Text Analytics

Text Classification

Ulf Leser

Content of this Lecture

- Classification
- Algorithms
- Case studies

Disclaimer

- This is not a course on Foundations of Machine Learning
- Classification/clustering are presented rather briefly
  - There exit many more methods, much work on comparing them empirically, and a lot of work on explaining the differences between the different approaches
- General experience: Choosing another classification / clustering typically will not lead to dramatic improvements
  - Instances are either well classifiable or not
  - Changing the classification method may yield 5-10% improvement, but usually not more
- More important: Choice of features
  - This 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?
  - Find a function v that maps a doc into a vector of features
    - For instance, its bag-of-words, possibly weighted by TF*IDF
  - Obtain a set D of docs with their classes
  - Find the characteristics of the features of docs in each class (= build a model)
    - What do they have in common?
    - How do they differ from docs in other classes?
  - Encode the model in a classifier function f operating on a feature vector: v: D→V, and f: V→C
    - We compute f(v(d))

Good Classifiers

- Our problem: Finding a good classifier
  – A good classifier assigns as many docs as possible to their "correct class"
- How do we know?
  – Supervised learning
  – Classification needs a sample S of docs with their correct classes
  – S is required for
    - Learning the model
    - Evaluating f: f is the better, the more docs are assign their correct class
      – Details on evaluation methods later

Overfitting

- We can easily build a perfect classifier for S
  - f(d) = {f(d'), if $\exists$d'$\in$S with d'=d; random otherwise)
  - Applied to only docs from S, f is a perfect classifier
- But: This classifier will not work well on "new" documents
- Improvement
  - f(d) = {f(d'), if $\exists$d'$\in$S with d'~d; random otherwise)
  - If S is small and "~" very narrow, this does not help a lot
    - But see kNN classifiers
- Overfitting
  - If the model strongly depends on S, f overfits – it will only work well if all future d's 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 for the docs in S
- Still, 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)
  - Partition S into k sets (typical: k=10)
    - Leave-one-out: k=|S|
  - Learn model on k-1 sets and evaluate on the k'th
  - Perform k times, each time evaluating on another partition
  - Estimated quality on new docs = average performance
  - Often the best we can do

Problem 1: Information Leakage

- Developing a classifier is an iterative process
  - Define feature vector
  - 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; then no more iterations

Problem 2: Biased S

- Very often, S is biased
  - 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 inst. of c’ in S, these are searched actively in D
  - Later, examples from other classes are added
  - But how many?
  - Fraction of c’ in S is much (?) higher than expected by chance
    - 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

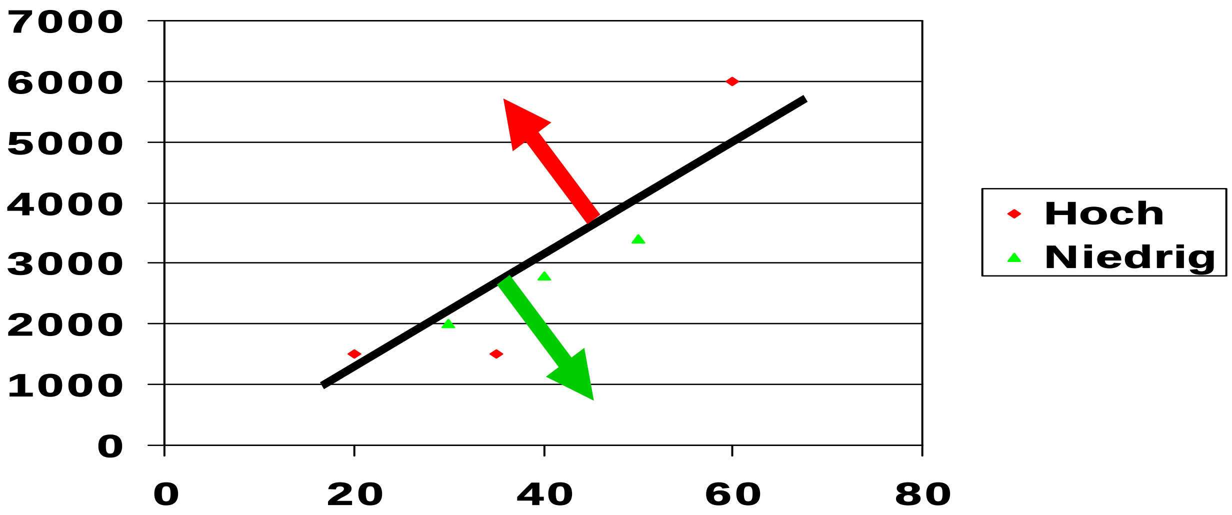
A Simple Example

- An aggregated history of credit loss in a bank

Class	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

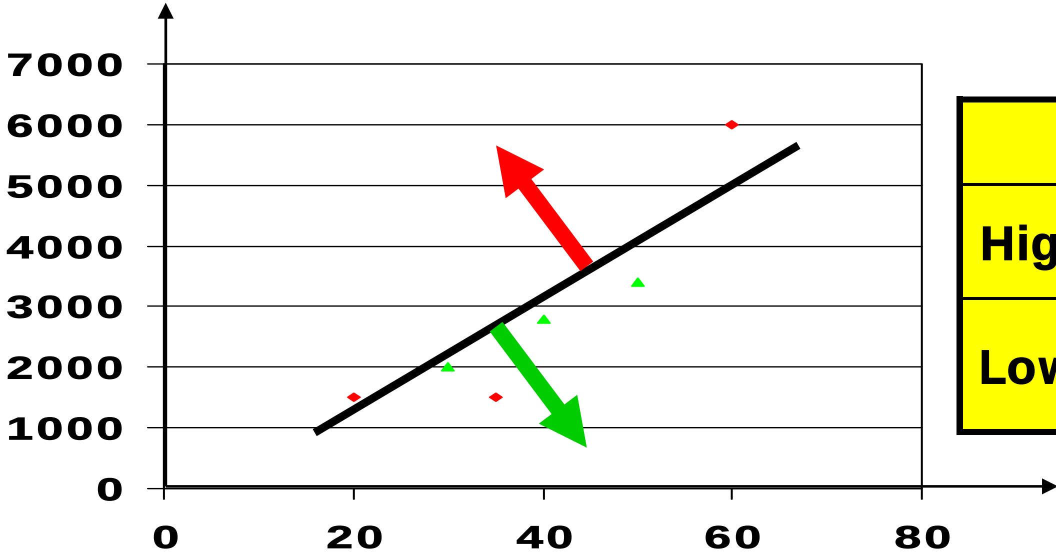
- Now we see a new person, 45 years old, 4000 Euro income
- What is his risk?

