Maschinelle Sprachverarbeitung

Text Classification

Ulf Leser

Content of this Lecture

- Classification
  - Approach, evaluation and overfitting
  - Examples
- Algorithms
- Case studies

Disclaimer

- This is not a course on Machine Learning
- Classification/clustering are presented from an application point-of-view
  - There exit more methods, much work on empirical comparisons, and a lot of work on analytically explaining the differences
- Experience: Choosing another classification / clustering method typically will not lead to dramatic improvements
  - Problems are either well classifiable or not
  - Also simple methods find the most discriminating properties
- More important: Choice of features
  - Requires creativity and must be adapted to every problem
  - We do not discuss feature selection

Text Classification

- Given a set D of docs and a set of classes C. A classifier is a function f: D→C
- How does this work in general (supervised learning)?
  - Function v mapping a doc into vector of features (feature space)
    - For instance, its bag-of-words, possibly weighted by TF*IDF
  - Obtain a set S of docs with their classes (training data)
  - Find the characteristics of the docs in each class (model)
    - Which feature values / ranges are characteristic?
    - What combinations or properties are characteristic?
  - Encode the model in a classifier function f operating on the feature vector: v: D→V, and f: V→C
  - Classification: Compute f(v(d))

Applications of Text Classification

- Language identification
- Topic identification
- Spam detection
- Content-based message routing
- Named entity recognition (is this token part of a NE?)
- Relationship extraction (does this pair of NE have the relationship we search for?)
- Author identification (which plays were really written by Shakespeare?)
- ...

Good Classifiers

- Problem: Finding a good classifier
  - Assigning as many docs as possible to their correct class
  - Involves finding a proper feature space
- How do we know?
  - Use a (separate) gold standard data set
  - Use training data twice (beware of overfitting)
    - Learning the model
    - Evaluating the model
  - f is the better, the more docs it assigns to their correct classes

Overfitting

- Let S be a set of instances 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 "~"
  - See kNN classifiers
- 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
- Still, often we only have S for evaluation ...
  - We need to extrapolate the quality of f to unknown docs
- Usual method: Cross-validation (leave-one-out, jack-knife)
  - Divide S into k disjoint partitions (typical: k=10)
    - Leave-one-out: k=|S|
  - 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

Problem 1: Information Leakage

- Developing a classifier is an iterative process
  - Define feature space
  - Evaluate performance using cross-validation
  - Perform error analysis, leading to others features
  - Iterate until satisfied with result
- 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

Problem 2: Biased S

- Very often, S is biased. Classical example:
  - Often, one class c' (or some classes) is much less frequent than the other(s)
    - E.g. finding text written in dialect
  - To have enough instances of c' in S, these are searched in D
  - Later, examples from other classes are added
  - But how many?
  - Fraction of c' in S is much (?) higher than in D
    - I.e., than obtained by random sampling
- Solutions
  - Try to estimate fraction of c' in D and produce stratified S
  - Very difficult and costly, often almost impossible
    - Because S would need to be very large

Content of this Lecture

- Classification
  - Approach, evaluation and overfitting
  - Examples
- Algorithms
- Case studies

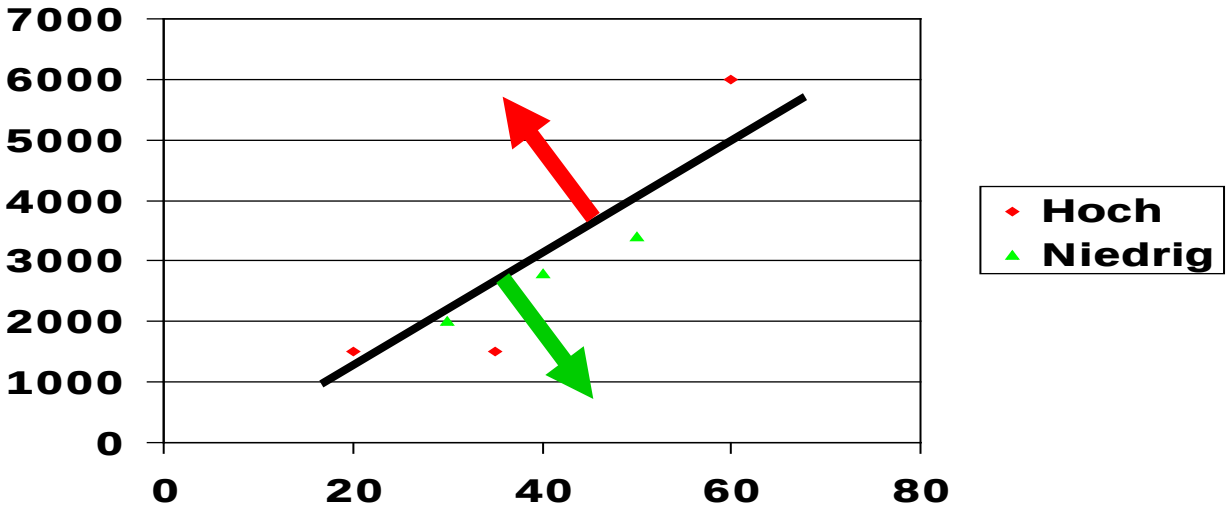
A Simple Example

- Aggregated history of credit loss in a bank

ID	Age	Income	Risk
1	20	1500	High
2	30	2000	Low
3	35	1500	High
4	40	2800	Low
5	50	3000	Low
6	60	6000	High

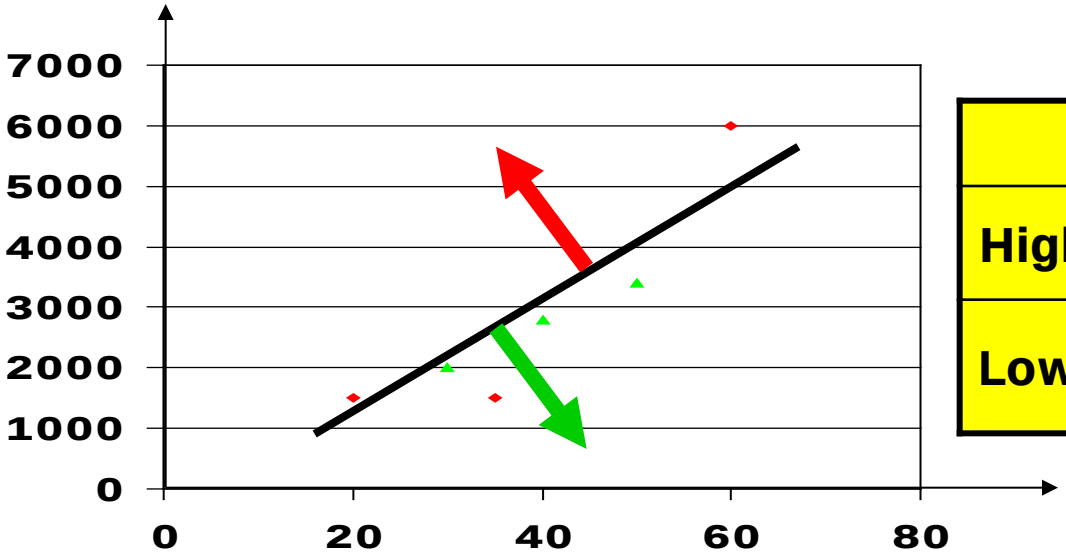
- Now we see a new person, 45 years old, 4000 Euro income
- What is his/her risk?

