explicit knowledge: white box model
- useful to learn models for data which can be hierarchically categorized

## Problems

- overfitting is a serious problem (idiosyncrasies in the data set)
- simpler, seemingly less accurate trees are often preferable → pruning
- **amount of training data in lower nodes** is quite small
  - because it is split at every decision node
- f**orces features to be checked in a certain order**, even if features are independent of each other

# k-nearest neighbor (KNN)

# Nearest neighbor prediction

UNIVERSITÄT DUISBURG ESSEN

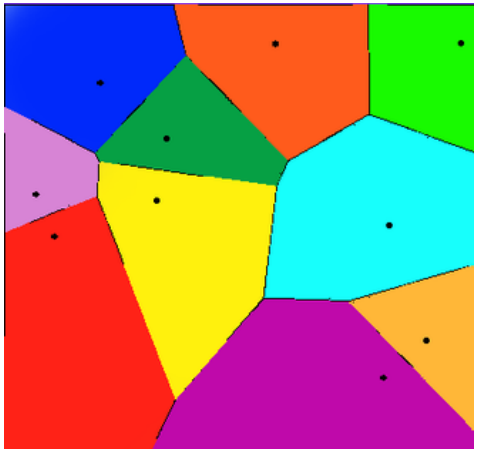
*Offen* im Denken

Simply predict the class of the nearest neighbor of the unseen instance in the representation space.

**Voronoi Diagram:** nearest neighbor in 2-dimensional space

- example: 10 training instances
- colored area: neighbor region of each training instance
- determine color for test instance
- assign class of training stance

- How do we determine "nearness"?



# Similarity

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

- Nearness in feature space = similarity between represented concepts
- **Assumption:** concepts that have similar values in many dimensions (= features) are similar.

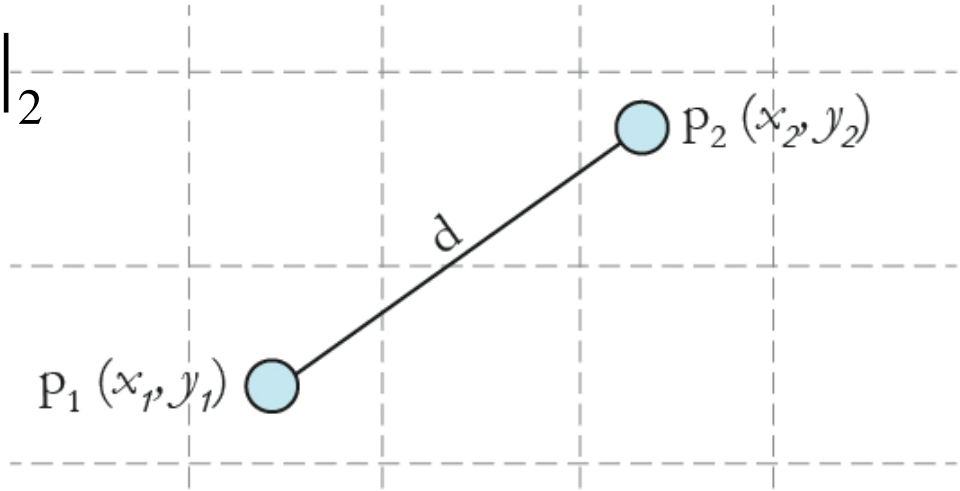
- **How do we measure similarity?**
  By measuring the **distance** between two feature vectors

# Euclidean distance

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

- Straight line distance between two vectors
- In 2-dimensional space —> Pythagoras
- In n-dimensional space: $\mathbf{d}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$
- also known as $L_2$-norm: $\|x - y\|_2$



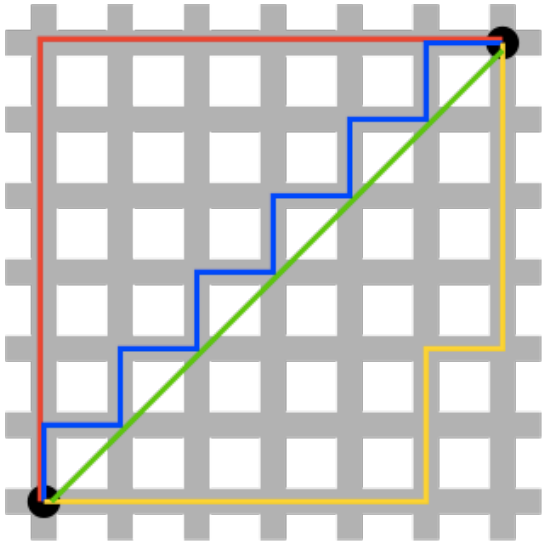

https://www.analyticsvidhya.com/blog/2015/08/learning-concept-knn-algorithms-programming/

# Manhattan distance

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

- Straight line is often not possible
- Step-wise distance between two points:
  sum of the absolute differences of their Cartesian coordinates.
- $$\mathbf{d}(\mathbf{x}, \boldsymbol{y}) = \sum_{i=1}^{n} \left| \boldsymbol{x}_i - \boldsymbol{y}_i \right|$$
- also known as $L_1$-norm: $\|x - y\|_1$



- red, blue, yellow: Manhattan distance
- green: Euclidean distance

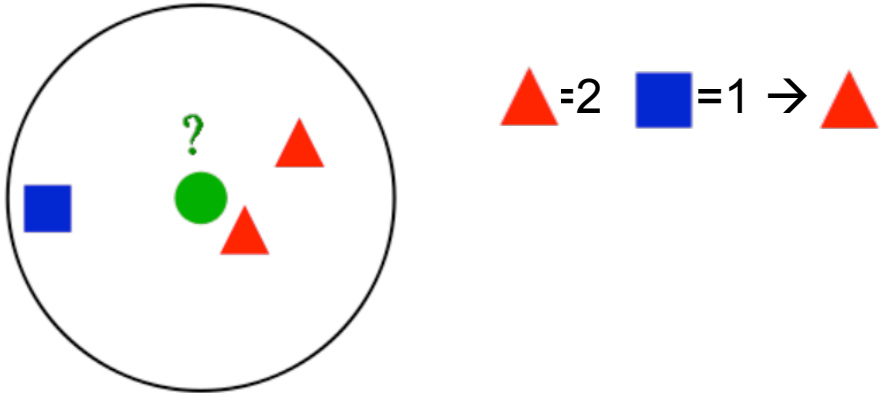
# k-nearest neighbor classification

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

Assign majority class of the k-nearest neighbors.