Regression



- Simple approach: Linear regression
  - Linear separation with minimum square of error
- Use location relative to regression line as classifier
- Compute parameters such that error is the smallest
  - This is one way of doing it; no details on regression here

Performance on the Training Data



	High	Low
High	2	0
Low	1	3

- Quality of predicting “high risk”
  - Precision = TP/(TP+FP)= 2/2, Recall = TP/(TP+FN) = 2/3, Accuracy: 5/6
- Regression makes many assumptions
  - Assumes linear correlations between attributes
  - Requires numerical attributes
  - Method of choice if C is continuous (infinitely many ordered classes)

Categorical Attributes

Class	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 classification was created by some insurance manager. What was in his head?
  - 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

Class	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 these data sets, result in the same classification?

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

A Third Approach

Class	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 always 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 you method using cross-validation

Text Classification

- Many problems in text analytics can be cast as classification
  - Language identification
  - Topic identification
  - Spam detection
  - Content-based message routing
  - Named entity recognition (is this token part of a NE?)
  - Author identification (which plays were really written by Shakespeare?)
  - …
- Common problem
  - No well discriminating single features
  - We need to use a high dimensional feature space

Classification Methods

- There are a zillion different methods
  - k-nearest neighbor
  - Naïve Bayes and Bayesian Networks
  - Decision Trees and Rainforests
  - Maximum Entropy, Maximum Entropy Markov Models, Conditional Random Fields
  - Support Vector Machines
  - Perceptrons, Neural Networks
  - …
- Effectiveness of classification depends on problem, algorithm, feature selection method, sample, evaluation, …
  - But: Often the difference between different methods are astonishing small

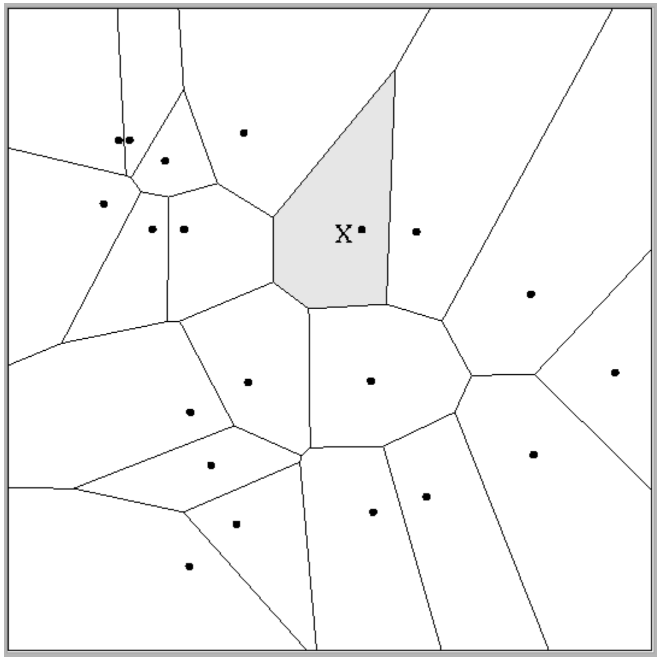
Content of this Lecture

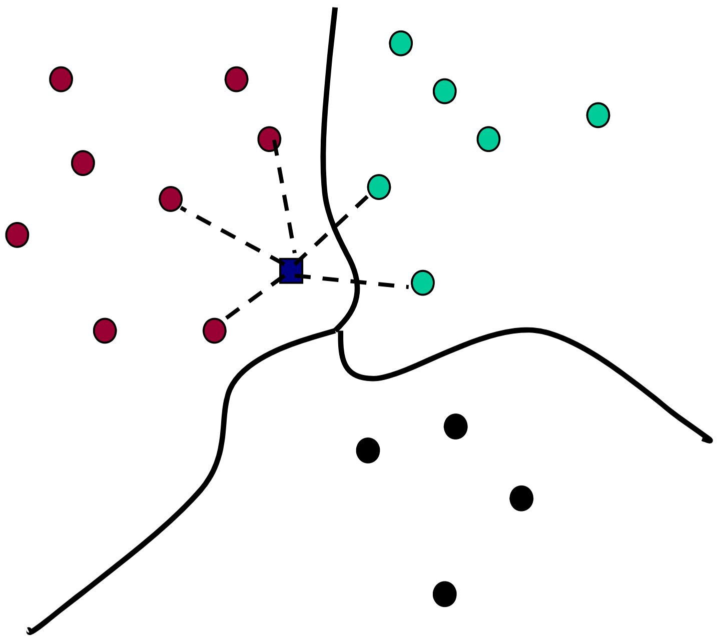
- Classification
- Algorithms
  - Nearest Neighbor
  - Naïve Bayes
  - Maximum Entropy
  - Linear Models and Support Vector Machines (SVM)
- Case studies

Nearest Neighbor Classifiers

- Very simple and effective method
- 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
  - Obviously, a proper distance function is very important
  - 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





1NN and Voronoi diagram marking the regions of influence around each data point (2D-space, each point a separate class)

A 5NN

Properties

- Assumption: Similar docs should have the same class
  - Depends a lot on the distance function
- kNN is simple and astonishing good
- kNN in general is more robust than NN
- (k)NN is an example of lazy learning
  - Actually, there is no learning
  - Actually, there is no model
  - Where are the features?
- Features
  - We still need to define features
  - These features are the input to the distance function

Disadvantages

- Major problem: Performance (speed)
  - We need to compute the distance between d and any doc in S
  - This requires d*|S| applications of the distance function
    - Often the cosine of two 100K-dimensional vectors
- Various suggestions for speeding-up the method
  - Clustering – aggregate groups of very close points in S into a single representative
    - Linear speed-up
  - Extreme case: Chose one representative per class
    - Usually not a good idea (high dimensional space!); no kNN any more; very fast and space efficient
  - Multidimensional index structures and metric embeddings
    - Map into a lower-dimensional space such that distances are preserved

kNN for Text

- In the VSM world, kNN is implemented very easily using the methods 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 usually much larger than the average 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