Regression



- Simple approach: Linear separation by line achieving the minimum squared error (regression)
- Use location relative to regression line as classifier
  - Bad method – regression does not take classes into account
  - But there are classifier based on regression

Performance on the Training Data



	High	Low
High	2	0
Low	1	3

- Quality of predicting “high risk”
  - Precision = 2/2, Recall = 2/3, Accuracy = 5/6
- Assumptions: Linear problems, numerical attributes

Categorical Attributes

ID	Age	Type of car	Risk of Accident
1	23	Family	High
2	17	Sports	High
3	43	Sports	High
4	68	Family	Low
5	25	Truck	Low

- Assume this is analyzed by an insurance agent
- What will he/she infer?
  - Probably a set of rules, such as

```
if      age > 50      then risk = low
elseif  age < 25      then risk = high
elseif  car = sports  then risk = high
else    risk = low
```

Decision Rules

ID	Age	Type of car	Risk of Accident
1	23	Family	High
2	17	Sports	High
3	43	Sports	High
4	68	Family	Low
5	25	Truck	Low

- Can we find less rules which, for this data set, result in the same classification quality?

```
if      age > 50      then risk = low
elseif  car = truck   then risk = low
else    risk = high
```

A Third Approach

ID	Age	Type of car	Risk of Accident
1	23	Family	High
2	17	Sports	High
3	43	Sports	High
4	68	Family	Low
5	25	Truck	Low

- Why not:

```
If      age=23 and car = family then risk = high
elseif  age=17 and car = sports then risk = high
elseif  age=43 and car = sports then risk = high
elseif  age=68 and car = family then risk = low
elseif  age=25 and car = truck  then risk = low
else    flip a coin
```

Overfitting - Again

- This was in instance of our “perfect classifier”
- We learn a model from a small sample of the real world
- Overfitting
  - If the model is too close to the training data, it performs perfect on the training data but learned any bias present in the training data
  - Thus, the rules do not generalize well
- Solution
  - Use an appropriate feature set and learning algorithm
  - Evaluate your method using cross-validation

Content of this Lecture

- Classification
- Algorithms
  - Nearest Neighbor
  - Naïve Bayes
  - Maximum Entropy
  - Linear Models and Support Vector Machines (SVM)
- Case studies

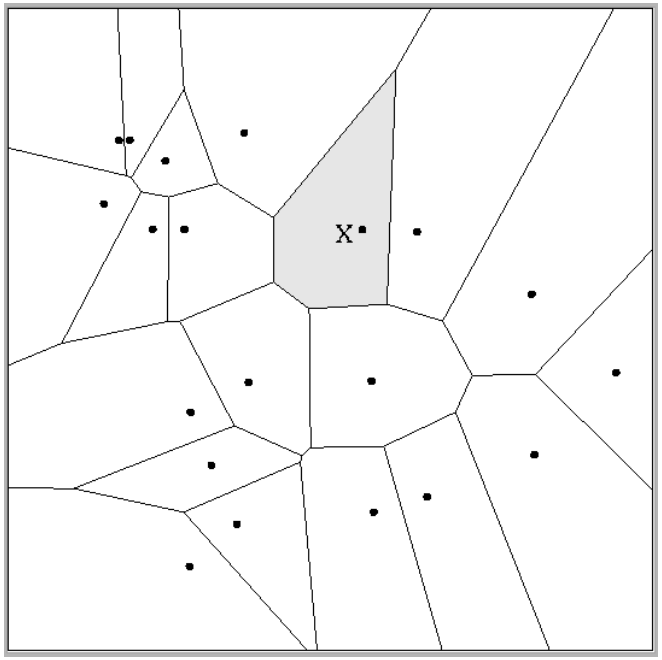
Classification Methods

- There are many different classification methods
  - k-nearest neighbor
  - Naïve Bayes, Bayesian Networks, Graphical models
  - Decision Trees and Rainforests
  - Maximum Entropy
  - Support Vector Machines
  - Perceptrons, Neural Networks
  - ...
- Effectiveness of classification depends on problem, algorithm, feature selection method, sample, evaluation, ...
- Differences when using different methods on the same data/representation are often astonishing small

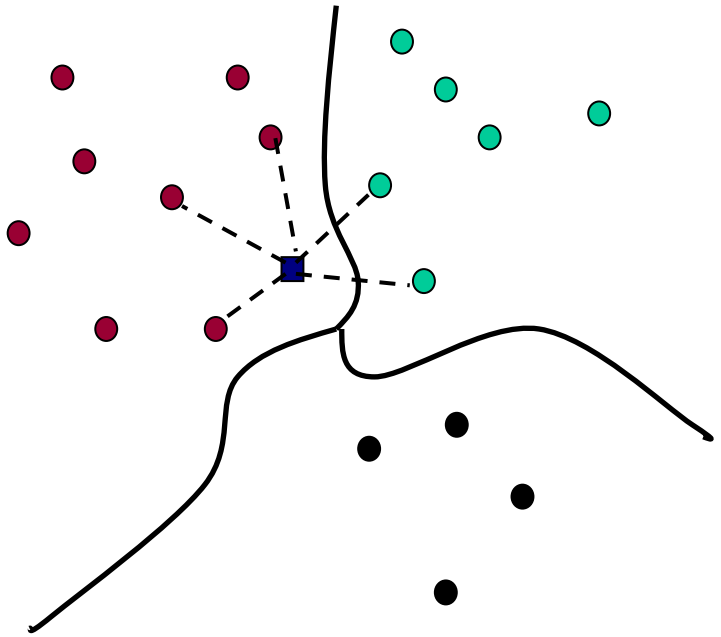
Nearest Neighbor Classifiers

- Definition
  Let S be a set of classified documents, m a distance function between any two documents, and d an unclassified doc.
  - *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*
- Remark
  - 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 have the same class
  - I.e.: The textual content of a doc determines the class
  - Depends a lot on the text representation
  - Depends a lot on the distance function
- kNN in general more robust than NN
- Example of lazy learning
  - Actually, there is no learning
  - Actually, there is no model
- Features often are defined implicitly through the distance function

Disadvantages

- Major problem: Performance (speed)
  - Need to compute the distance between d and all docs in S
  - This requires |S| applications of the distance function
    - Often the cosine of two 100K-dimensional vectors
- Suggestions for speed-up
  - Clustering: Merge groups of close points in S into a single representative
    - Linear speed-up (size of groups)
  - Use multidimensional index structure (see DBS-II)
  - Map into lower-dimensional space such that distances are preserved as good as possible
    - Metric embeddings, dimensionality reduction
    - Not this lecture

kNN for Text

- In the VSM world, kNN is implemented very easily using the tools we already learned
- How?
  - Use cosine distance of bag-of-word vectors as distance
  - The usual VSM query mechanism computes exactly the k nearest neighbors when d is used as query
  - Difference
    - |d|>>|q|: usually has many more keywords than a typical IR-query q
    - We might need other ways of optimizing “queries”

Content of this Lecture

- Classification
- Algorithms
  - Nearest Neighbor
  - Naïve Bayes
  - Maximum Entropy
  - Linear Models and Support Vector Machines (SVM)
- Case studies

