Maschinelle Sprachverarbeitung

Text Classification

Ulf Leser

Content of this Lecture

- Classification
  - Approach, evaluation and overfitting
  - Examples
- Classification Methods
- Feature Selection
- Case studies

Disclaimer

- This is not a course on Machine Learning
- Methods are presented from an applied point-of-view
  - There exist more methods, much work on empirical comparisons, and a lot of work on analytically explaining differences between methods
- Experience: Choosing another classification / clustering method typically does not lead to dramatic improvements
  - Problems are either "well classifiable" or not
  - Most methods find the most discriminating properties
- Decisive for performance: Choice of features, representing the text, preprocessing
  - Requires creativity and must be adapted to every problem

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)?
  - Design function v mapping a doc into feature vector (feature space)
    - E.g. bag-of-words, possibly TF*IDF
  - Obtain a set S of docs with their classes (training data)
    - Often, this is the most critical issue
  - Find the characteristics of the docs in each class (model)
    - Which feature values / ranges are characteristic?
    - What (non-linear?) combinations of features 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

- Problems
  - Finding enough training data
  - Finding the best pre-processing (tokenization, case, POS tag set …)
  - Finding the best features
  - Training a good classifier
    - Assigning as many docs as possible to their correct class
- How do we know?
  - Use a (separate) gold standard data set
  - Use training data in two roles (beware of overfitting)
    - Learning the model
    - Evaluating the model

Problem 1: Overfitting

- Let S be a set of texts with their classes (training data)
- We can easily build a perfect classifier for S
  - f(d) = {f(d'), if ∃d'∈S with d'=d; random otherwise)
  - f is perfect for any doc from S
  - But: Produces random results for any new document
- Improvement
  - f(d) = {f(d'), if ∃d'∈S with d'~d; random otherwise)
  - Improvement depends on |S| and definition of "~"
  - 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
- But usually 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 2: Information Leakage

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

Problem 3: 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
- Classification Methods
- Feature Selection
- Case studies

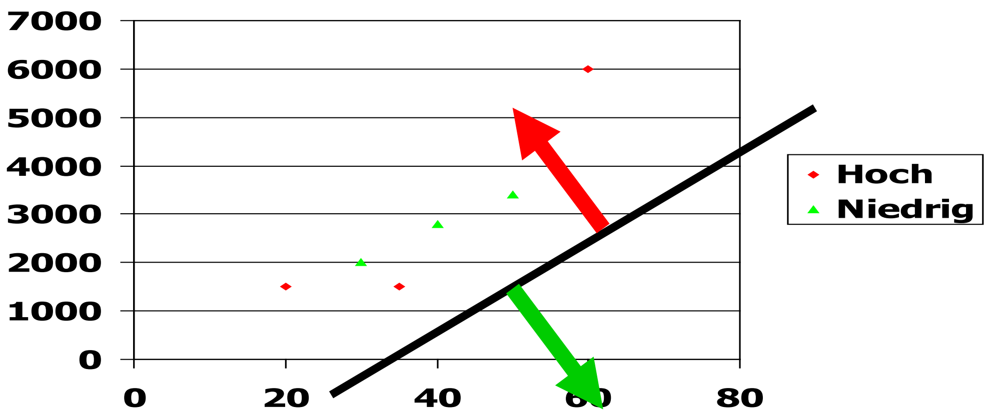
A Simple Example

- 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

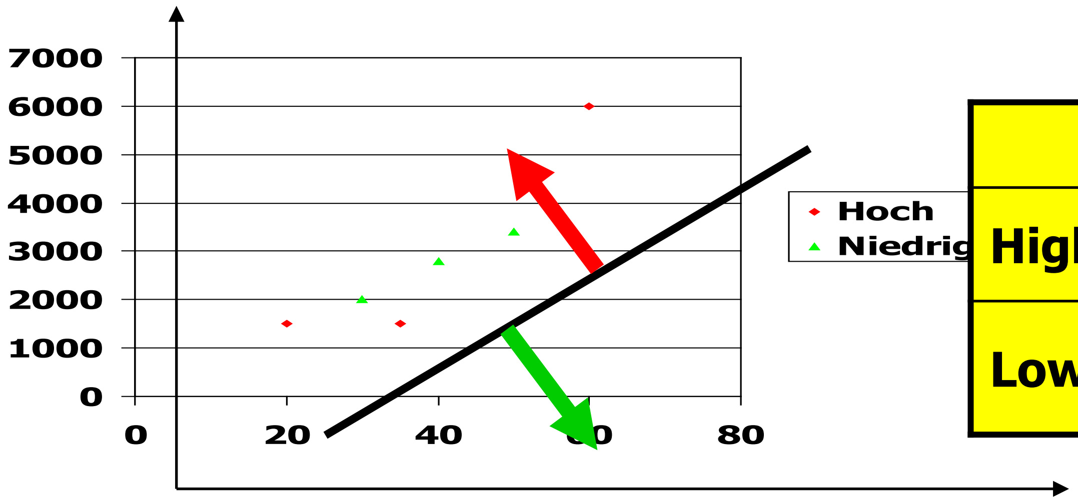
- Now we see a new person, 45 years old, 4000 Euro income
- What is the risk?

Regression



- Simple approach: Separating hyperplane
  - Linear separation by line with the minimum squared error
  - Use location relative to regression line as classifier
  - [Many tricks to improve this principle]

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: Linearly separable problem, feature ranges correlate with class memberships, 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
```

Why 50? Why not 43/67?

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?
  - Less rules – less influence of training data – better generalization