- Simple method based on relative frequencies of features in the different classes
- Given
  - Set S of docs and set of classes $C=\{c_1, c_2, \ldots c_m\}$
  - Docs are described as a set F of binary features
    - Usually the presence/absence of terms in d
- We seek $p(c_i|d)$, the probability of a doc $d \in S$ being a member of class c_i
- d eventually is assigned to c with $p(c|d) = \text{argmax } p(c_i|d)$
- Replace d with feature representation

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

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 $c \in C$
    - 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

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

Term can be dropped; value is identical for all classes, and we only want to rank the p(c|d)

Naïve Bayes

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

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

- And finally

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

Properties

- Simple algorithm, quite robust, comparably fast, needs extensive smoothing
- Often used as baseline for other methods
- Learning the model is simple, and the model is quite compact (O(|K|*|C|) space)
- When we use the logarithm (equally well for ranking), we see that NB is a (log-)linear classifier

$$
\begin{aligned}
p(c \mid d) &\approx \log\Big(p(c) * \prod p(t_i \mid c)\Big) \\
&= \log\big(p(c)\big) + \sum \log\big(p(t_i \mid c)\big)
\end{aligned}
$$

Feature Selection

- One can easily speed-up classification by using only a subset of all features
- Simplest case: Use those t where p(t|c) show the biggest differences between the different classes
- Numerous methods for feature selection
  - Information gain, statistical tests, Bayesian information criterion, GINI score, …
  - 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"
- Same methods benefit from feature selection, some not
  - SVM usually not, Bayes usually yes (think of redundant features)

Content of this Lecture

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

Discriminative versus Generative Models

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

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

- Notation
  - Approaches that estimate p(d|c) are called generative
    - p(d|c) is the probability of class c producing data d
    - Thus, NB is a generative model
  - Approaches that directly estimate p(c|d) are called discriminative

Maximum Entropy Modeling

- Maximum Entropy (ME) is discriminative
- Given a set of binary features, it directly learns conditional probabilities p(c|d)
- Definition
  *Let s_{ij} be the score of feature i for doc d_j (such as TF*IDF). We derive from s_{ij} a binary indicator function f_i for doc j and class c:*

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

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

Classification with ME

- Since p(c,d)=p(c|d)*p(d) and p(d) is the same for all c, we directly use p(c|d)~p(c,d)
- The ME approach models the joint probability p(c,d) as

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

  - Z is a normalization constant
  - The feature weights α_i are learned from the data
  - K is the number of features
- Classification with ME
  - Compute p(c,d) for all c and return the class with the highest value

Finding Feature Weights

- Of course, the problem is finding appropriate α_i
- We want to choose the α_i such that the probability of the training data S given the model M is maximized

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

- This choice must take the dependencies between the features in the model into account
- Naïve Bayes computes α-like values independently for each feature and uses their linear combination for classification
  - This only works if statistical independence holds
  - For instance, using the same feature multiple times does bias the NB result

Maximum Entropy Models

- Essentially, ME applies a search strategy to find those α_i
- Problem: There are indefinitely many combinations of weights that may all give rise to the same maximal probability of S
- ME chooses the model with the largest entropy
  - Abstract formulation: The training data leaves too much freedom. We want to choose M such that all "undetermined" probability mass is distributed equally
    - ME tried to make as few assumptions as possible given the data
  - Such a distribution exists and is unique
  - The search strategy needs to take this into account

Entropy of a Distribution

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

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

- Thus, ME searches 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. Now we want to classify an document d. Which probabilities would you 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

- Now 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

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 now 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 more complicated if we have >100k features

Example 2 [Pix,Stockschläder, WS07/08]

- Assume we count features “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,blue) of blue-eyed, left-handed tamarins?
  - We don’t now
  - 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 here 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 only)

- How do we find M in general?
- Generalized Iterative Scaling
  - Iterative procedure finding the optimal solution
  - Essentially, it starts from a random guess of all the p(c,d) and iteratively redistributes probability mass until convergence
  - See [MS99] for the algorithm
- Problem: Usually converges very slowly
  - Long training times
- Several improved algorithms 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
  - Feature weights are learned by always taking the entire distribution into account
  - Two “redundant” features will simply get 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

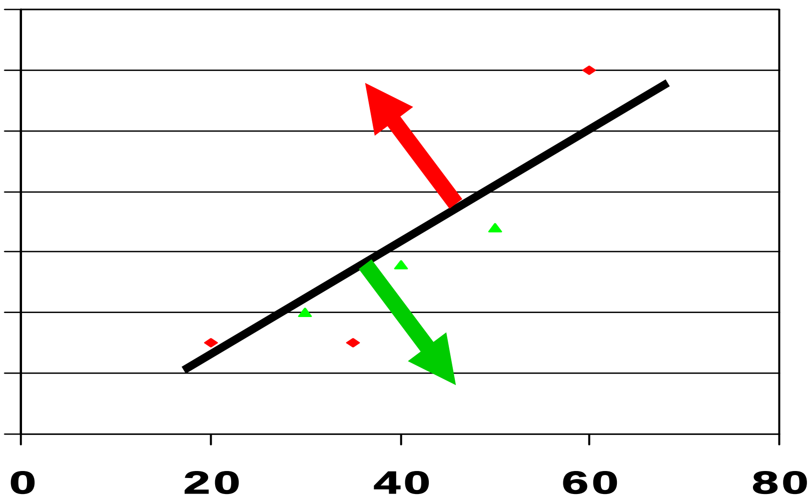
- Classification
- Algorithms
  - Nearest Neighbor
  - Naïve Bayes
  - Maximum Entropy
  - Support Vector Machines (SVM)
- Case studies

Class of Linear Classifiers

- Many common classifiers are (log-)linear classifiers
  - Naïve Bayes
  - Perceptron / Winnow
  - 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 of the infinite number of possible separating hyperplanes is chosen?
  - How are non-separable data sets (by a linear model) handled?
- Experience: Classifiers more powerful than linear often don't perform better (on text)