Bayes‘ Classification

- Uses frequencies of feature values in the different classes
- Given
  - Set S of docs and set of classes C={c_1, c_2, ... c_m}
  - Docs are described as a set F of discrete features
    - Usually the presence/absence of terms in d
- We seek p(c_i|d), the probability of a doc d∈S being a member of class c_i
- d eventually is assigned to c_i with argmax p(c_i|d)
- Replace d with feature representation

$$p(c \mid d) = p(c \mid v(d)) = p(c \mid f_1[d],..., f_n[d]) = p(c \mid t_1,...,t_n)$$

Probabilities

- What we learn from the training data (MLE)
  - The a-priori probability p(t) of every term t
    - How many docs from S have t?
  - The a-priori probability p(c) of every class c∈C
    - How many docs in S are of class c?
  - The conditional probabilities p(t|c) for term t being true in class c
    - Proportion of docs in c with term t among all docs in c
- Rephrase and use Bayes‘ theorem

$$p(c \mid t_1,...,t_n) = \frac{p(t_1,...,t_n \mid c) * p(c)}{p(t_1,...,t_n)} \approx p(t_1,...,t_n \mid c) * p(c)$$

Naïve Bayes

- We have $p(c \mid d) \approx p(t_1,...,t_n \mid c) * p(c)$
- The first term cannot be learned accurately with any reasonably large training set
  - There are 2^n combinations of (binary) feature values
- „Naïve“ solution: Assume statistical independence of terms
- Then

$$p(t_1,...,t_n \mid c) = p(t_1 \mid c) * ... * p(t_n \mid c)$$

- Finally

$$p(c \mid d) \approx p(c) * \prod_{i=1}^{n} p(t_i \mid c)$$

Properties

- Simple, robust, fast
- Needs smoothing: Avoid any probability to become zero
- Can be extended to ranges of TF*IDF values instead of binary features
  - Requires appropriate binning – more parameter
- Learning is simple, model is compact (O(|K|*|C|) space)
- Often used as baseline for other methods
- When we use the logarithm (produces equal ranking), we see that NB is a log-linear classifier

$$p(c \mid d) \approx \log\left(p(c) * \prod p(t_i \mid c)\right)$$
$$= \log\left(p(c)\right) + \sum \log\left(p(t_i \mid c)\right)$$

Feature Selection

- Good idea: Use only subset of all features
  - Faster, reduction of noise
- Simple method: Use those t where p(t|c) show the biggest differences between the different classes
  - Needs to assess differences; e.g., entropy, information gain, ...
- Numerous methods for feature selection
  - Finding the best features is not the same as finding the best subset of features
  - Overfitting is an issue: "Best features for S" ≠ "best features for D"
- Some methods benefit more than others
  - MaxEnt and SVM usually not much, Bayes usually a lot (think of redundant features)

Content of this Lecture

- Classification
- Algorithms
  - Nearest Neighbor
  - Naïve Bayes
  - Maximum Entropy
  - Linear Models and Support Vector Machines (SVM)
- Case studies

Discriminative versus Generative Models

- Naïve Bayes uses Bayes' Theorem to estimate p(c|d)

$$p(c \mid t_1,...,t_n) = \frac{p(t_1,...,t_n \mid c) * p(c)}{p(t_1,...,t_n)} \approx p(t_1,...,t_n \mid c) * p(c)$$

- Approaches that estimate p(d|c) are called generative
  - p(d|c) is the probability of class c producing data d
  - Naïve Bayes is a generative model
- Approaches that directly estimate p(c|d) are called discriminative
  - But: We only have a very small sample of the document space
  - Many models perform equally well on the training data
  - Generalization is very difficult

Example: Maximum Entropy (ME) Modeling

- Given a set of (binary) features derived from d, it directly learns conditional probabilities p(c|d)
- Since p(c,d)=p(c|d)*p(d) and p(d) is the same for all c, we compute p(c,d)~p(c|d)
- Definition
  *Let s_{ij} be the score of a feature i for doc d_j (such as TF*IDF of a token). We derive from s_{ij} a binary indicator function f_i*

$$f_i(d_j, c) = \begin{cases} 1, & if\ s_{ij} > 0 \wedge c = c(d_j) \\ 0 & otherwise \end{cases}$$

  - $c(d_j)$: Class of d_j
- Remark
  - We will often call those indicator functions “features”, although they embed information about classes (“a feature in a class”)

Classification with ME

- The ME approach models the joint probability p(c,d) as

$$p(c,d) = \frac{1}{Z} * \prod_{i=1}^{K} \alpha_{i,c}^{f_i(d)}$$

  - Z is a normalization constant to turn the scores into probabilities
  - The feature weights α_i are learned from the data
  - K is the number of features
  - This particular function is determined by optimization algorithm
- Application: Compute p(c,d) for all c and return best class

Finding Feature Weights

- Problem: Learning optimal feature weights α_i
- Choose α_i such that probability of S given M is maximal

$$p(S \mid M) = \sum_{d \in S} p(c(d), d \mid M)$$

- Choice should consider dependencies between features
- Recall Naïve Bayes
  - Computes α-like values independently for each feature (rel freq)
  - Uses log-linear combination for classification
  - This only works well if statistical independence holds
  - For instance, using the same feature multiple times does influence a NB result

Maximum Entropy Principle

- Problem: There are usually many combinations of weights that may all give rise to the same maximal probability of S
- ME chooses the model with the largest entropy
  - ME tries to make as few assumptions as possible given the data
  - Abstract formulation: The training data leaves too much freedom. We want to choose M such that all “undetermined” probability mass is distributed equally
  - Such a distribution exists and is unique
  - Optimization needs to take this into account

Entropy of a Distribution

- Let F be a feature space and M be an assignment of probabilities to each feature s in F. The entropy of the probability distribution M is defined as

$$h(M) = -\sum_{s \in F} p(s \mid M) * \log(p(s \mid M))$$

- Search M such that P(S|M) is maximal and h(M) is maximal

Example [NLTK, see http://nltk.googlecode.com/svn/trunk/doc/book/ch06.html]

- Assume we have 10 different classes A-J and no further knowledge. We want to classify a document d. Which probabilities should we assign to the classes?

	A	B	C	D	E	F	G	H	I	J
(i)	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
(ii)	5%	15%	0%	30%	0%	8%	12%	0%	6%	24%
(iii)	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%

- Model (i) does not model more than we know
- Model (i) also has maximal entropy

Example continued

- We learn that A is true in 55% of all cases. Which model do you chose?