```
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
```

Content of this Lecture

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

Classification Methods

- There are many classification methods
  - Bayesian Networks, Graphical models
  - Decision Trees and Random Forests
  - Linear / Logistic Regression
  - Perceptrons, Neural Networks [deep learning]
  - …
- Effectiveness of classification depends on problem, algorithm, feature selection method, sample, evaluation, …
- Differences when using different classification methods on the same data/representation are often astonishing small

Nearest Neighbor Classifiers

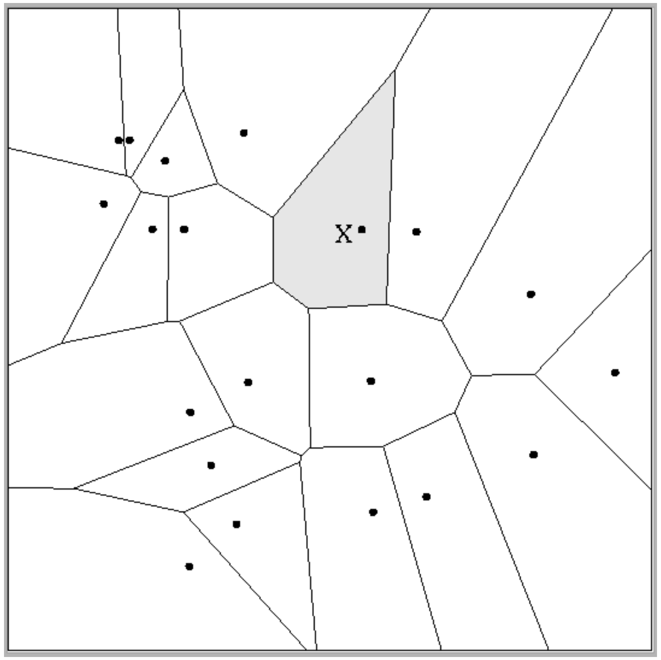
- Definition

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

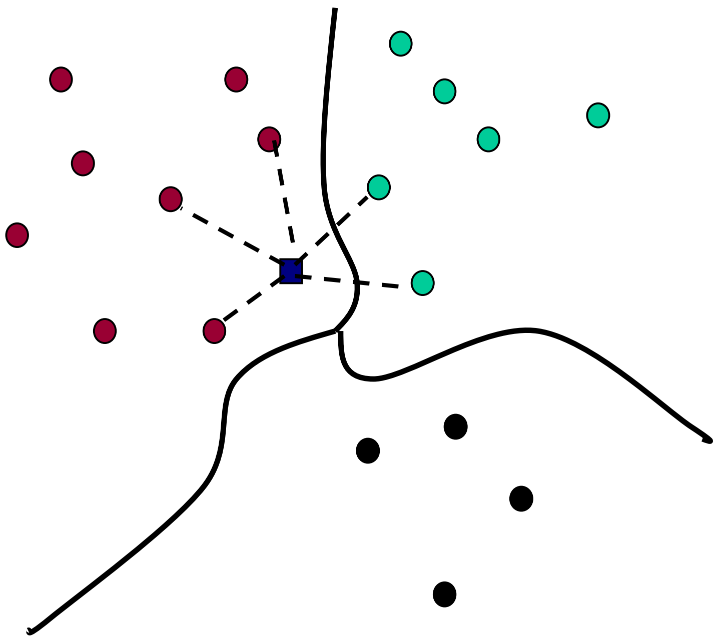
  - *A nearest-neighbor (NN) classifier assigns to d the class of the nearest document in S wrt. m*
  - *A k-nearest-neighbor (kNN) classifier assigns to d the most frequent class among the k nearest documents in S wrt. m*

- Remarks
  - Very simple and effective, but slow
  - We may weight the k nearest docs according to their distance to d
  - We need to take care of multiple docs with the same distance

Illustration – Separating Hyperplanes

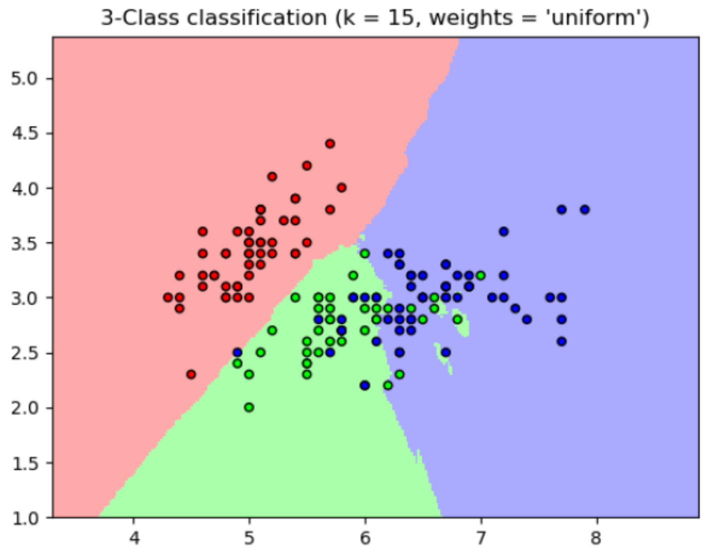


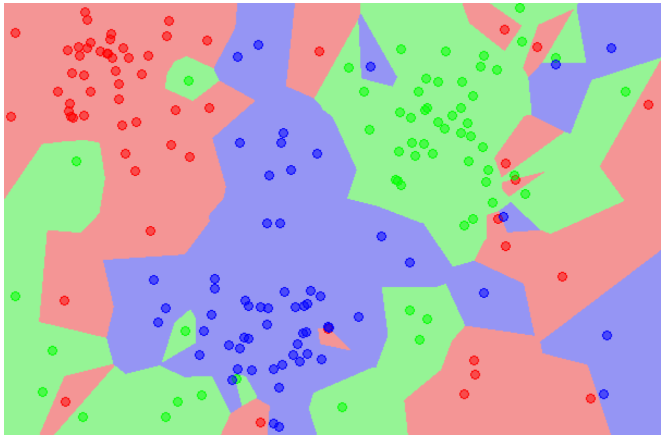
Voronoi diagram in 2D-space
(for 1NN)



5NN

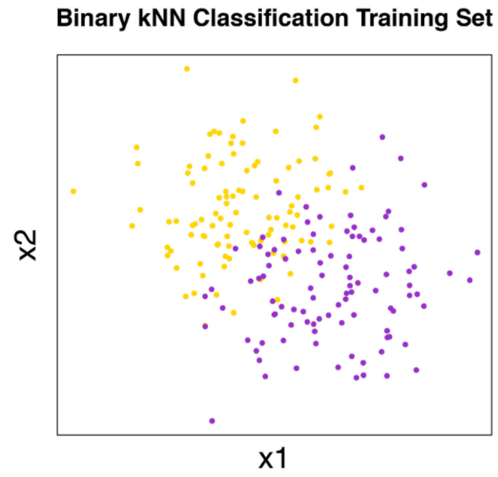
Examples



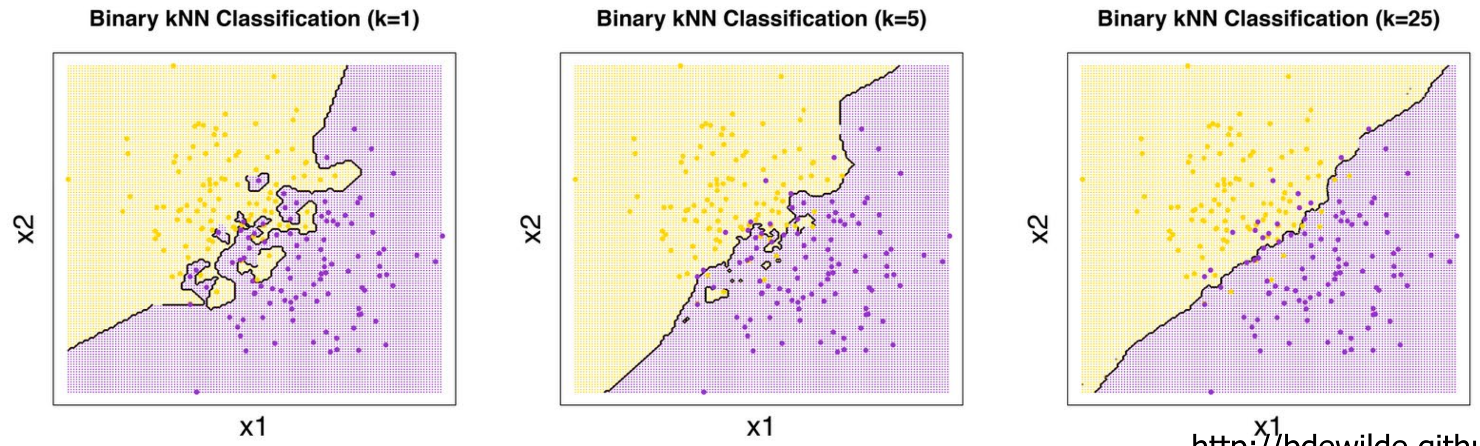


- Assumes dense and homogeneous areas
- Need not cover the space entirely (generalization)
- Border points are difficult to classify (impact of k)
- Can cope with nested areas (impact of k)

Impact of k



- Larger k
  - Spurious objects are errors and smoothed
  - Less effects of single objects
  - Class frequencies become very important
- Very large k
  - Important subareas might disappear
  - Dominant class invades minority class



http://bdewilde.github.io/blog

Properties

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

Disadvantages

- How to choose k?
- 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
  - 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
    - Document to be classified usually has many more keywords than a typical IR-query q
    - We need other ways of optimizing queries

Content of this Lecture

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

Bayes` Classification

- Uses frequencies of feature values in the different classes
- Given