NB and Regression

- Using linear regression, we compute a separating hyperplane using error minimization
- If we assume binary Naïve Bayes, we may compute



$$\frac{p(c|d)}{p(\neg c|d)} \approx \log\left(\frac{p(c)}{p(\neg c)}\right) + \sum \log\left(\frac{p(t_i|c)}{p(t_i|\neg c)}\right)$$

$$= a + \sum_{k \in K} b_k * TF_k$$

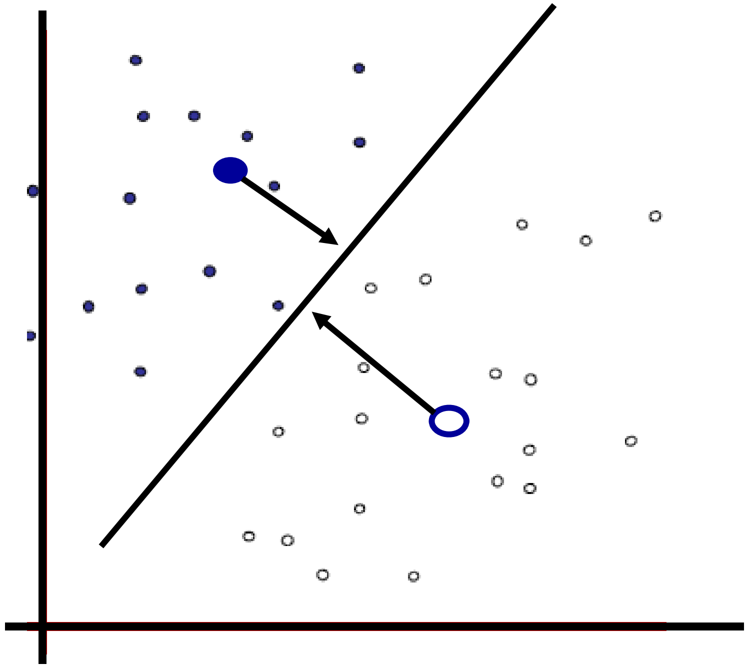
This is a linear hyperplane; value>0 gives c, value<0 gives not c

ME is a Log-Linear Model

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

Roccio Classification

- Recall relevance feedback in the VSM using Roccio
  - Compute initial result
  - Build new query by aggregating all true positives and discounting all/some false positives
- This idea can be turned into a classifier
  - Compute the centroid of all positive examples
  - Compute the centroid of all negative examples
  - Compute the hyperplane with minimal distance to both centroids

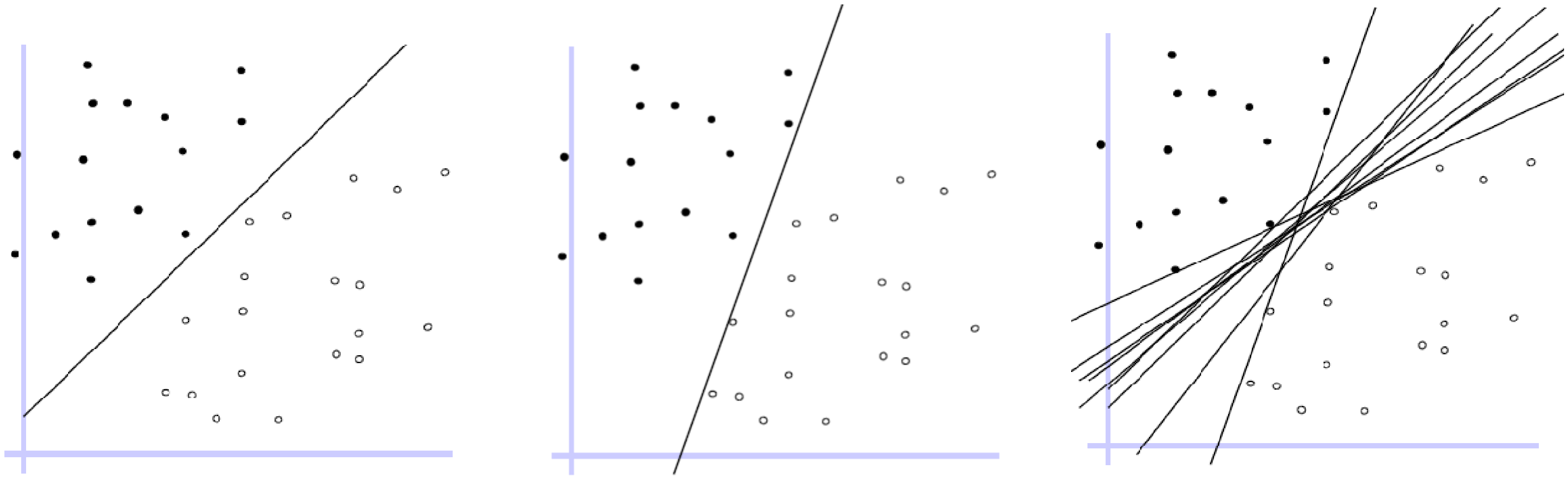


Text = High Dimensional Data

- Document co-ordinates are zero along almost all axes
- Most document pairs are very far apart (i.e., not strictly orthogonal, but only share very common words)
- In classification terms: virtually all document sets are separable for essentially any classification
  - 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

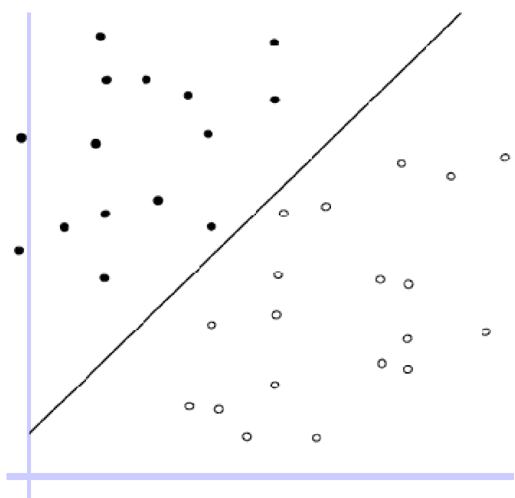
- Hyperplane separating classes in high dimensional space
  - For illustration, we stay in 2-dimensional
- But which?