**k=3**



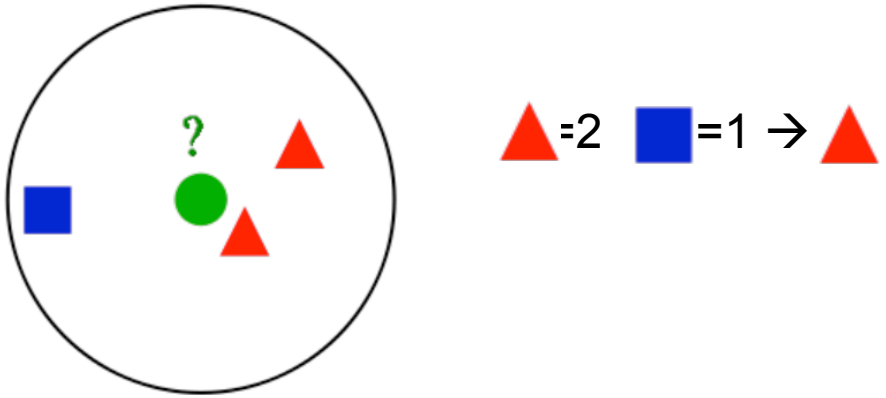
# k-nearest neighbor classification

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

Assign majority class of the k-nearest neighbors.

**k=3**




- No training!
- Just store all instances
  → might build special lookup data-structure to optimize comparison
- k is a hyperparameter

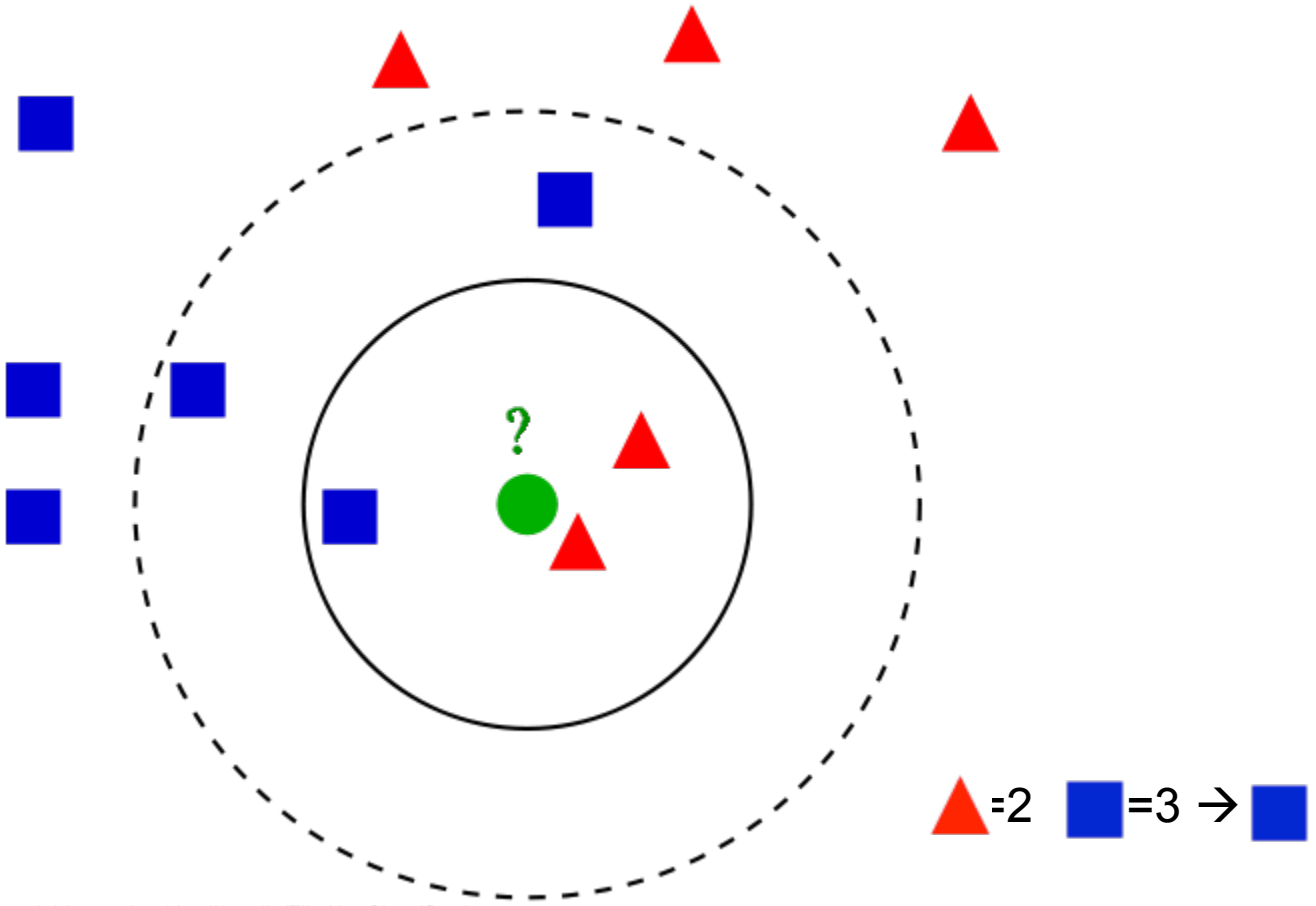
Also known as ***Collaborative Filtering*** or ***Instance-based Learning***.

# Choice of k

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

3-nearest neighbors vs. 5-nearest neighbors prediction




https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#/media/File:KnnClassification.svg

# Nearest neighbor search

UNIVERSITÄT DUISBURG ESSEN

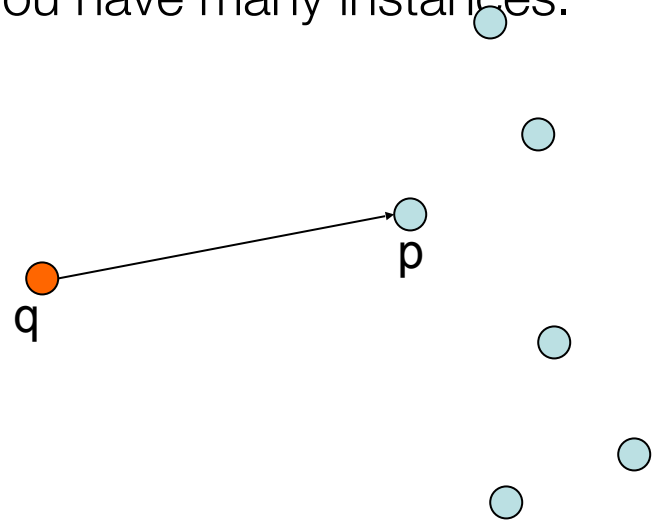
*Offen* im Denken

- **Given:** a set $P$ of $n$ points in $R^d$ , a test instance q
- **Goal:** find the nearest neighbor $p$ of $q$ in $P$
- **Naïve approach:** compute the distance from the query point to every other point in the database, keeping track of the **"best so far"**

**The naïve approach is very expensive,** if you have many instances.

→ $n$ comparisons for each test instance.