  - Set S of docs and set of classes $C=\{c_1, c_2, \ldots c_m\}$
  - Docs are represented as feature vectors
  - For now, assume only categorical attributes (e.g. 0/1 coordinates)
- We seek $p(c_i|d)$, the probability of a doc $d \in S$ being a member of class c_i

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

- d eventually is assigned to c_i with argmax $p(c_i|d)$

Probabilities

- What we (can) easily learn from the training data (MLE)
  - The a-priori probability p(t) of every term (feature) 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
    - Needs smoothing if there are few docs in c
- Rephrase and use Bayes�too theorem

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

Naïve Bayes

- We have $p(c \mid d) \approx p(t_1,\ldots,t_n \mid c) * p(c)$
- The first term on the right 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

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

- Finally

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

Naive Bayes for Continuous Values

- We assumed features to be categorical
  - Computed relative frequencies of each value in each class
  - This is called Multinomial Naïve Bayes
- What if a feature has a continuous, ordered domain?
  - Simple option: Precompute ranges (bins) of values and transform feature into one feature per range
    - Also called discretization
    - Problem: Which ranges?
    - Information on order of values get lost (no "hyperplanes")
  - Gaussian Bayes: Approximate values per class by a normal distribution and use probability of given value given this distribution
    - Compute mean and STDDEV
    - Make sure values are roughly normally distributed first!

Properties

- Simple, robust, fast
- Needs smoothing: Avoid probabilities to become zero
- Instead of taking the most probable class, one may also take the class where p(c|d)-p(¬c|d) is maximal
- Efficient learning, space-efficient model (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

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

Content of this Lecture

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

Discriminative versus Generative Models

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

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

- Approaches that estimate p(d|c)*p(c) are called generative
  - p(d|c) is the probability of class c producing data d
  - Naïve Bayes, HMM are a generative models
- Approaches that estimate p(c|d) are called discriminative