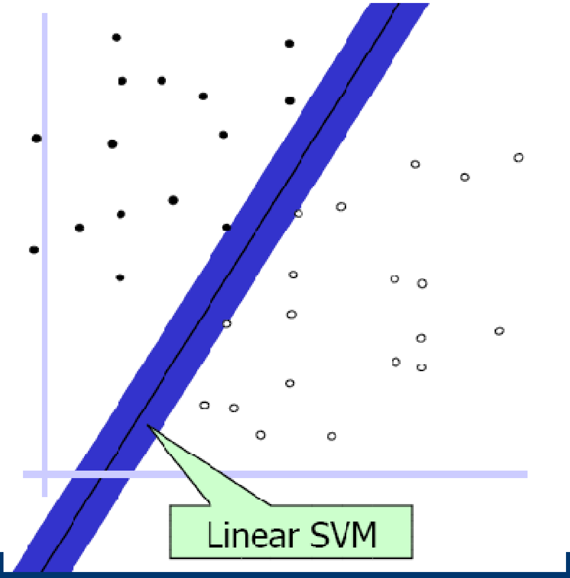


Quelle: Xiaojin Zhu, SVM-cs540

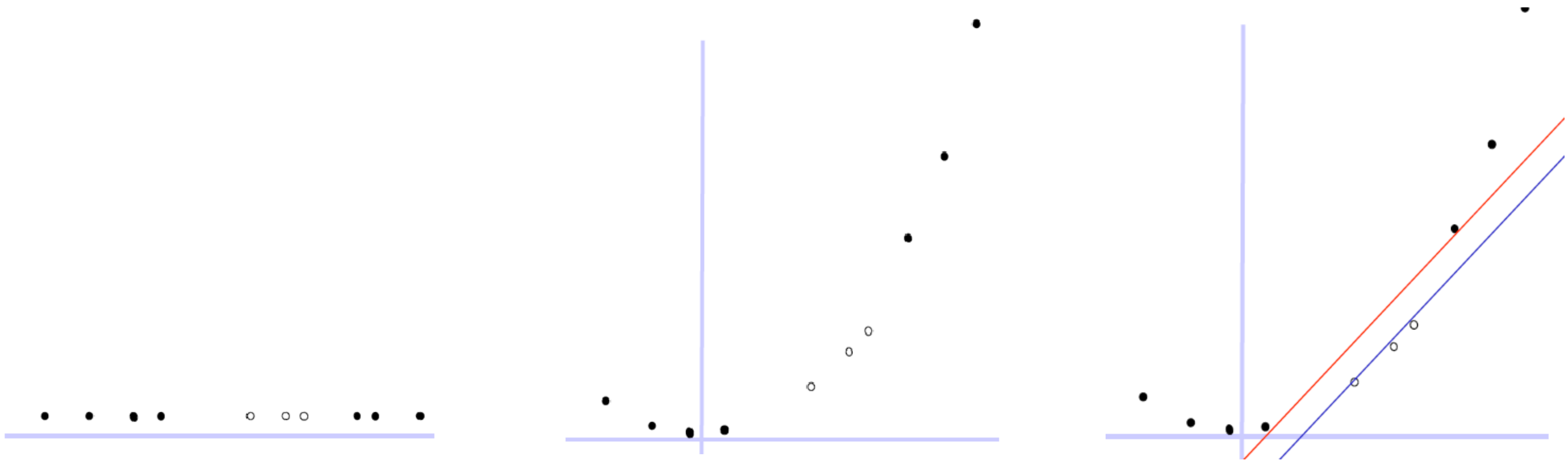
Support Vector Machines (sketch only)

- SVMs compute the hyperplane which maximizes the margin
  - I.e., is as far away from any data point as possible
- Can be cast in a linear optimization problem and solved efficiently
  - Classification finally only depends on the support vectors – efficient
    - Points most closest to hyperplane
  - Complication since usually the classes are not linearly separable
  - Minimizes the error under some assumptions





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

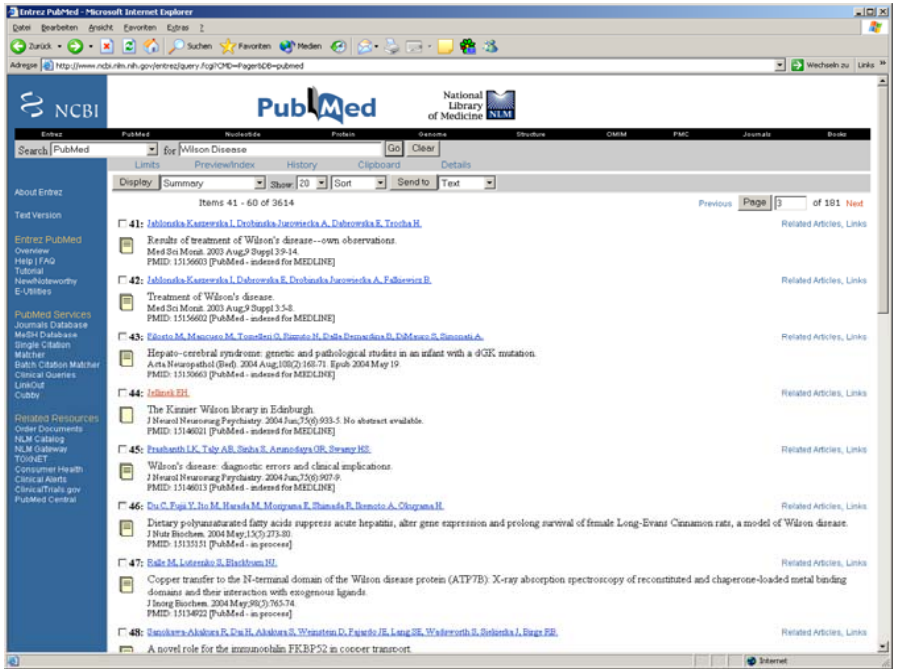
- State of the art in text classification
- Might require long training time
  - Worst case quadratic in training data
  - Various clever tricks and heuristics exist
- Classification is rather fast
  - Only distance to hyperplane is needed
  - Hyperplane is defined by only few vectors (support vectors)
- SVM are quite good "as is", but lot of tuning possible
  - Kernel function, biased margins, …
- Several 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