	A	B	C	D	E	F	G	H	I	J
(iv)	55%	45%	0%	0%	0%	0%	0%	0%	0%	0%
(v)	55%	5%	5%	5%	5%	5%	5%	5%	5%	5%
(vi)	55%	3%	1%	2%	9%	5%	0%	25%	0%	0%

- Model (v) also has maximal entropy under all models that incorporate the knowledge about A

Example continued

- We additionally learn that if the word “up” appears in a document, then there is an 80% chance that A or C are true. Furthermore, “up” is contained in 10% of the docs.
- This would result in the following model
  - We need to introduce features
  - The 55% a-priori chance for A still holds

	A	B	C	D	E	F	G	H	I	J
+up	5.1%	0.25%	2.9%	0.25%	0.25%	0.25%	0.25%	0.25%	0.25%	0.25%
-up	49.9%	4.46%	4.46%	4.46%	4.46%	4.46%	4.46%	4.46%	4.46%	4.46%

- Things get complicated if we have >100k features

Example 2 [Pix, Stockschläder, WS07/08]

- Assume we count occurrences of “has blue eyes” and “is left-handed” among a population of tamarins
- We observe p(eye)=1/3 and p(left)=1/3
- What is the joint probability p(eye, left) of blue-eyed, left-handed tamarins?
  - We don’t know
  - It must be 0≤p(eye,blue)≤min(p(eye),p(left))=1/3
- Four cases

p(...,...)	left-handed	not left-handed	sum
blue-eyed	x	1/3-x	1/3
not blue-eyed	1/3-x	1-2/3+x	2/3
sum	1/3	2/3	1

Maximizing Entropy

- The entropy of the joint distribution M is

$$h(M) = -\sum_{i=1}^{4} p(x,y) * \log(p(x,y))$$

- The value is maximal for dH/dx = 0
- Computing the first derivative and solving the equation leads to x=1/9
  - Which, in this case, is the same as assuming independence, but this is not generally the case
- In general, finding a solution in this analytical way is not possible

Generalized Iterative Scaling (idea)

- No analytical solution to the general optimization problem exists
- Generalized Iterative Scaling to find optimal α_i
  - Iterative procedure finding the optimal solution
  - Start from a random guess of all weights and iteratively redistribute probability mass until convergence to a optimum for p(S|M) under h(M) constraint
  - See [MS99] for the algorithm
- Problem: Usually converges very slowly
- Several improvements are known
  - Improved Iterative Scaling
  - Conjugate Gradient Descent

Properties of Maximum Entropy Classifiers

- In general, ME outperforms NB
- ME does not assume independence of features
  - Learning of feature weights always considers entire distribution
  - Two highly correlated features will get only half of the weight as if there was only one feature
- Very popular in statistical NLP
  - Some of the best POS-tagger are ME-based
  - Some of the best NER systems are ME-based
- Several extensions
  - Maximum Entropy Markov Models
  - Conditional Random Fields

Content of this Lecture

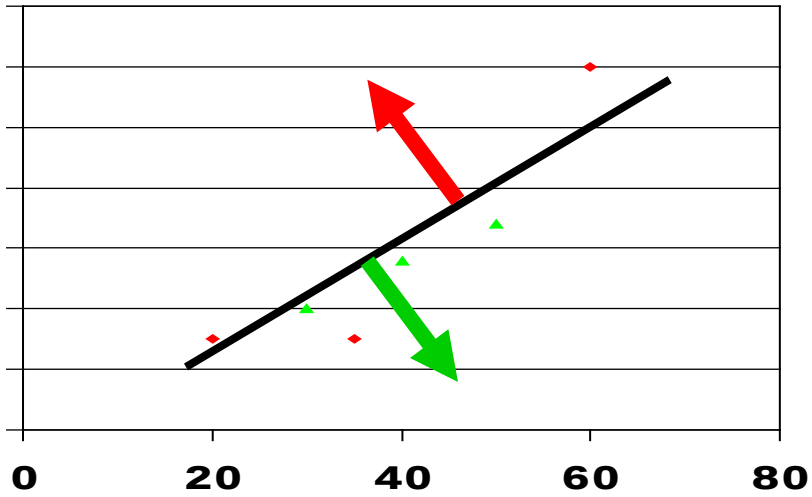
- Classification
- Algorithms
  - Nearest Neighbor
  - Naïve Bayes
  - Maximum Entropy
  - Support Vector Machines (SVM)
- Case studies

Class of 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?
- Experience: Classifiers more powerful than linear often don't perform better (on text)

NB and Regression

- Regression computes a separating hyperplane using error minimization



- If we assume binary Naïve Bayes, we may compute

$$p(c \mid d) \approx \log\big(p(c)\big) + \sum \log\big(p(t_i \mid c)\big)$$
$$= a + \sum b_i * TF_i$$

Linear hyperplane; value>0 gives c, value<0 gives ¬c

ME is a Log-Linear Model

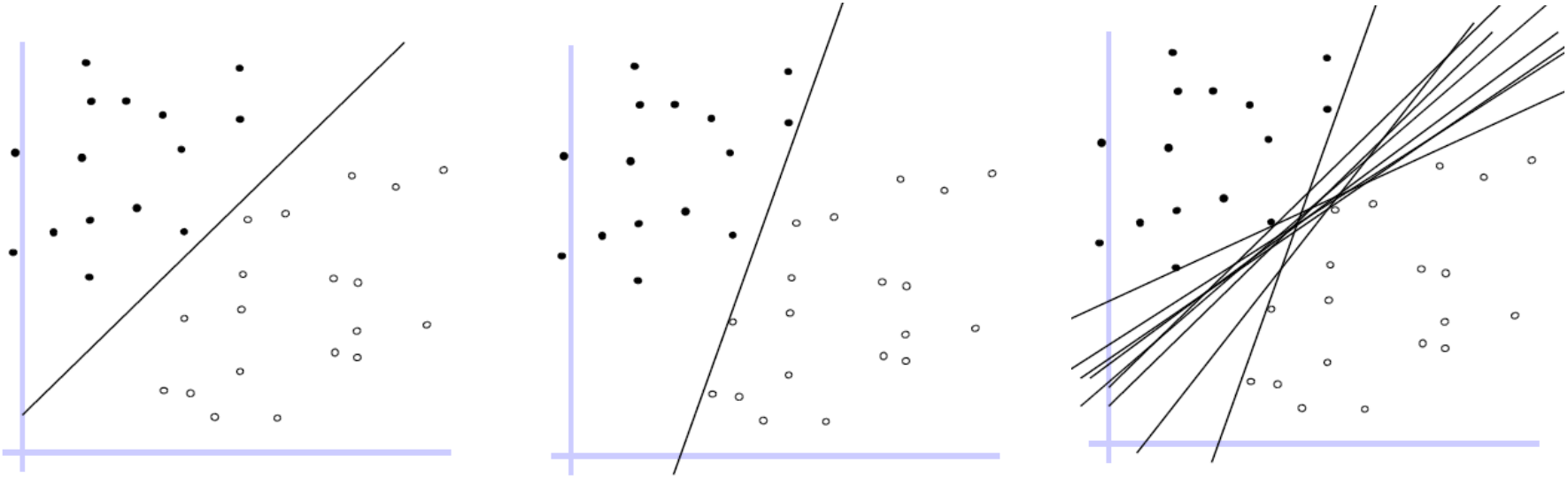
$$p(c,d) = \frac{1}{Z} * \prod_{i=1}^{K} \alpha_i^{f_i(d,c)} \approx \log\left(\frac{1}{Z}\right) + \sum_{i=1}^{K} f_i(d,c) * \alpha_i$$

Text = High Dimensional Data

- High dimensionality: 100k+ features
- Sparsity: Feature values are almost all zero
- Most document pairs 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

Linear Classifiers (2D)