→ Apply more efficient search algorithms.



# Properties of kNN algorithm

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

## Inductive bias

- similar classification of nearby instances
- similar to human heuristics

## Curse of dimensionality

- similarity metric might be misled by irrelevant attributes

# Naïve Bayes

# Probabilistic classifier

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

- Calculate a probability distribution over all classes (not only a single class)
- More realistic setting for many tasks → model uncertainty

http://vis.stanford.edu/papers/termite

# Example task: Document classification

UNIVERSITÄT DUISBURG ESSEN

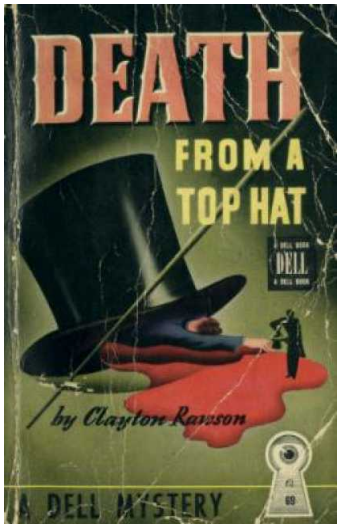
*Offen* im Denken

We have a training dataset with documents from 3 different categories:

***Automotive***


***Sports***

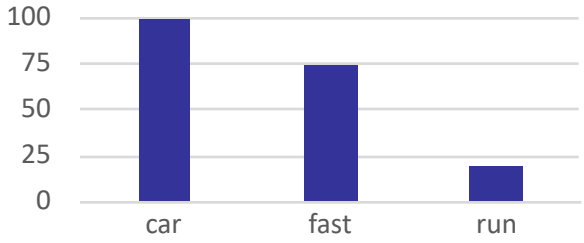

***Murder Mystery***



# Feature representation: words in the document

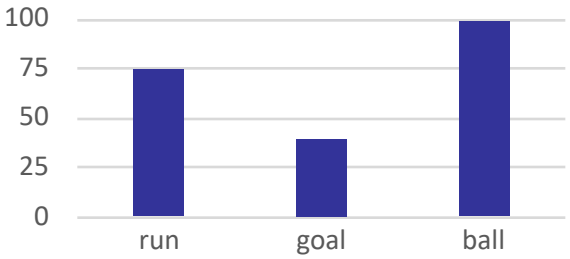
UNIVERSITÄT DUISBURG ESSEN

*Offen im Denken*

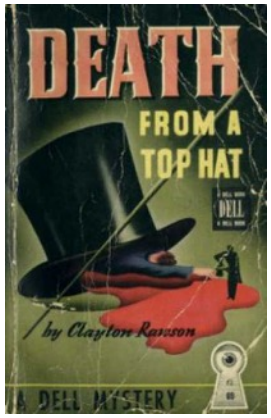



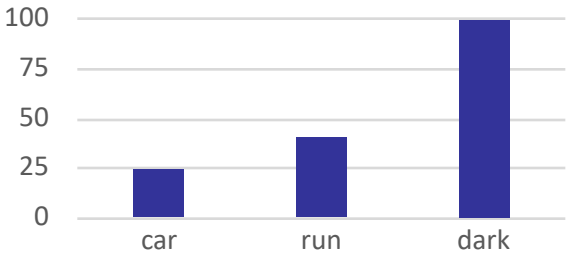

Word frequency

Some words are prototypical for a category. Each occurrence of a word in a document is an indicator for the category.


Word frequency




Word frequency

# Prior probability

UNIVERSITÄT DUISBURG ESSEN

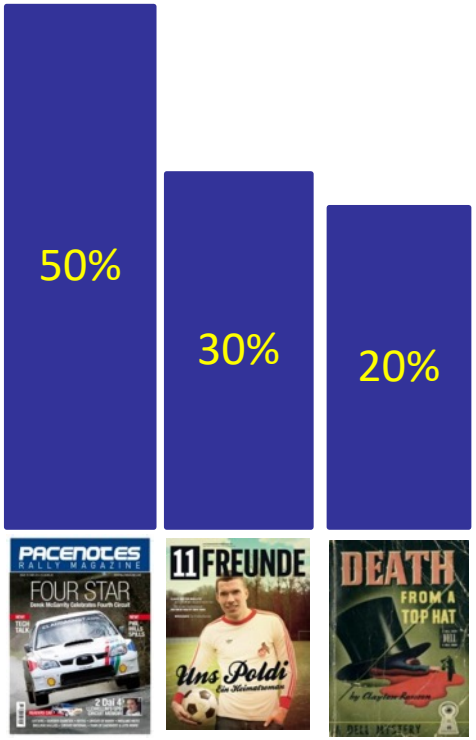
*Offen* im Denken

- How likely is a category?

→ Frequency of each category in the training data

- **If we assume that training data are representative,**

→ we expect that about half of the documents in the test data belong to the automotive category




Prior probability

# Likelihood = Prior prob * feature contribution

UNIVERSITÄT DUISBURG ESSEN

*Offen im Denken*

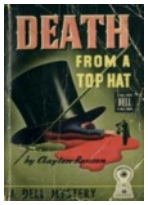
50%


* run * fast * … = Likelihood for automotive

30%


* run * fast * … = Likelihood for sport

20%




* run * fast * … = Likelihood for mystery thriller

# Naïve Bayes classifier

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

**Goal: find the class with the maximum probability.**

$$\max_{c \in classes} \frac{p\left(\left[x1, x2, x3, \ldots x_n\right] \middle| c\right) * p(c)}{p(\left[x1, x2, x3, \ldots x_n\right])}$$

The denominator is a constant and can be dropped:

$$\max_{c \in classes} p\left(\left[x1, x2, x3, \ldots x_n\right] \middle| c\right) * p(c)$$

It is very unlikely that we find an instance with the exact same feature representation in the training data.

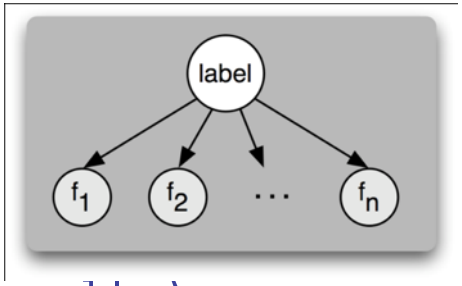
→ Assume independence of features!

# Independence assumption

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

- Given a label, all features are independent of each other.
- This assumption makes it much easier to combine different features, as we do not need to worry about interactions between features.