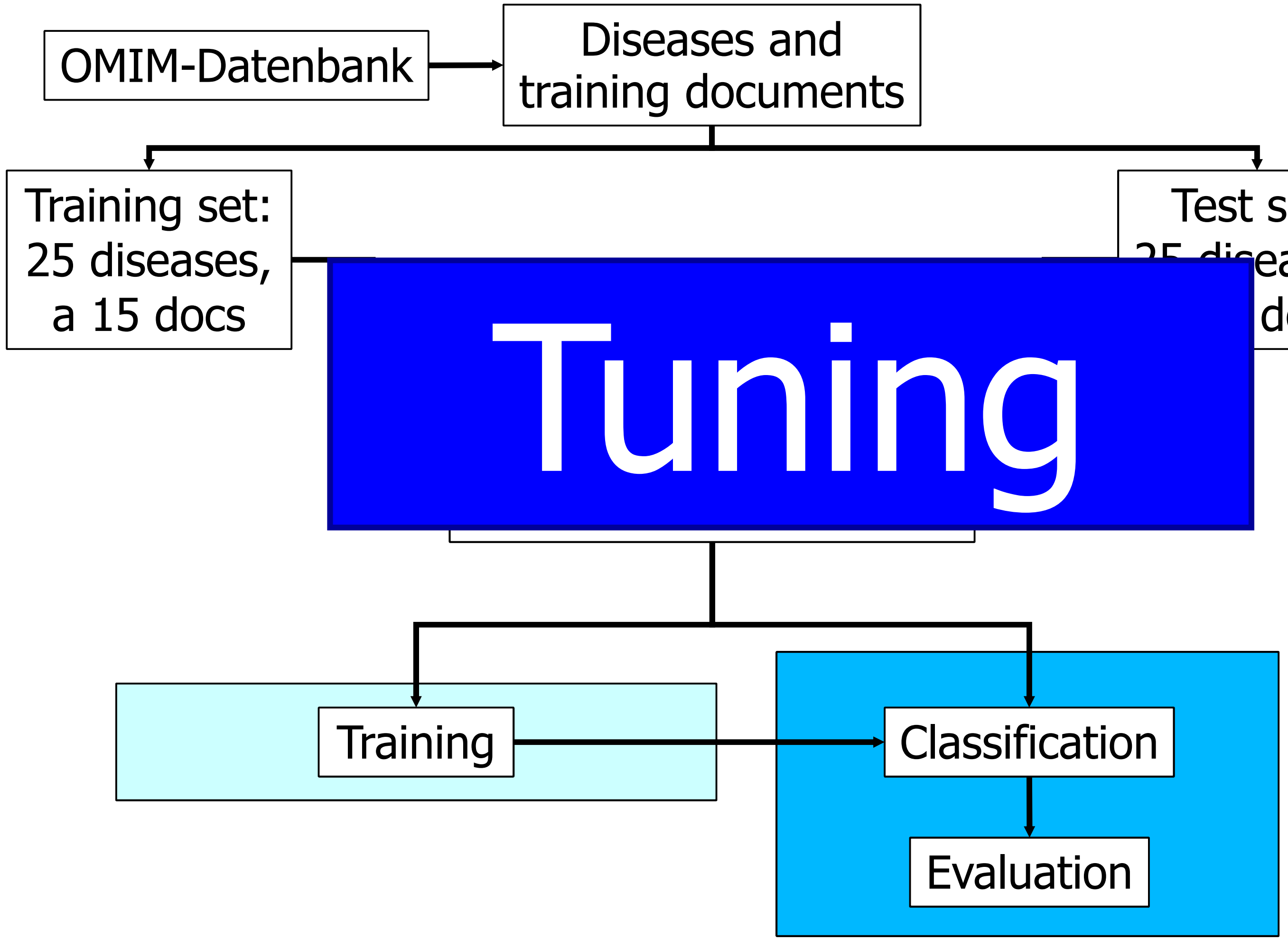
Idea

- 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

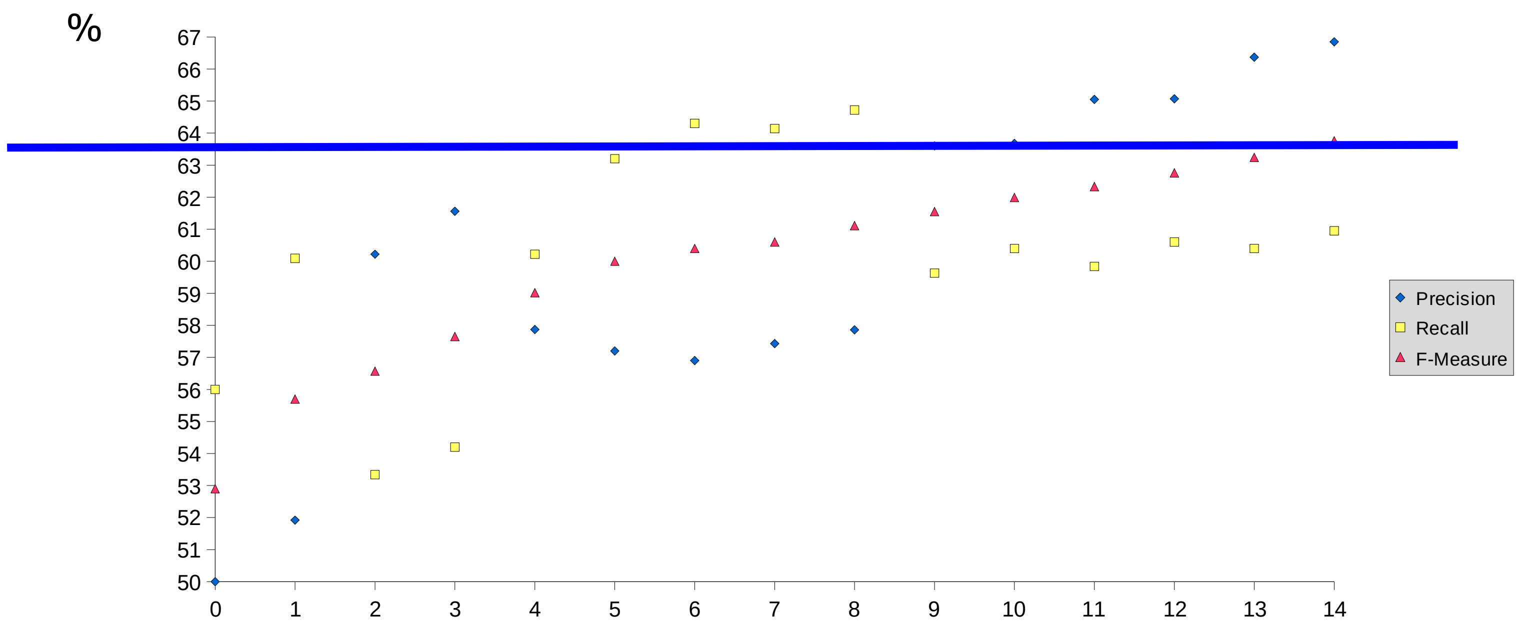
- Classification: Given a new text, find the model which fits best and predict the associated class (disease)

- What can we learn from 20 documents?