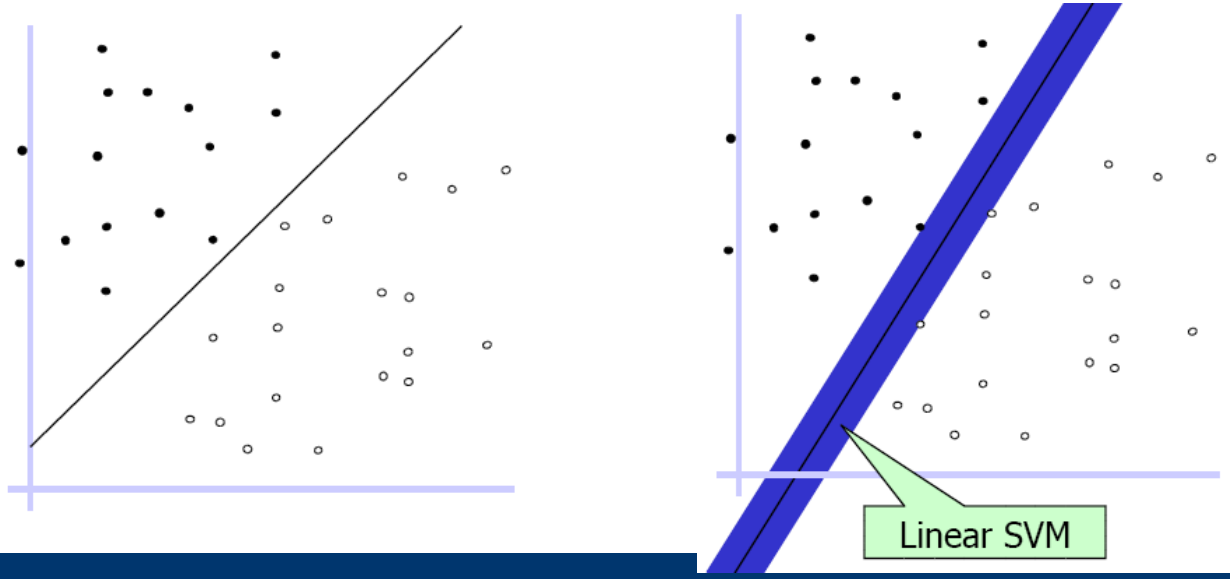
- Hyperplane separating classes in high dimensional space
- But which?



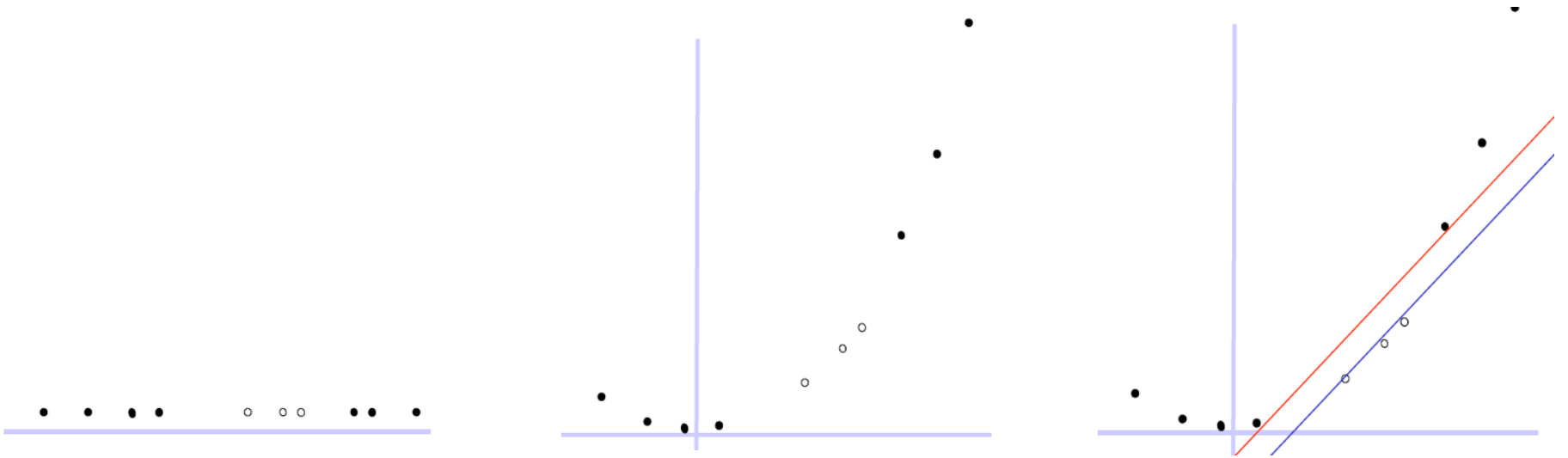
Quelle: Xiaojin Zhu, SVM-cs540

Support Vector Machines (sketch)

- 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
  - Minimizes a particular type of error



Kernel Trick: Problems not Linearly Separable



- Map data into an even higher dimensional space
- Not-linearly separable sets may become linearly separable
- Doing this efficiently requires a good deal of work
  - The "kernel trick"

Properties of SVM

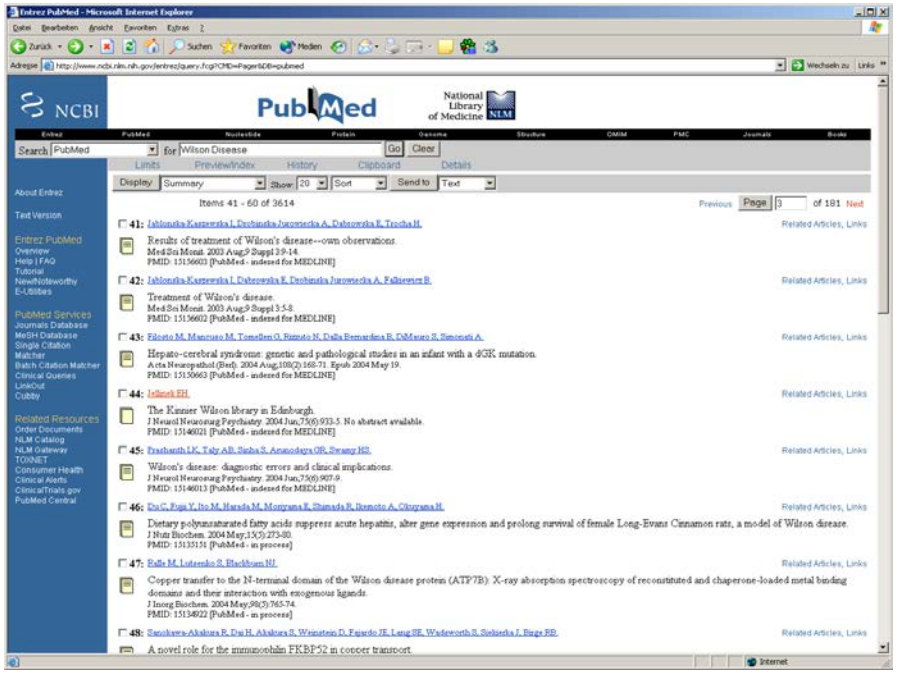
- State-of-the-art in text classification
- Often requires long training time
- Classification is rather fast
  - Only distance to hyperplane is needed
  - Hyperplane is defined by only few vectors (support vectors)
- SVM are quite good “as is”, but tuning possible
  - Kernel function, biased margins, ...
- Several free implementations exist: SVMlight, libSVM, ...