  - But: We only have a very small sample of the document space
  - The training data – always small compared to size of doc space
  - Logistic Regression, MaxEnt, SVM are discriminative models

Discriminative Models

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

- We cannot know the true probabilities
  - We have seen too few combinations of terms
- Idea: Learn a function over features computing the class
  - That's the trick: Not just counting relative frequencies
- Problem: As our data is so sparse, there usually are many functions performing equally well on training data
  - Generalization is very difficult
- Maximum Entropy: Use that function that makes the least assumptions apart from the training data
  - And use a particular class of function which allows this idea to be implemented efficiently

Maximum Entropy (ME) Modeling

- Given a set of (binary) features derived from d, MEM 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 and we only want to rank, we also have 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

- MEM 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 scores into probabilities
  - The feature weights α_i are learned from the data
  - K is the number of features (often very many – many parameters)
  - This particular function allows efficient learning (later)
- Classification: Compute p(c,d) for all c and return class with highest probability

Maximum Entropy Principle

- MEM learning: Learning optimal feature weights α_i
- Choose α_i such that probability of the training data S given M is maximal

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

- Problem: There are usually many combinations of weights that all give rise to the same maximal probability of S
- ME chooses the model with the largest entropy
  - Intuition: The training data leaves too much freedom. We choose M such that “undetermined” probability mass is distributed equally
  - Such a distribution exists and is unique
  - Computation of α_i takes this idea into account as an additional optimization goal

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

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

- MEM: 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
  - Thus: p(+up)=10%, p(-up)=90%, p(A|+up)+p(A|-up)=55%, …
  - Note: Table gives joint probabilities, not conditional probabilities

	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

Emperor tamarin

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 (computing derivatives) is not possible