Complete Workflow

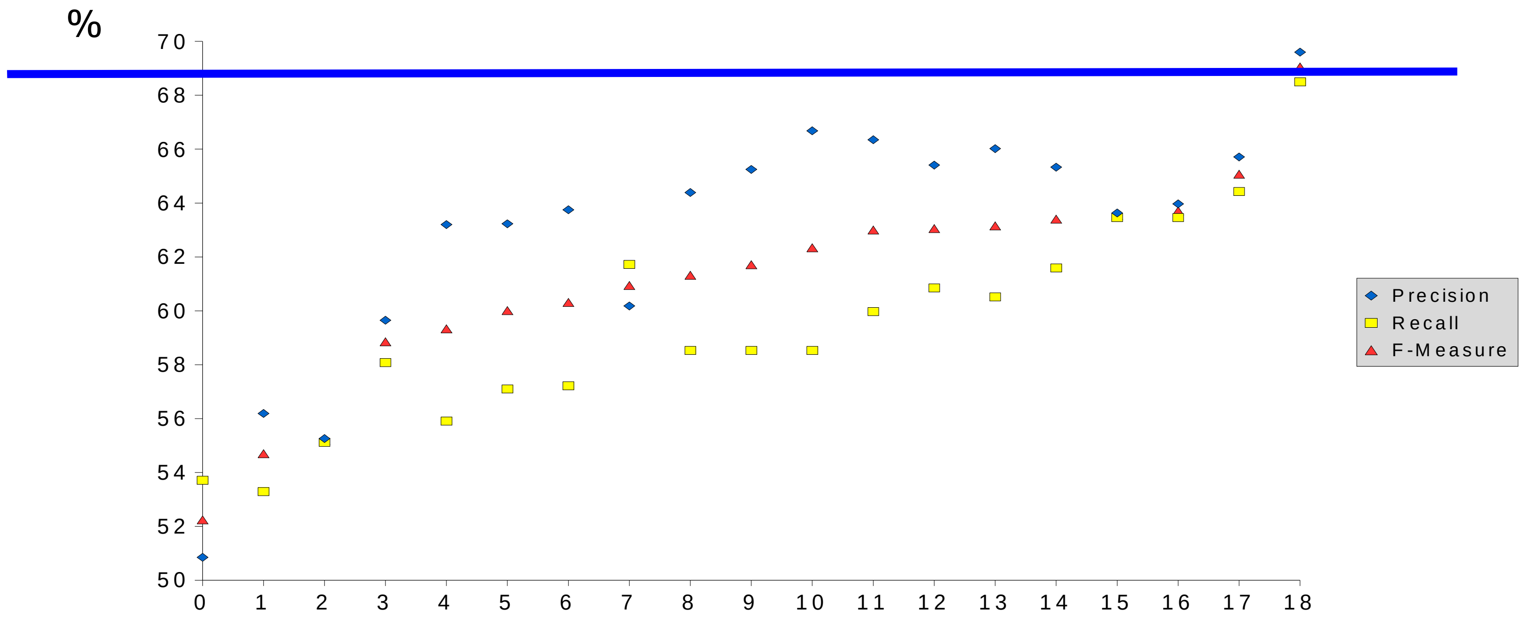


Results (Nearest-Centroid Classifier)



- 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



- For fixed configuration, 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**

Naive Bayes

- Best results

Versuche:	A	B	C	D	E
Precision	64.55	64.80	**66.00**	69.94	64.55
Recall	62.82	62.61	**65.35**	55.20	62.82
F-Measure	63.67	63.69	**65.68**	61.70	63.67

  - A = Stemmer: ON, Stoppwörter: 10 000, Nomen-Tagging: ON, VerbenTagging: ON
  - B = Stemmer: OFF, Stoppwörter : 10 000, Nomen-Tagging: ON, VerbenTagging: ON
  - C = Stemmer: ON, Stoppwörter : 10 000, Nomen-Tagging: ON, VerbenTagging: OFF
  - D = Stemmer: ON, Stoppwörter : OFF, Nomen-Tagging: OFF, VerbenTagging: OFF
  - E = Stemmer: ON, Stoppwörter : 10 000, Nomen-Tagging: ON, VerbenTagging: ON

- Nearest Centroid outperforms Naïve Bayes
  - On this particular problem and training set …