The conditional probability that feature $x_i$ occurred in class $c$ in the training data.

$$p\left(\left[x1, x2, x3, \ldots x_n\right] \middle| c\right) \; -- > \; \text{independence assumption:} \; \prod_{i=1}^{n} p(xi \mid c)$$

Naïve Bayes Classifier:

$$\max_{c \in \text{classes}} \prod_{i=1}^{n} p\left(xi \middle| c\right) * p(c)$$

# The naïvety of independence

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

It's unreasonable to assume that all features are independent.

- If *football* occurs, it's more likely that *goal* also occurs in the same text.

What happens if we ignore this dependence?

- Possibility of double-counting highly correlated features
- *football* and *goal* are highly correlated, but assumed independent, so each feature is counted separately in the training corpus
- This pushes the classifier closer to a given label than justified.

→ Correlated features are problematic for Naïve Bayes classifiers

# Naïve Bayes: Summary

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

- This is a probabilistic learning method which uses knowledge about the prior probabilities of classes and about the conditional probability of features occurring in a particular class.
- Training is very fast.
- Binary, numeric and nominal features can be mixed.
- If the independence assumption is violated too much, the classifier fails.
  - Identical or highly correlated features are problematic.

  → Can be addressed with feature selection algorithms

# Support Vector Machines (SVM)

# How to split a feature space into 2 classes?

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

- 1-dimensional ( = 1 feature)

→ By a point: if feature x >= 0.7 then class 1 else class 2

- 2-dimensional ( = 2 features)

→ By a line: $w\ x - b$

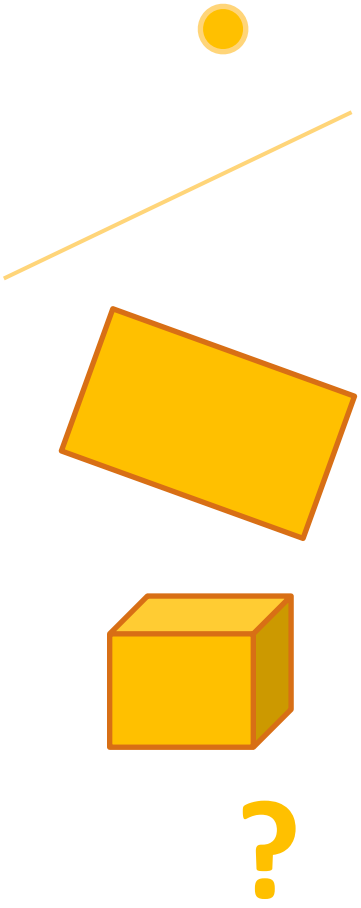
- 3-dimensional ( = 3 features)

→ By a plane: $w_1\ x_1 + w_2\ x_2 - b$

- n-dimensional ( = n features)

→ By a hyperplane: $\vec{w} \bullet \vec{x} - b$

- A hyperplane in $R^n$ is a $n-1$ dimensional subspace of $R^n$.

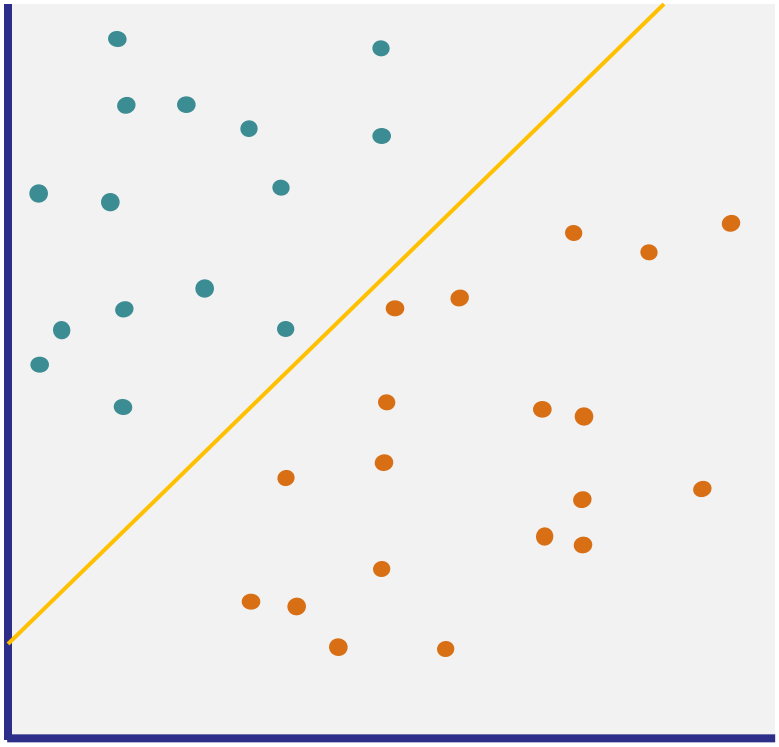


# Linear separation

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

How would you classify these data points?

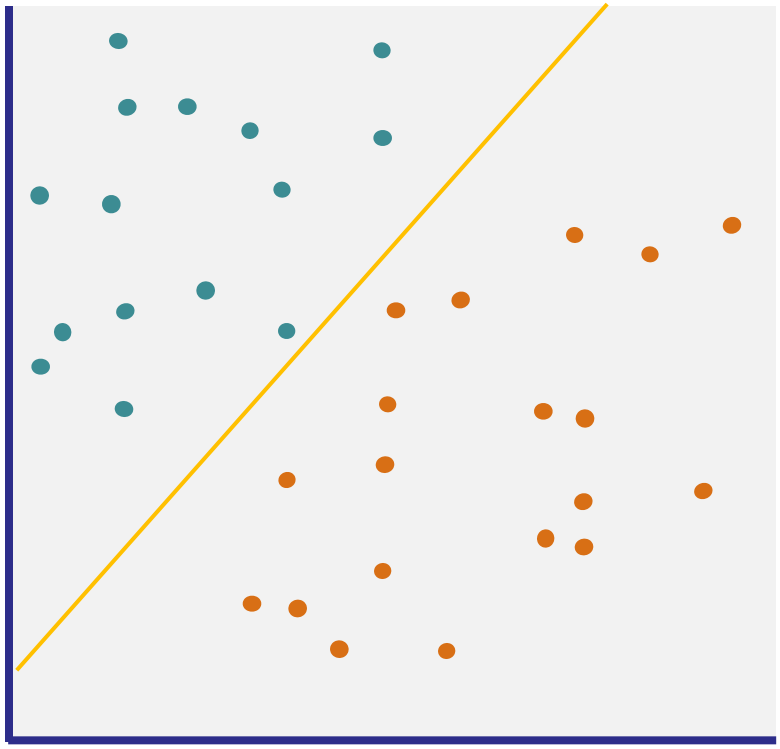


# Linear separation

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

How would you classify these data points?