Generalized Iterative Scaling (idea)

- No analytical solution to the general optimization problem exists (with many features and some sums given)
- Generalized Iterative Scaling
  - 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]
- Problem: Usually converges very slowly
- Several faster variations 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
- Popular in statistical NLP (still?)
  - 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

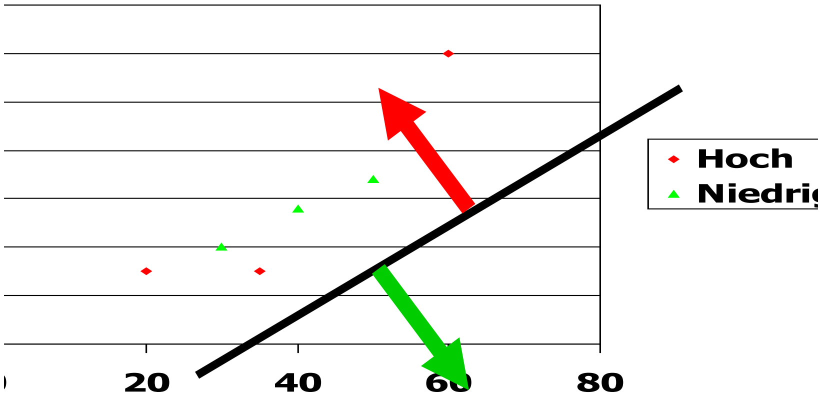
- Classification
- Classification Methods
  - Nearest Neighbor
  - Naïve Bayes
  - Maximum Entropy
  - Support Vector Machines (SVM)
- Feature Selection
- Case studies

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 $\neg$c

ME is a Log-Linear Model

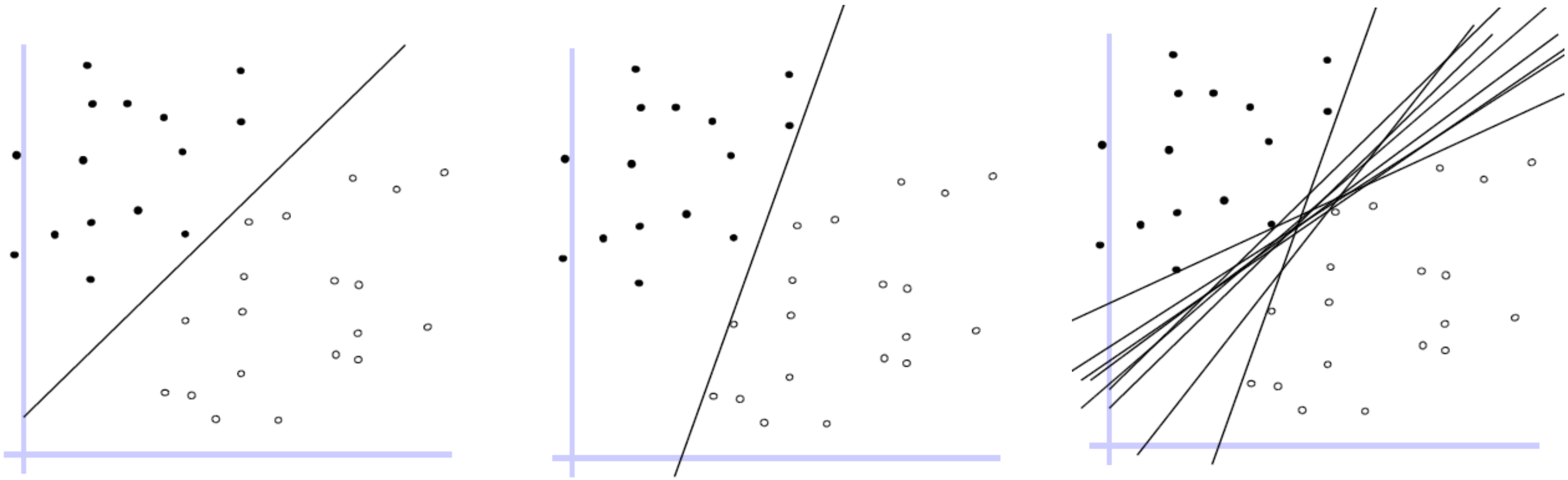
$$p(c,d) = \frac{1}{Z} * \prod_{i=1}^{K} \alpha_i^{f_i(d,c)} \approx \log\left(\frac{1}{Z}\right) + \sum_{i=1}^{K} f_i(d,c) * \alpha_i$$

Text = High Dimensional Data

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

Linear Classifiers (2D)

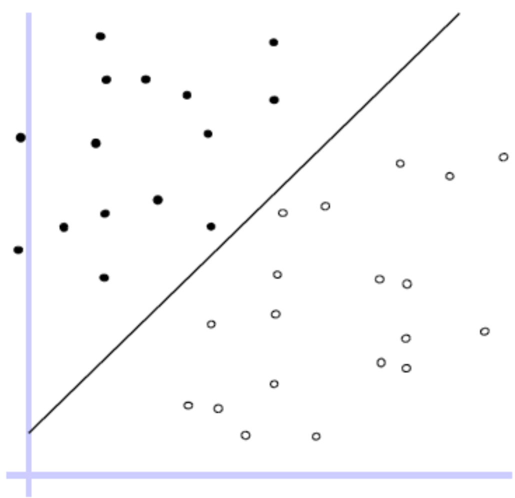
- Hyperplane separating classes in high dimensional space
- But which?

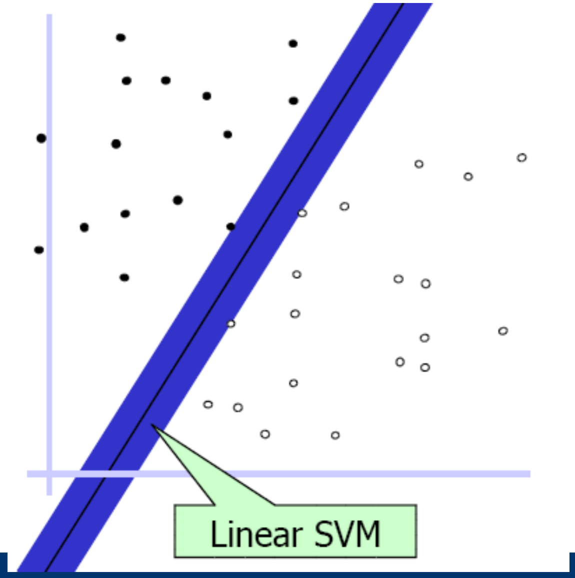


Quelle: Xiaojin Zhu, SVM-cs540

Support Vector Machines (sketch)

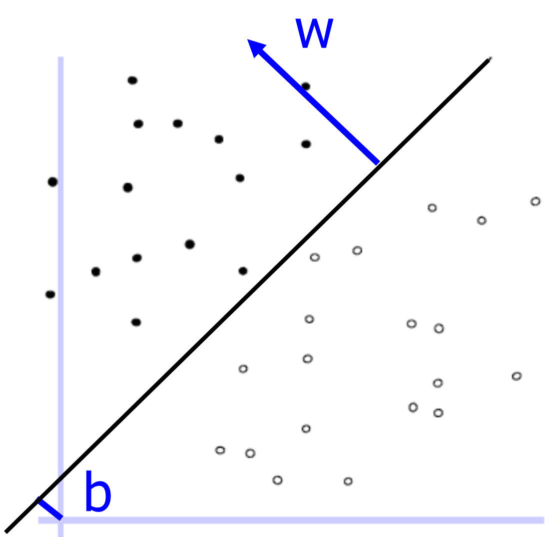
- SVMs: Hyperplane which maximizes the margin
  - I.e., is as far away from any data point as possible
  - Cast in a linear optimization problem and solved efficiently
  - Classification only depends on support vectors – efficient
    - Points most closest to hyperplane





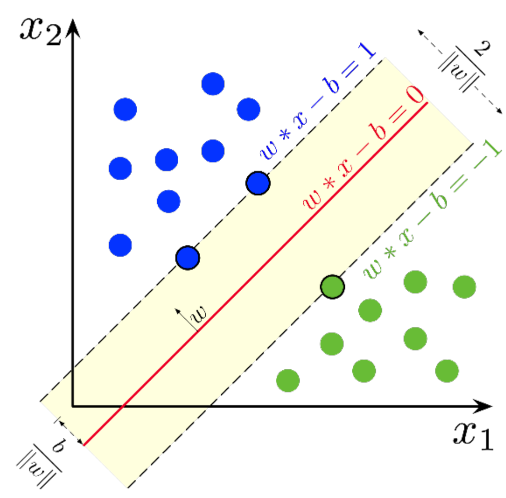
Some Math: Separating Hyperplanes

- Assume a set of training samples x_i with labels $y_i \subseteq \{-1,+1\}$
- Assume, for now, that the classes are linearly separable: There exists a hyperplane h such that all instances above h have label +1, all instances below h have label -1
- A hyperplane h can be characterized by ortho-normal vector w and a bias b: All points y with $<w*y> + b=0$
  - <> is the dot product, i.e. $\Sigma w_i * y_i$
- Thus, we seek a pair w,b such that $\forall i: y_i = sgn(<w*x_i>+b)$
- Problem: If one such hyperplane exists, then there are infinitely many (infinitesimally different)



Maximum Margin Classifier

- SVM seeks the one which maximizes the margin, i.e., find $h=(w,b)$ such that $m = \min_i |<w*x_i>+b|$ is maximal
  - Compute distance between h and all x_i; the minimal distance is the width of the margin; find h such that this margin m is maximal