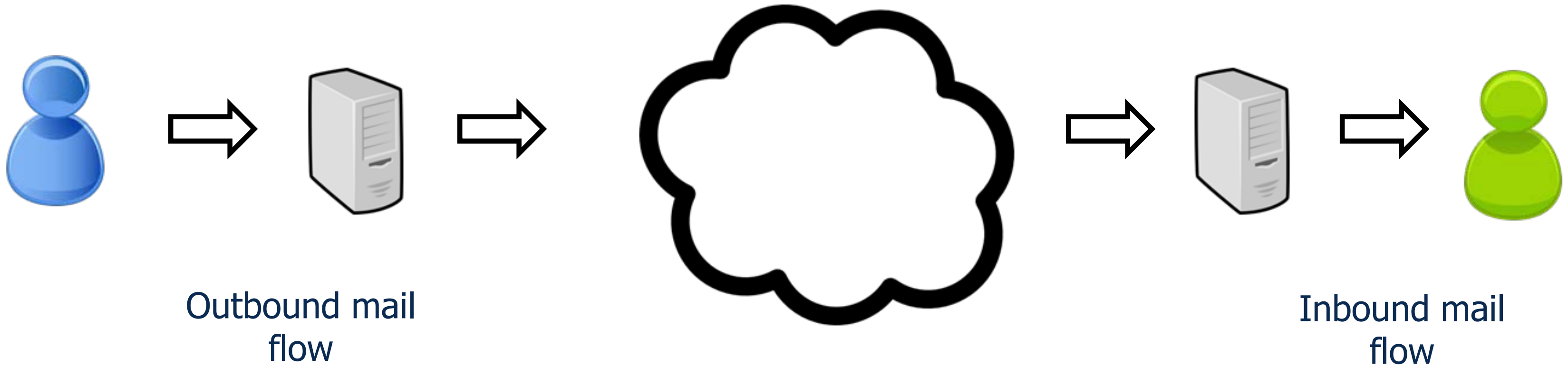
Content of this Lecture

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

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

Spam

- Unsolicited Bulk E-Mail
- Old „problem“: 1978 first spam e-mail 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

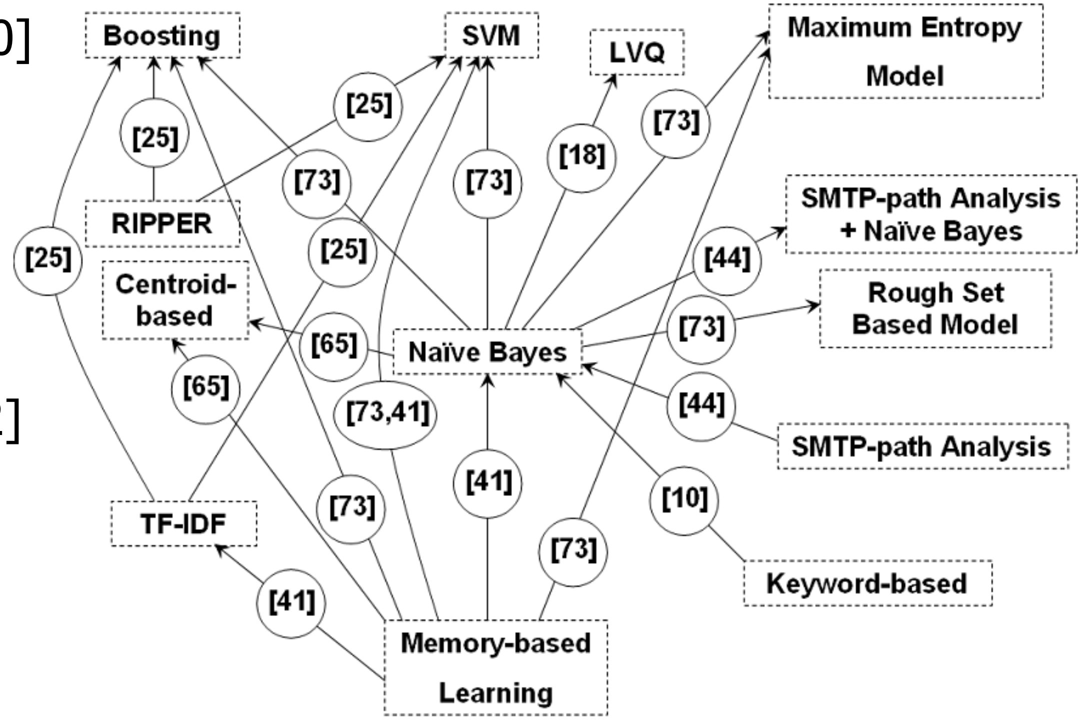


SPAM Detection as a Classification Task

- 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, relatively fast classification
  - Using TF, TF-IDF, Information Gain, …
  - Stemming (mixed reports)
  - Stop-Word removal (seems to help)

Many Further Suggestions

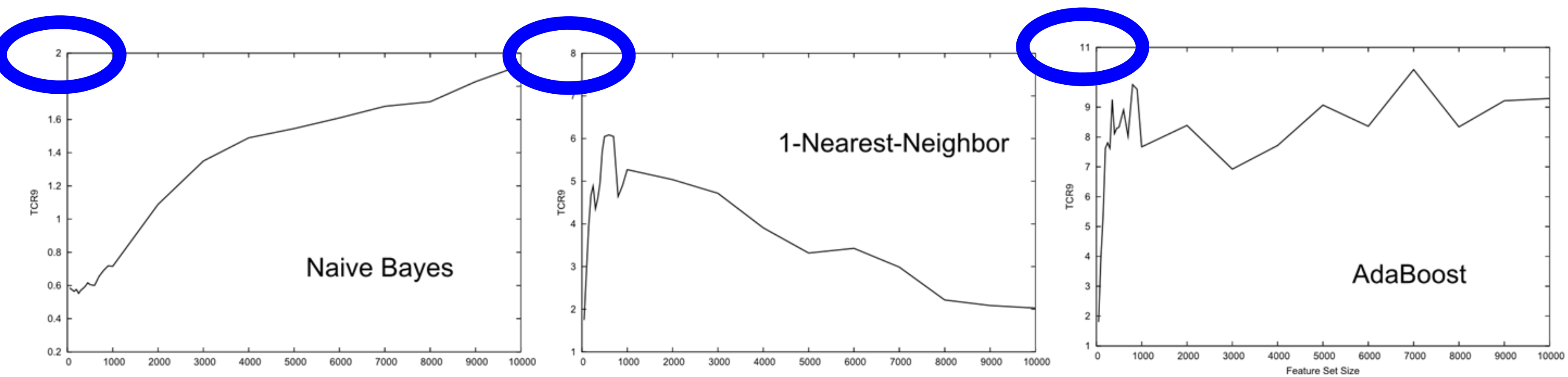
- Rule learning
  [Cohen, 1996]
- k-Nearest-Neighbors
  [Androutsopoulos *et al.*, 2000]
- SVM
  [Kolcz/Alspector, 2001]
- Decision trees
  [Carreras/Marquez, 2001]
- Centroid-based
  [Soonthornphisaj et al., 2002]
- Artificial Neural Networks
  [Clark *et al.*, 2003]
- Logistic regression
  [Goodman/Yih, 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%)
- Spam ≠ Spam: Concept drifts
  - 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