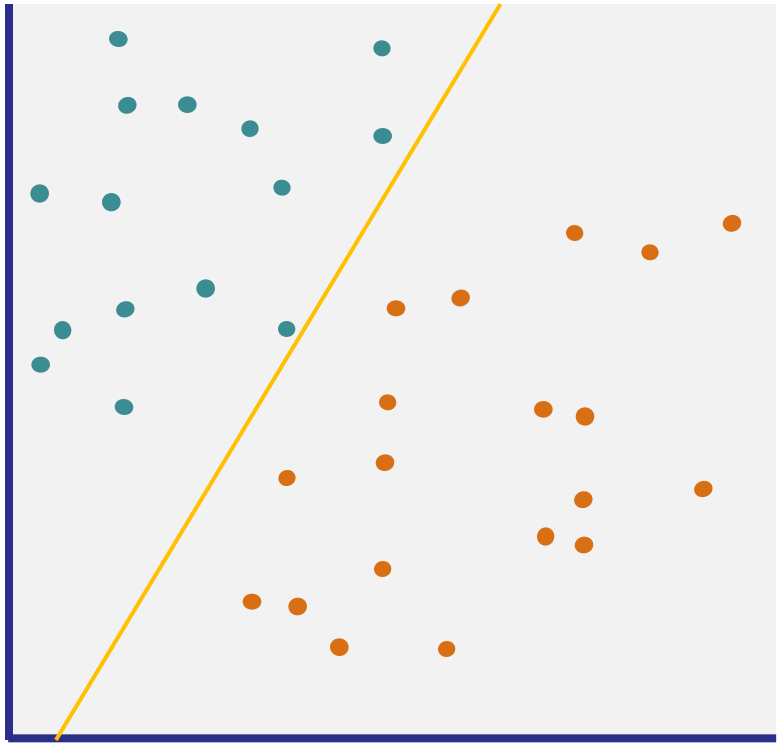


# Linear separation

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

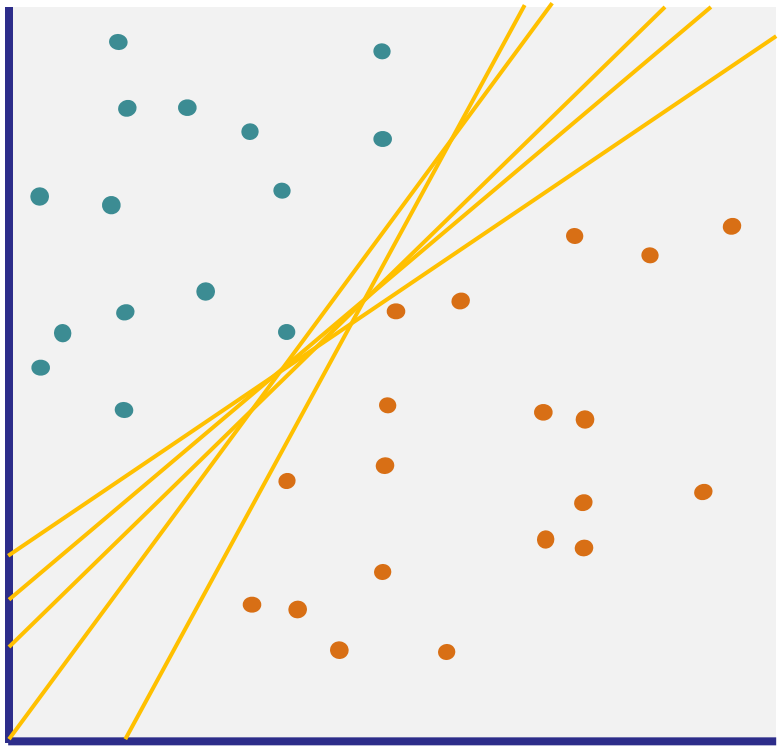
How would you classify these data points?



# Linear separation



Any of these lines would be fine. How do we find the best?