Content of this Lecture

- Classification
- Algorithms
- Case studies
  - Topic classification
  - Spam filtering

Topic Classification [Rutsch et al., 2005]

- Find publications treating the molecular basis of hereditary diseases
- Pure key word search generates too many results
  - "Asthma": 84 884 hits
    - Asthma and cats, factors inducing asthma, treatment, ...
  - "Wilson disease": 4552 hits
    - Including all publications from doctors named Wilson
- Pure key word search does not cope with synonyms



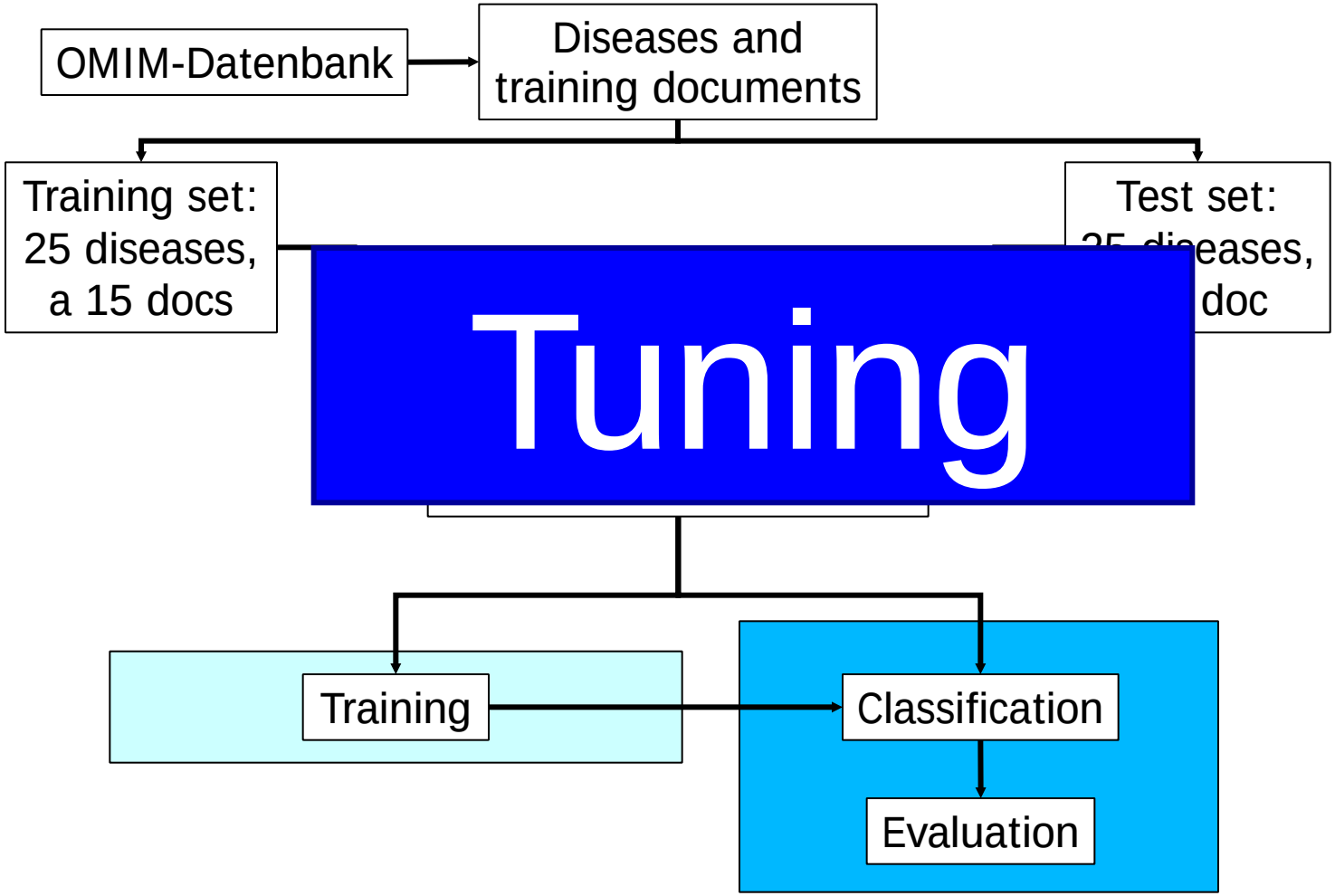
Idea

- Learn what is typical for a paper treating molecular basis of diseases from examples
  - 25 hereditary diseases
  - 20 abstracts for each disease
- We call this “typical” a model of the data
- Models are learned using some method

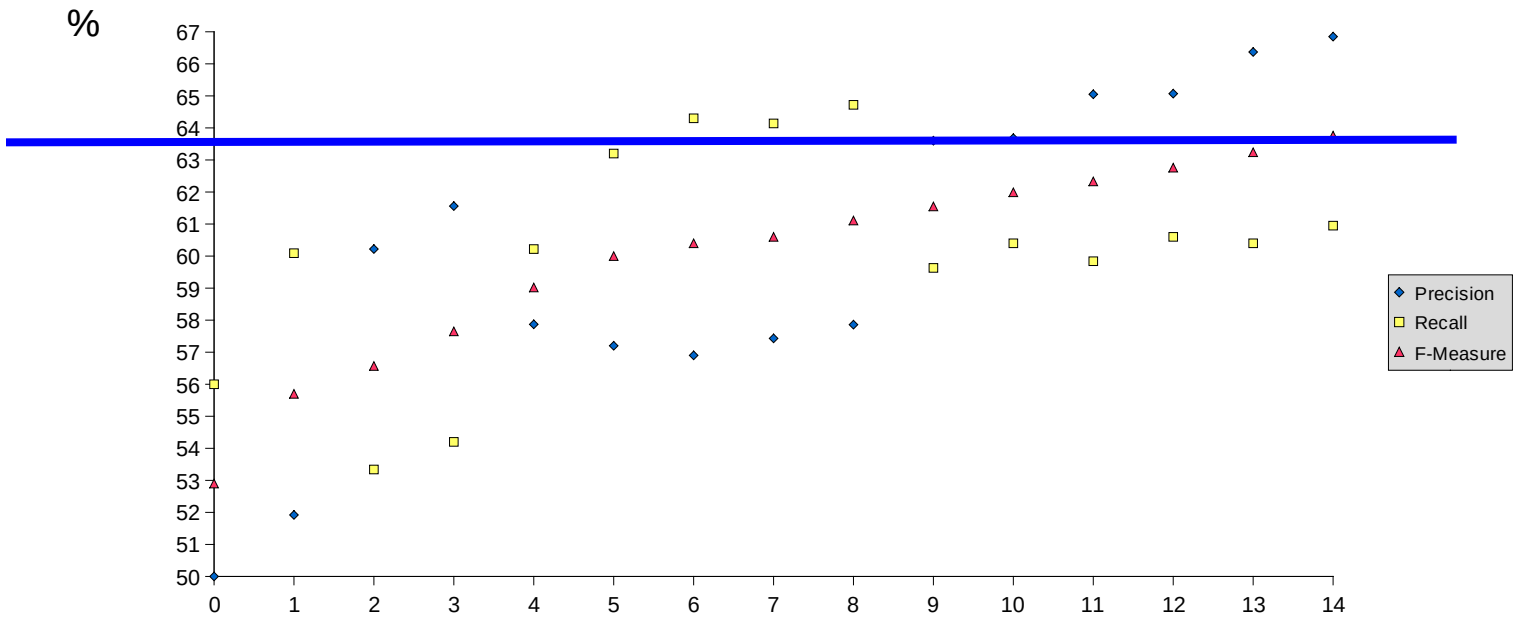
- Classification: Given a new text, find the model which fits best and predict the associated class (disease)

- What can we learn from 20 documents?