- The margin is actually defined by two parallel hyperplanes h1, h2; one defined by the closest points with y=1, one by the closest negative points. Thus, there are x’ and x” with
  - h1: <w*x’>+b=1 and
  - h2: <w*x”>+b=-1



- The distance between h_1 and h_2 is $2/||w||$
- Thus: A minimal ||w|| gives maximal margin

Optimization Problem

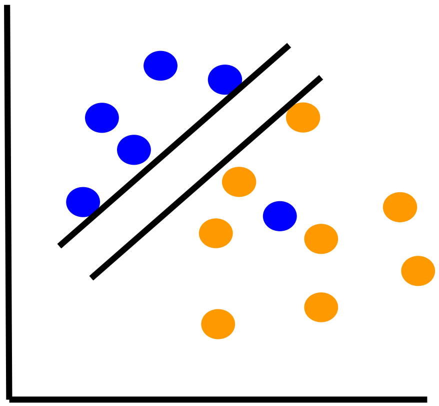
- Of course, h must also separate the two classes
- This gives the following constrained optimization problem

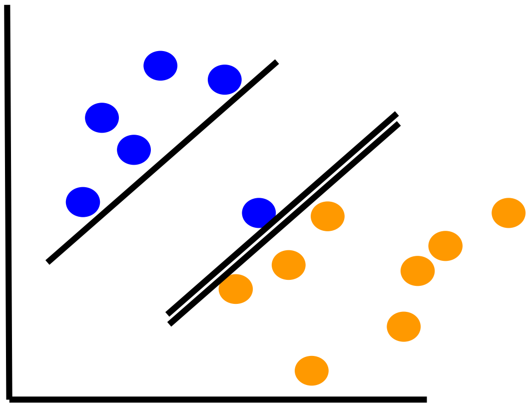
$$\text{Minimize } ||w|| \\ \text{under the constraint: } \forall i: \; y_i(<w*x_i>+b) \geq 1$$

- Not yet done: Most data sets are not linearly separable

Misclassification

- We need to account for instances that are misclassified
  - There should not be many, though
  - Might even be useful if the data set is linearly separable
  - We need a parameter defining how hard to "punish" training instances that are misclassified





Slack Variables

- For each training instance (x_i, y_i), we introduce a slack variable ξ_i which measures the error wrt the correct side of the hyperplane, i.e.: $\forall i: \; y_i(<w*x_i>+b) + \xi_i \geq 1$
  - Ideally, ξ_i is zero for all i
- New constraint: The sum of all errors ξ_i should be minimal
- New constrained optimization problem

$$\boxed{\begin{array}{c}\text{Minimize } ||w|| + C*\Sigma\xi_i \\[1em] \text{under the constraint: } \forall i: \; y_i(<w*x_i>+b)+\xi_i \geq 1\end{array}}$$

  - C controls the influence of misclassification: Large C leads to few misclassifications and small margins, small C lead to more misclassifications and larger margins

Solutions

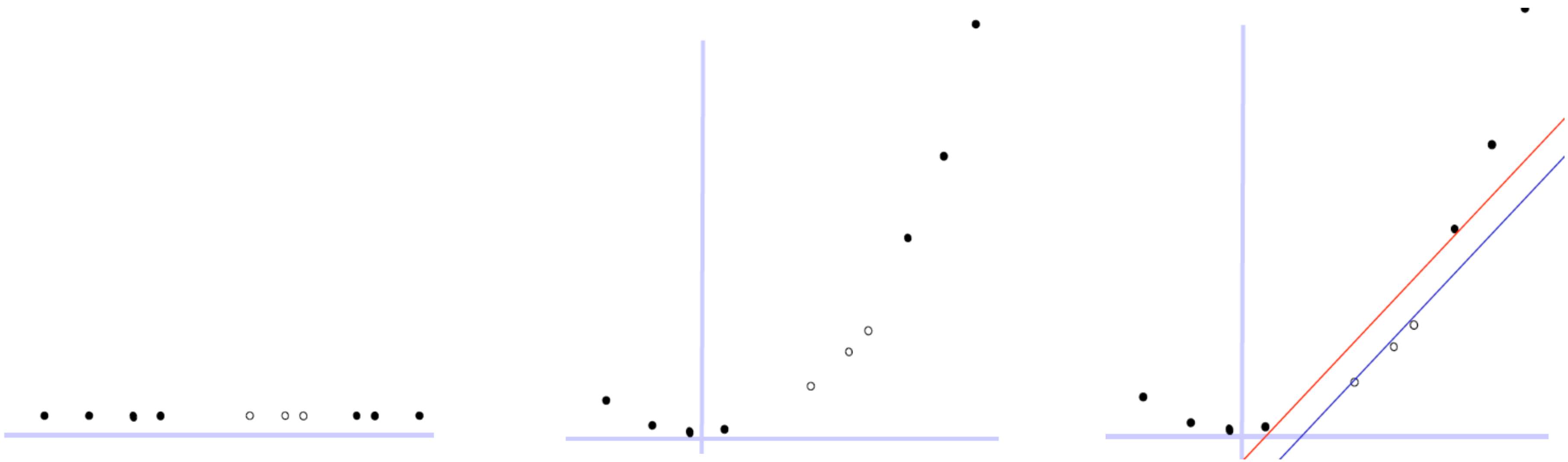
- Fortunately, this is a convex optimization problem and usually can be solved efficiently
  - But training with millions of dimensions and thousands of training instances still may take considerable time
- Classification (a new x) is fast: Compute sgn(<w*x>+b)

- "Support Vector" machine: The hyperplane only depends on the instances at the border of the margin; these are called "support vectors"
- History: Victor Vapnik, "The Nature of Statistical Learning Theory", 1995

Properties of SVM

- State-of-the-art in text classification (still?)
- Can cope with millions of dimensions
  - Might require long training time
- Classification is rather fast
- Quite robust to overfitting
- SVM are quite good “as is”, but tuning possible