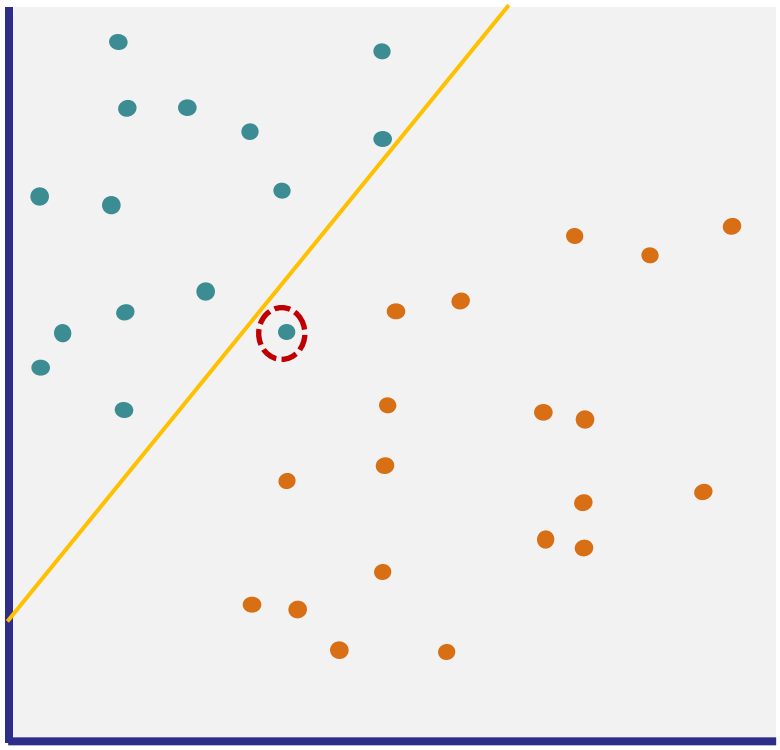


# Linear separation

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

Avoid misclassification

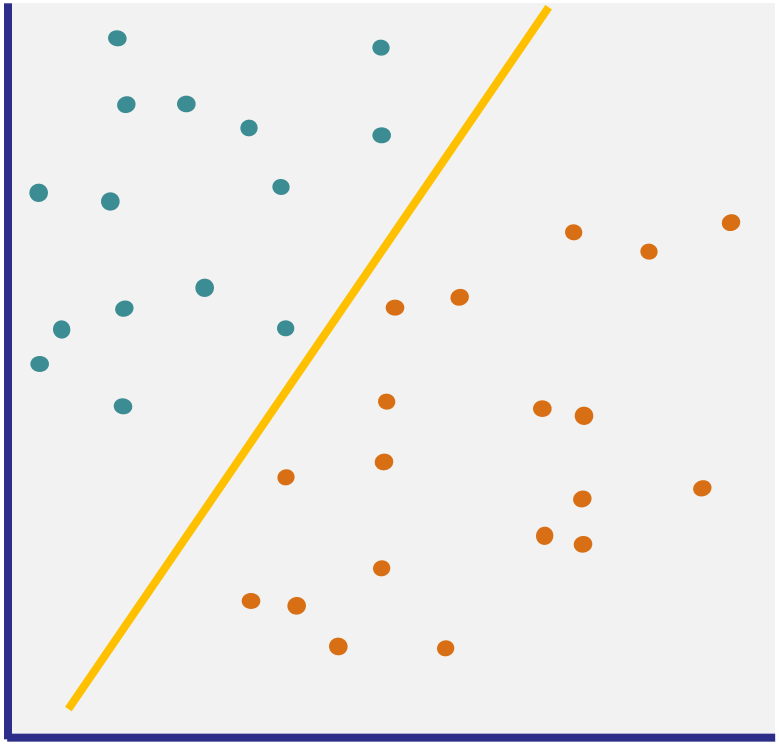


# Classifier margin

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

**Margin of a linear classifier:** the width by which the boundary could be increased before hitting a datapoint



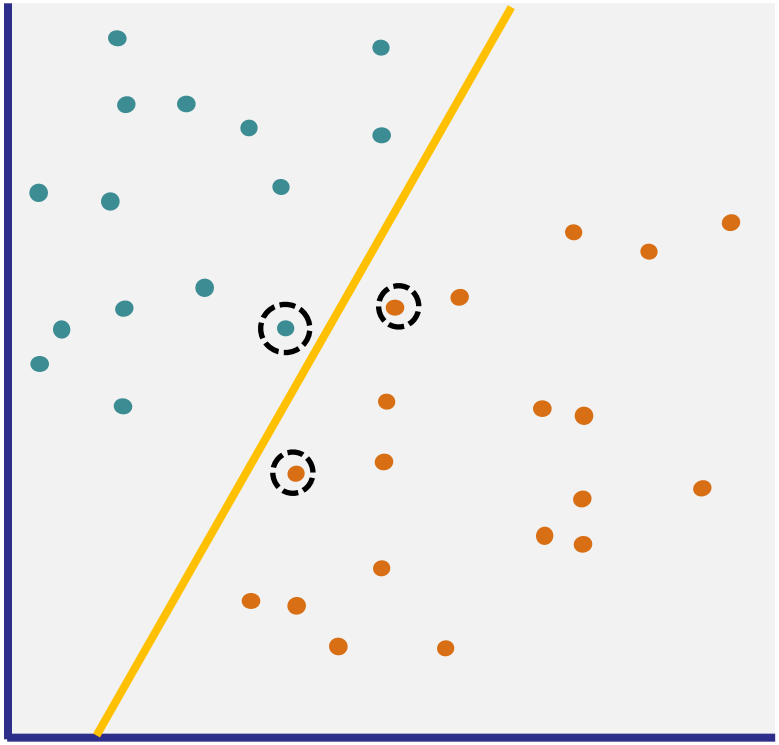
# Support vectors

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

**Goal:** Find the linear classifier with the maximum margin.

**Support vectors:** datapoints which are touched by the margin.



→ simplest form of support vector machines (SVM).

# Formally

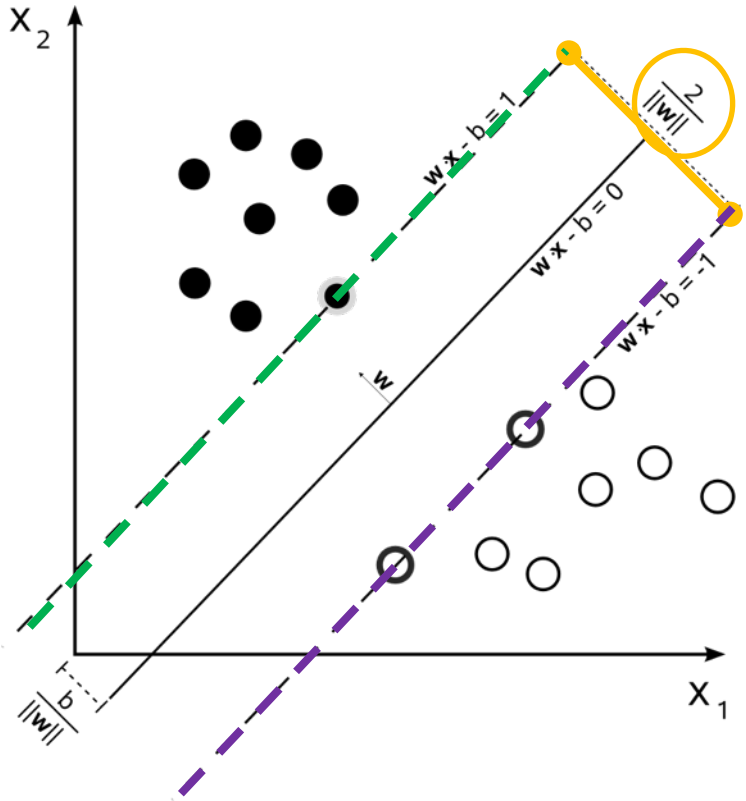
UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

- $\vec{w} \bullet \vec{x} - b = 0$ defines a hyperplane, w is the normal vector of the hyperplane
- left boundary: $\vec{w} \bullet \vec{x} - b = 1$
- right boundary: $\vec{w} \bullet \vec{x} - b = -1$

1. maximize the margin
   = the distance between left and right boundary
   $= \frac{2}{\|\vec{w}\|}$
   →minimize $\vec{w}$

2. determine the best $\vec{w}$ and $b$, such that
   for all points $\vec{x}_i$ it holds that
   - $\vec{w} \bullet \vec{x}_i - b \geq 1, \; if \; y_i = class \; 1$
   - $\vec{w} \bullet \vec{x}_i - b \leq -1, \; if \; y_i = class \; 2$



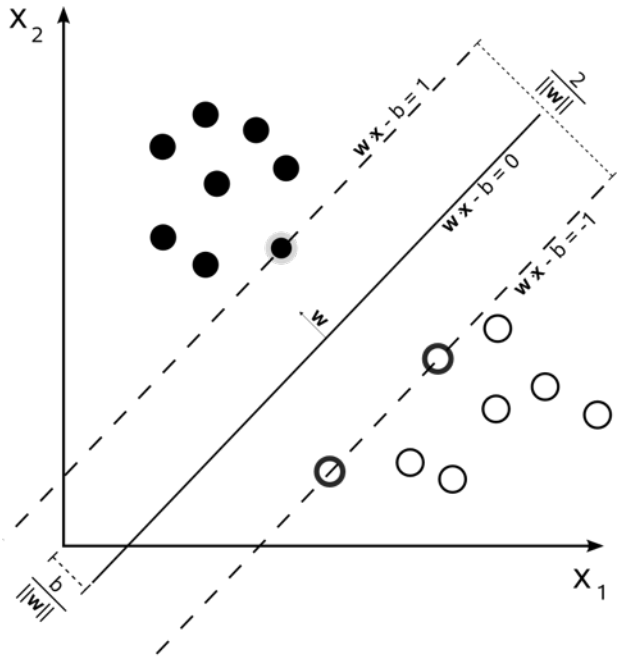

https://en.wikipedia.org/wiki/Support_vector_machine#/media/File:Svm_max_sep_hyperplane_with_margin.png