Complete Workflow

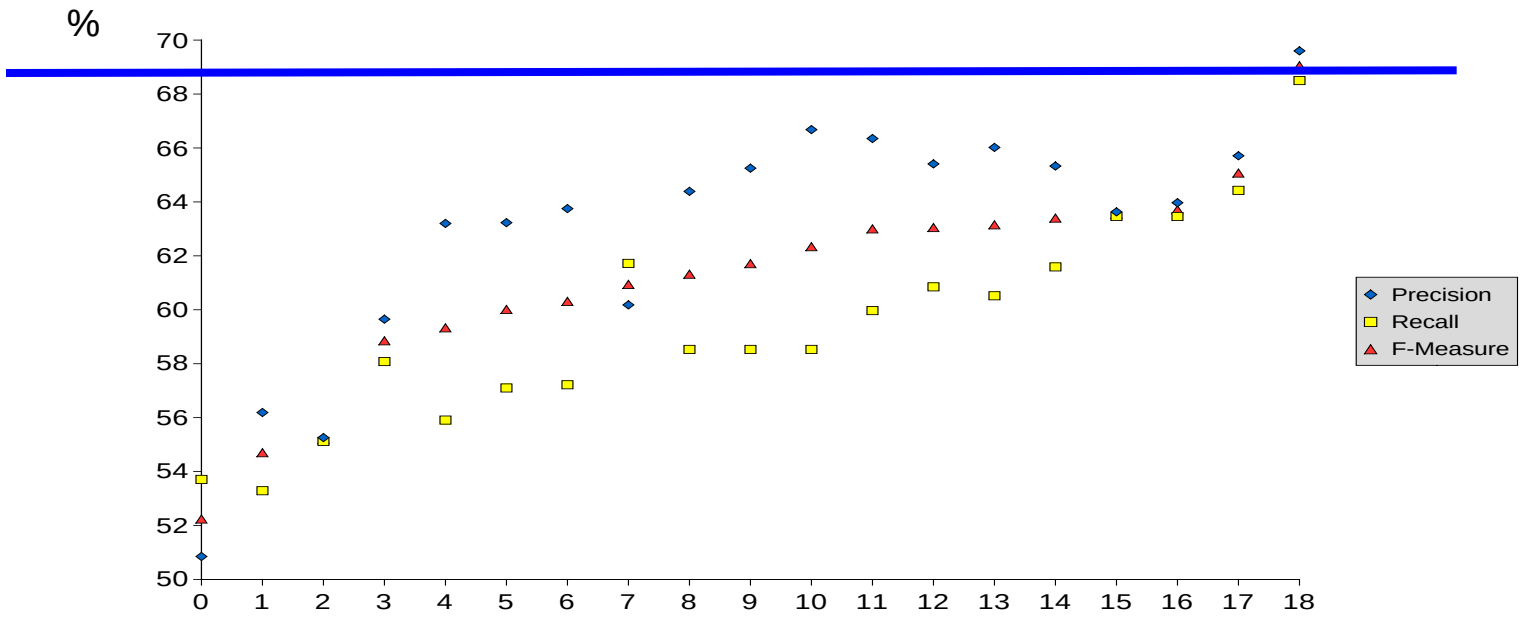


Results (Nearest-Centroid Classifier)



- Configurations (y-axis)
  - Stemming: yes/no
  - Stop words: 0, 100, 1000, 10000
  - Different forms of tokenization
- Best: No stemming, 10.000 stop words

Results with Section Weighting



- Use different weights for terms depending on the section they appear in
  - Introduction, results, material and methods, discussion, ...

Influence of Stemming

Mit stemmer			
Nomen und Verben			
	100	1000	10000
Precision	61,00	63,07	**67,42**
Recall	59,29	60,51	**65,01**
F-Measure	60,13	61,76	**66,19**

Ohne Stemmer			
Nomen und Verben			
	100	1000	10000
Precision	62,90	64,94	**66,17**
Recall	62,59	62,38	**62,71**
F-Measure	62,75	63,63	**64,39**

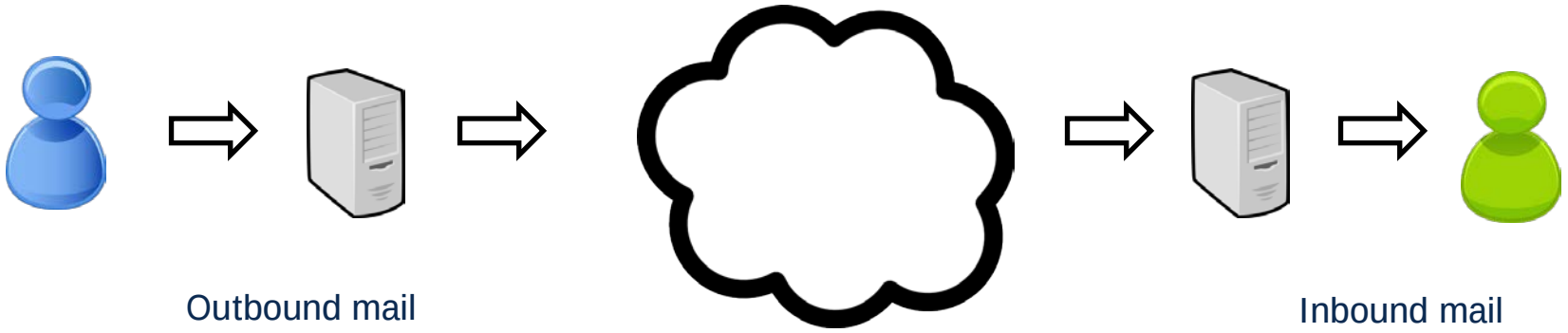
Content of this Lecture

- Classification
- Algorithms
- Case studies
  - Topic classification
  - Spam filtering

Thanks to: Conrad Plake, “Vi@gra and Co.: Approaches to E-Mail Spam Detection”, Dresden, December 2010

Spam

- Spam = Unsolicited bulk email
- Old „problem“: 1978 first spams for advertisement
- Estimate: >95% of all mails are spam
- Many important issues not covered here
  - Filtering at provider, botnets, DNS filtering with black / gray / white lists, using further metadata (attachments, language, embedded images, n# of addressees, ...) etc.
  - Legal issues