  - Kernel function, biased margins, …
- Several free implementations exist: SVMlight, libSVM, …

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"

Content of this Lecture

- Classification
- Classification Methods
- Feature Selection
- Case studies
  - Topic classification
  - Competitive Evaluation (Seminar, 2017)
  - Spam filtering

Some ideas for features

- Classical standard: BoW
  - Every distinct token is a feature
- Classical alternatives
  - Remove stop words (no signal)
  - Remove rare words (too strong a signal)
  - Use bi-grams, tri-grams … (beware sentence breaks)
  - Perform part-of-speech tagging and keep only verbs and nouns
  - Perform shallow parsing and only keep noun phrases
  - Use noun phrases as additional features
  - Use different tokenizations at the same time
  - …
- Word2Vec: Represent words as distributions (later)

Feature Selection

- Features are redundant, correlated, irrelevant, …
- Many features bring much noise
  - Difficult to separate the signal from the noise
  - Most methods get slower with more features
- Traditional pre-processing step: Feature Selection
  - Goal: Reduce noise
  - Approach: Reduce set of all initial features to a smaller subset
  - Smaller models, easier to understand, faster classification

Types of FS methods

- Find a subset of features by …
- Wrapper methods
  - Find the best set of features by trying many subsets in CV
    - Requires an initialization and a search procedure
    - Very expensive / slow
- Embedded methods
  - Perform feature selection as part of model construction
- Filter methods
  - Score each feature and remove the bad ones

Filter Method: Mutual Information

- Mutual information: How much does the presence of a feature tell me about the class of a document?
- For each feature e_t, compute

$$\sum_{e \in \{0,1\}} \sum_{c \in \{0,1\}} p(e,c) * \log\left(\frac{p(e,c)}{p(e) * p(c)}\right)$$

  - e: Feature present or not (for binary features)
  - c: The two classes (for binary classification)
- Keep only features with highest MI

Filter Method: Chi-Square

- Chi-Square: Which features are significantly more often in one class than expected?
- For each feature e_t, compute

$$X^2 = \sum_{e \in \{0,1\}} \sum_{c \in \{0,1\}} \frac{(freq(e,c) - \exp(e,c))^2}{\exp(e,c)}$$

  - freq: Frequency of e in c (~p(e,c))
  - exp: Expected frequency of e in c assuming independence
  - Small X^2 values: Deviation from mean is significant, i.e., probably not created by chance