# Maximum margin

UNIVERSITÄT DUISBURG ESSEN

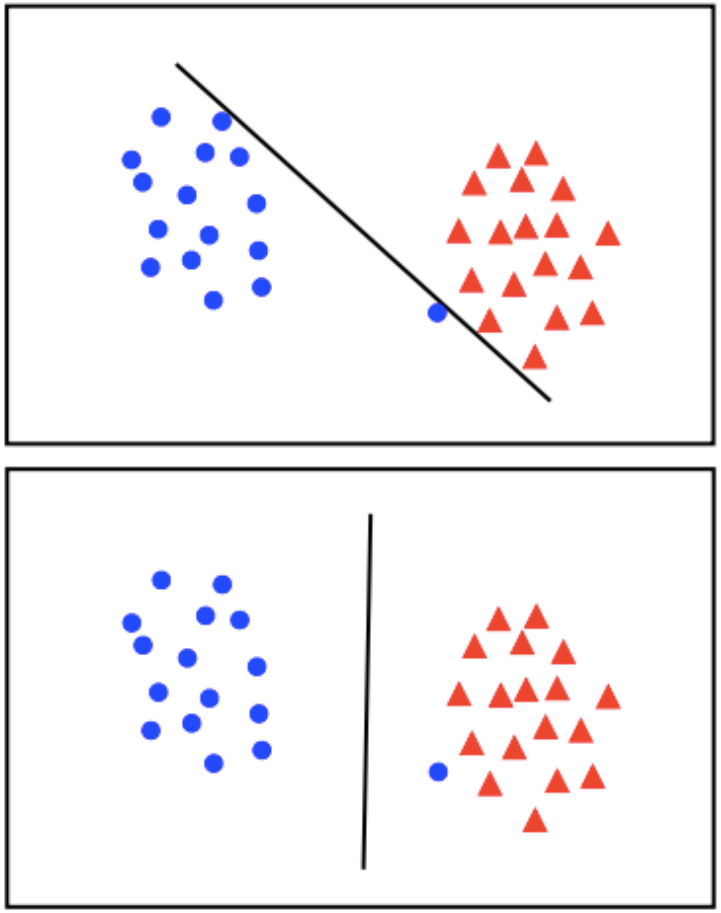
*Offen* im Denken

- Maximizing the margin is an intuitive approach.
- Consequence: only support vectors (the borderline cases) are important; other training examples can be skipped.
- Empirically it often works very well.



# Handling Noise





http://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

- Which solution is better?
- **Hard Margin:**
  All data points should be classified correctly → No training error
- What if the training set is noisy?

# Cut me some slack!

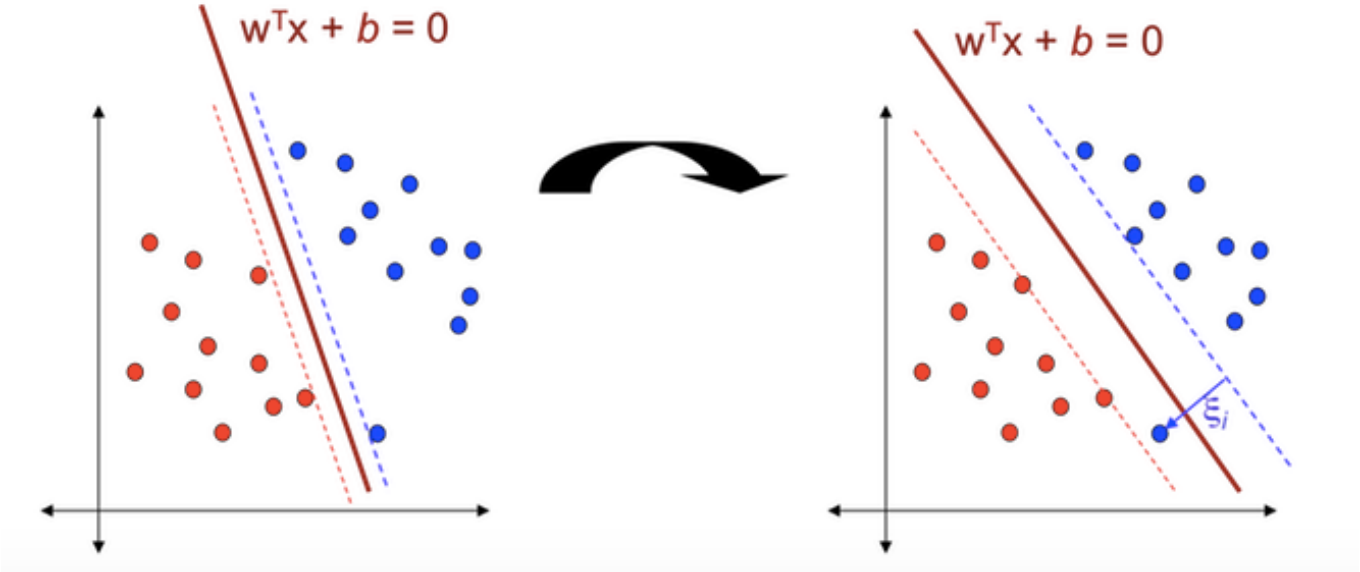
UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

**Soft margin classification:**

◊ Slack variables $\xi_i$ allow misclassification of difficult or noisy examples.

→ A trade-off between the margin size and the number of mistakes on the training data.

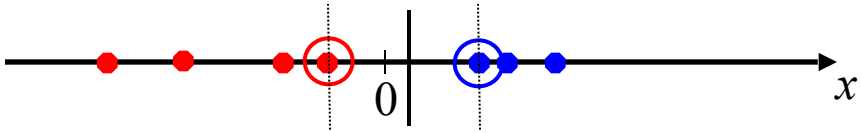


# Non-linear SVMs

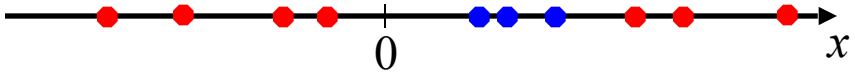
UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

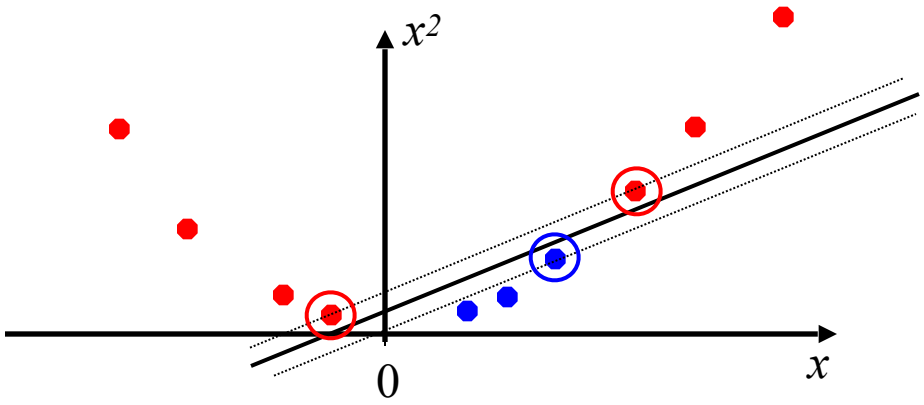
Datasets that are linearly separable with some noise work out great:




But what are we going to do if the dataset is just too hard?