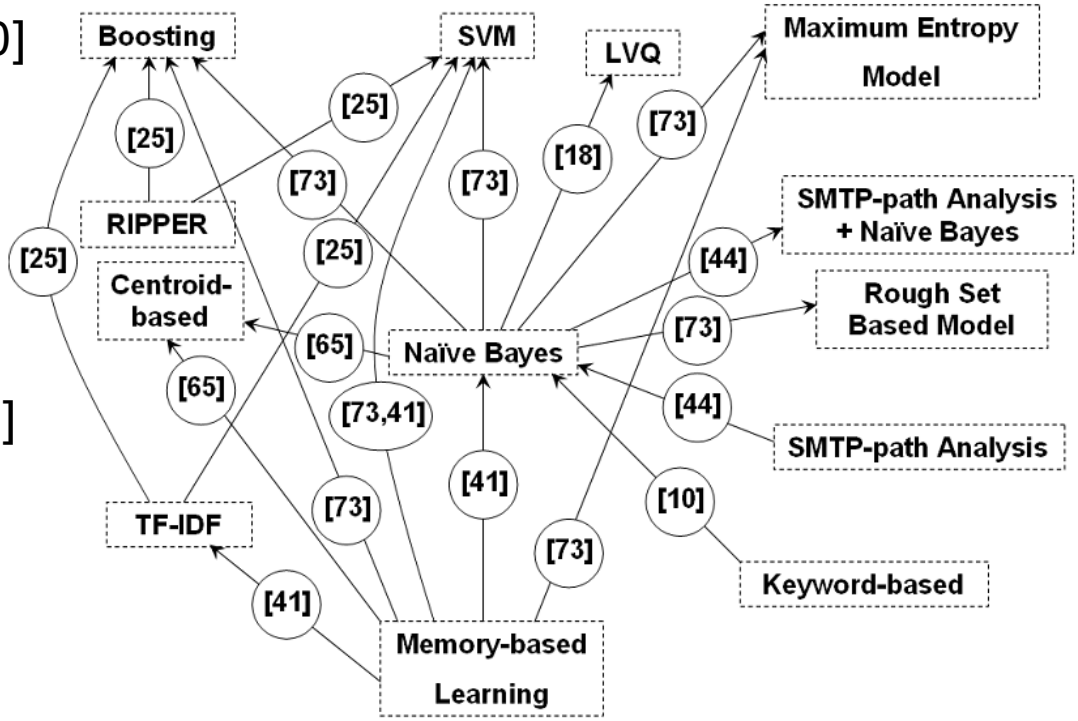


SPAM Detection as a Classification Task

- Content-based SPAM filtering
- Task: Given the body of an email – classify as SPAM or not
- Difficulties
  - Highly unbalanced classes (97% Spam)
  - Spammer react on every new trick – an arms race
  - Topics change over time
- Baseline approach: Naïve Bayes on VSM
  - Implemented in Thunderbird and MS-Outlook
  - Fast learning, iterative learning, relatively fast classification
  - Using TF, TF-IDF, Information Gain, ...
  - Stemming (mixed reports)
  - Stop-Word removal (seems to help)

Many Further Suggestions

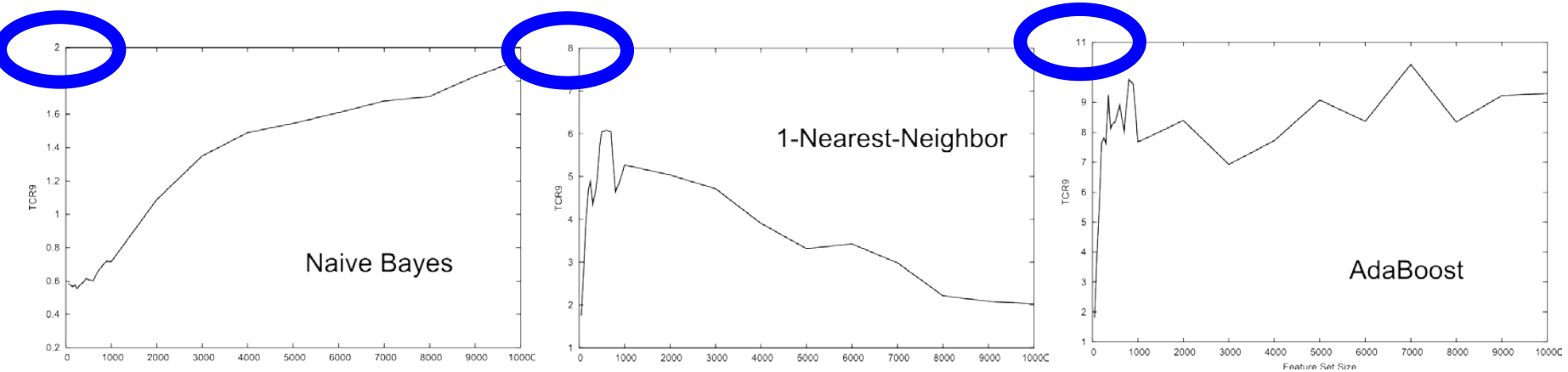
- Rule learning
  [Cohen, 1996]
- k-Nearest-Neighbors
  [Androutsopoulos *et al.*, 2000]
- SVM
  [Kolcz/Alspector, 2001]
- Decision trees
  [Carreras/Marquez, 2001]
- Centroid-based
  [Soonthornphisaj et al., 2002]
- Artificial Neural Networks
  [Clark *et al.*, 2003]
- Logistic regression
  [Goodman/Yih, 2006]
- Maximum Entropy Models
- ...



Source: Blanzieri and Bryl, 2009

Measuring Performance

- We so far always assumed that a FP is as bad as a FN
  - Inherent in F-measure
- Is this true for Spam?
  - Missing a non-spam mail (FP) usually is perceived as much more severe than accidentally reading a spam mail (FN)
- Performance with growing feature sets and c(FP)=9*c(FN)



Problem Solved?

- Tricking a Spam filter
  - False feedback by malicious users (for global filters)
  - Bayesian attack: add “good“ words
  - Change orthography (e.g., viaagra, vi@gra)
  - Tokenization attack (e.g., free -> f r e e)
  - Image spam (already >30%)
- Concept drift
  - Spam topics change over time
  - Filters need to adapt

CEAS 2008 Challenge: Active Learning Task

- CEAS: Conference on Email and Anti-Spam
- Active Learning
  - Systems selected up to 1000 mails
  - Selection using score with pre-learned model
  - Classes of these were given
  - Simulates a system which asks a user if uncertain
- 143,000 mails

Name	Spam Caught %	Blocked Ham %	1-AUC %
Logistic Regression + Active Learning	99.92	0.12	0.0033
Online SVM (TREC07-tftS) - Entry 1	98.65	0.08	0.0250
Online SVM (TREC07-tftS) - Entry 3	98.65	0.07	0.0257
Heilongjiang Institute of Technology - Entry 3	98.66	0.14	0.0303
Online SVM (TREC07-tftS) - Entry 2	98.61	0.07	0.0331
Heilongjiang Institute of Technology - Entry 2	98.64	0.19	0.0557
PPM Compression (TREC07-ijsppm)	94.28	0.01	0.1031
Communication and Computer Network Lab (South China Univ. of Technology) - Entry 3	99.98	27.55	0.1500
Dynamic Markov Compression(TREC07-wat2)	98.11	0.34	0.2988
Communication and Computer Network Lab (South China Univ. of Technology) - Entry 2	99.88	25.53	0.5234
IGF (Ígor Assis Braga) - Entry 3	72.57	0.01	1.4495
IGF (Ígor Assis Braga) - Entry 2	80.59	0.01	8.9047
Kosmopoulos Aris - Entry 2	81.84	51.14	27.1210
Kosmopoulos Aris - Entry 1	86.20	57.20	28.7998