How about… mapping data to a higher-dimensional space:



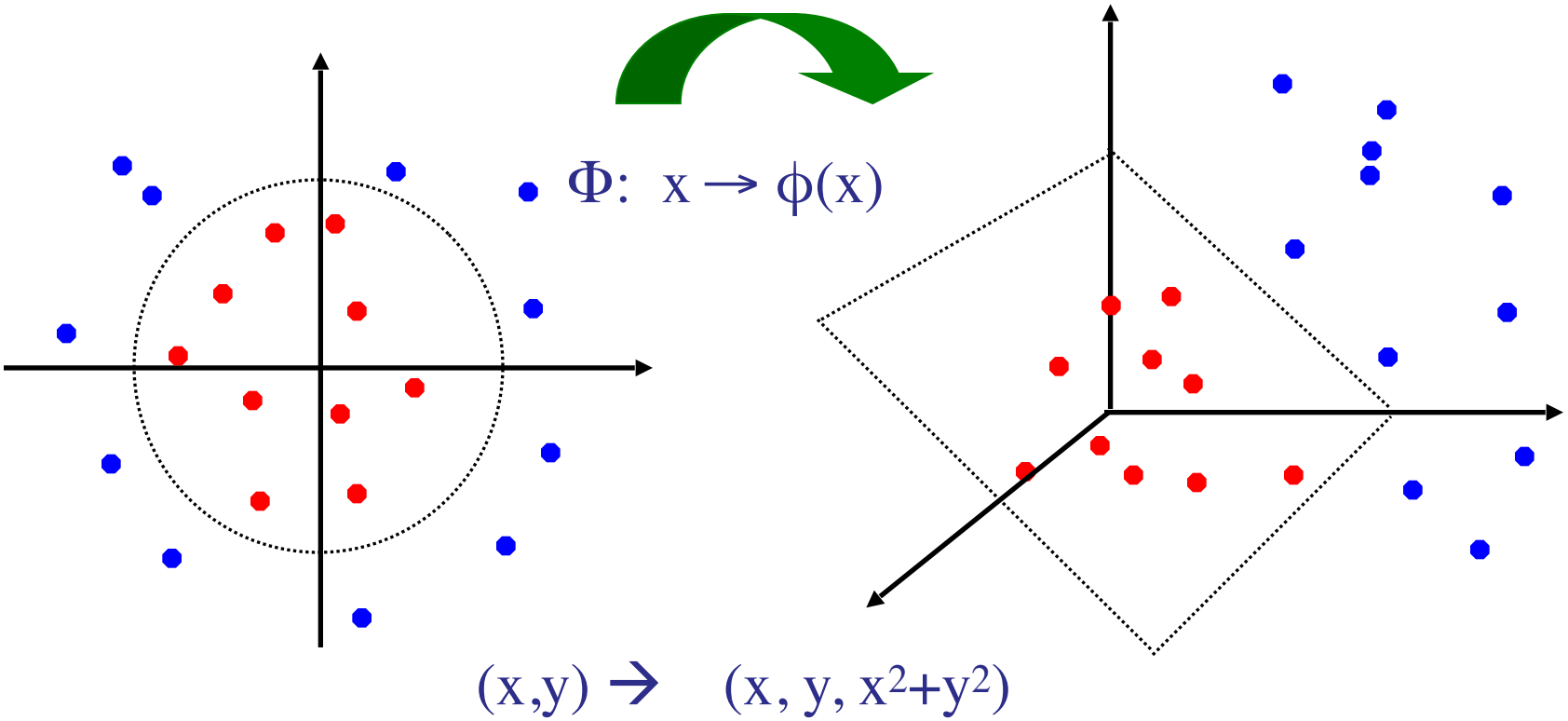

$\Phi: x \rightarrow (x, x^2)$

# Non-linear SVMs

UNIVERSITÄT DUISBURG ESSEN

*Offen im Denken*

**Idea:** the original input space can be mapped to some higher-dimensional feature space where the training set is separable.



# Multi-class classification with SVMs

UNIVERSITÄT DUISBURG ESSEN

*Offen im Denken*

SVMs can only separate two classes. What if we have more than two?

**One vs. all:** with m classes, learn m SVMs

- SVM 1 learns "Output==1" vs "Output != 1"
- SVM 2 learns "Output==2" vs "Output != 2"
- ...
- SVM m learns "Output==m" vs "Output != m"

→use prediction with largest distance to the decision plane

**One vs. one:** train $\frac{m * (m-1)}{2}$ "one vs. one" classifiers

- SVM 1 learns "Output==1" vs "Output == 2"
- SVM 2 learns "Output==1" vs "Output == 3"
- ...

→ choose the class that is predicted most often

**Expensive, but often remarkably fast as one vs. one decision might be easier and uses less instances.**

# SVM advantages and limitations

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

## Advantages

- deals well with high-dimensional, sparse vectors
- **very flexible:** different kernel functions, number of support vectors
- robust classifiers, noise-tolerant
- **overfitting** can be controlled by **soft margin approach**

## Limitations

- only two classes: must use **1 vs. all** or other schemes for multiclass problems
- high **computational complexity**
  - but **efficient implementations available**
- **choice of kernel function and its parameters is a trial-and-error** enterprise

# Summary



KNN, linear regression and decision trees

- are easy to understand and produce interpretable output.
- struggle with high-dimensional input.
- are prone to overfitting.

Naïve Bayes

- is a fast and often effective algorithm.
- can deal with mixed feature types.
- struggles with correlated features (independence assumption).

SVMs

- can deal with high-dimensional feature spaces.
- have many parameters to tune.

# Reading

UNIVERSITÄT DUISBURG ESSEN

*Offen* im Denken

Mandatory

- https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/
  → advertised as a “7 min read” ☺

Optional

- any good machine learning introduction