- Keep only features with highest significance

Unsupervised Feature

- Consider (all) pairs of features to identify redundant ones
  - Unsupervised: Disregard distribution of feature values over classes
- Simple approach: Pearson correlation

$$\frac{\frac{1}{n-1}\sum_{i=1}^{n}\left(e_{t,i}-\bar{e_t}\right)*\left(e_{s,i}-\bar{e_s}\right)}{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(e_{t,i}-\bar{e_t}\right)^2}*\sqrt{\frac{1}{n-1*}\sum_{i=1}^{n}\left(e_{s,i}-\bar{e_s}\right)^2}}$$

  - e_t, e_s are features, $\underline{e}$ is mean, n=|D|
  - Range [-1;1]; 0 means no (linear) correlation, -1/1 perfect (anti-)correlation
- When correlation is high, remove one (which one?)
- Value is independent of classes – “unsupervised”

Alternative: Feature Extraction

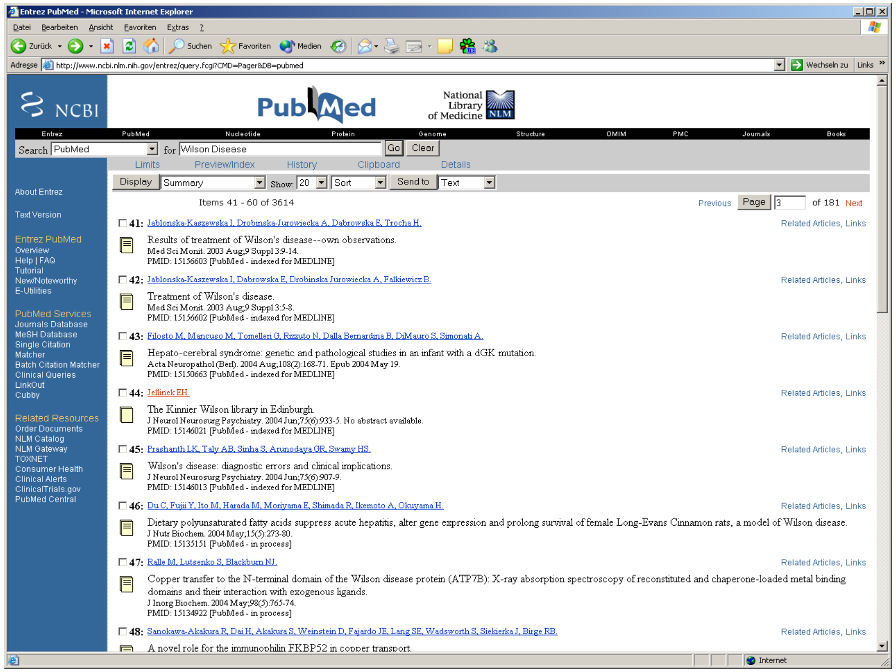
- Derive a set of new features by …
- Dimensionality reduction methods
  - Find a low-dimensional representation such that … (for instance)
  - Principal component analysis: Variance in data is preserved
  - Multidimensional scaling: Distances between points are preserved
  - …
- Note: Many classifiers compute “new” features by combining existing ones
  - Linear classifiers: Linear combinations of features
  - ANN: Non-linear combinations

Content of this Lecture

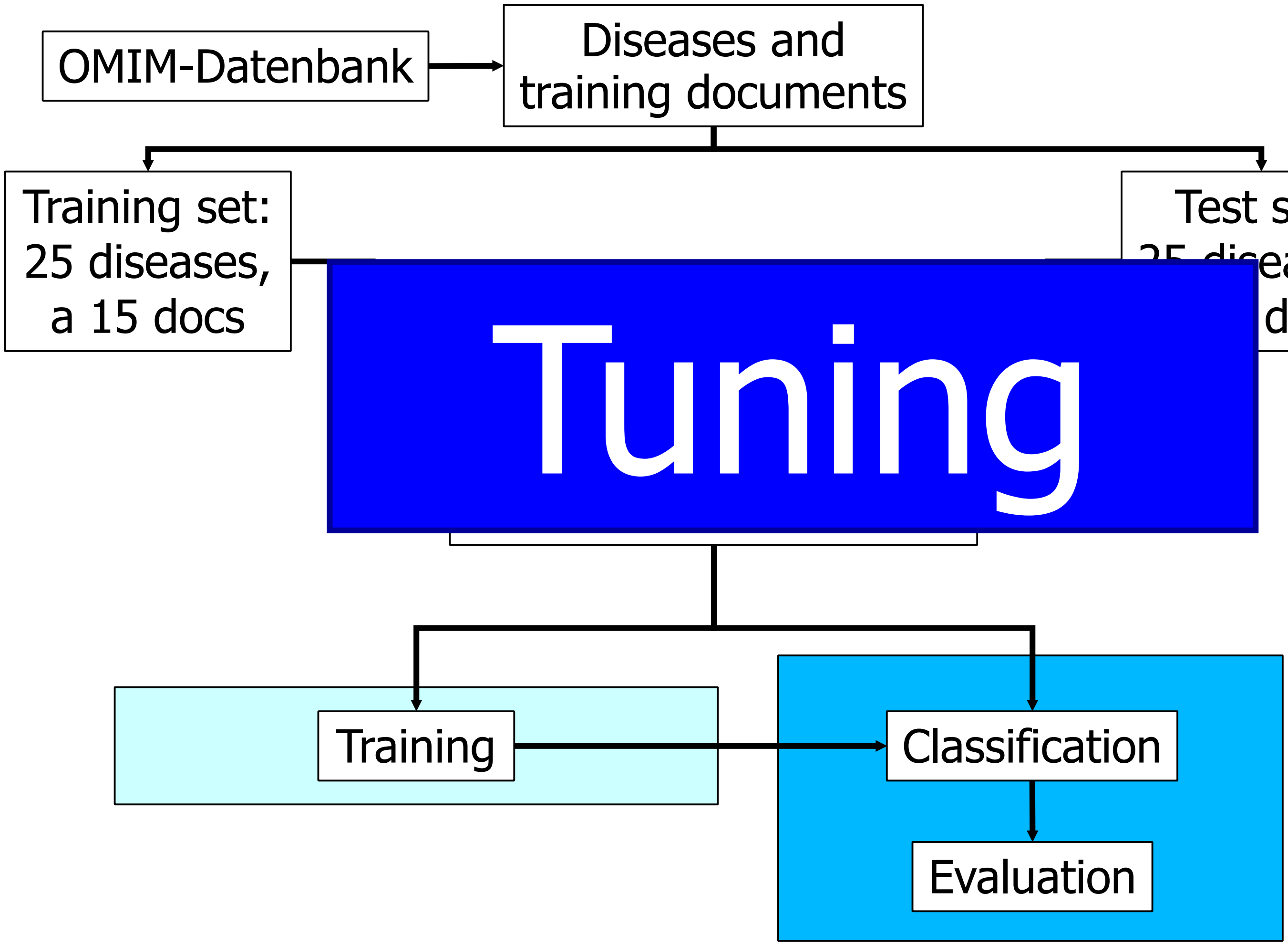
- Classification
- Classification Methods
- Feature Selection
- Case studies
  - Topic classification
  - Competitive Evaluation (Seminar, 2017)
  - Spam filtering

Topic Classification [Rutsch et al., 2005]

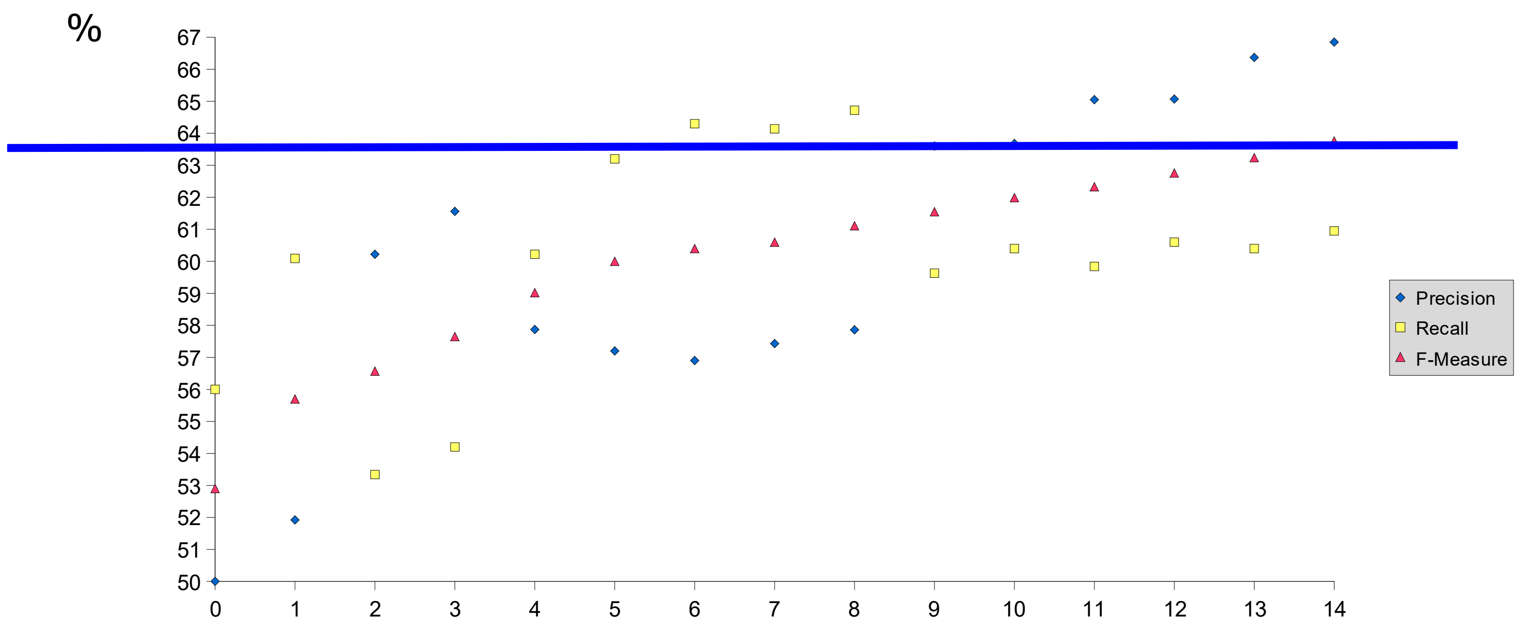
- Find publications treating the molecular basis of hereditary diseases
- Pure key word search generates too many results
  - "Asthma": 84 884 hits
    - Asthma and cats, factors inducing asthma, treatment, …
  - "Wilson disease": 4552 hits
    - Including all publications from doctors named Wilson
-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



Complete Workflow

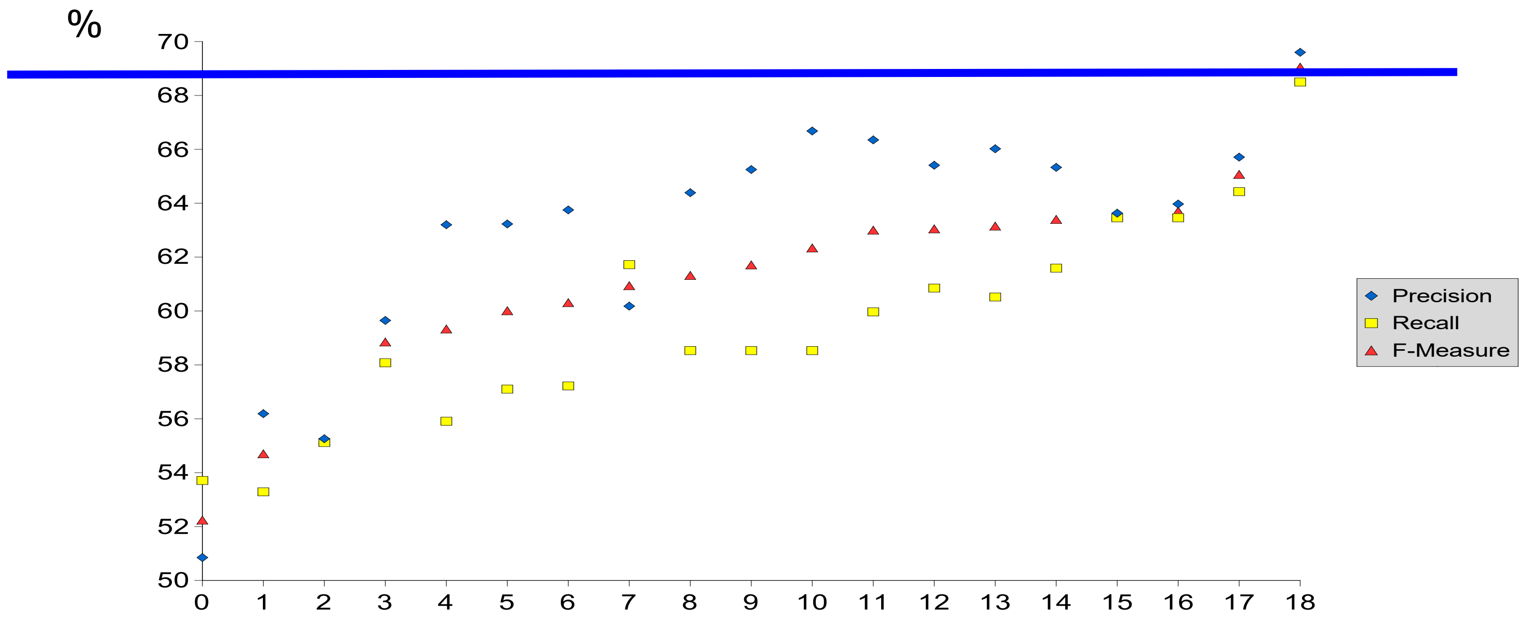


Results (Nearest-Centroid Classifier)



- 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
- Classification Methods
- Feature Selection
- Case studies
  - Topic classification
  - Competitive Evaluation (Seminar, 2017)
  - Spam filtering

Competition 2017

- Seminar „Text Classification"
- Six teams, each one method
  - RandomForest, Naive Bayes, SVM, kNN, ANN, logistic regression
- Two tasks: Binary / multiclass
  - Binary: Classify ~2000 docs in „cancer related" or not
  - Multiclass: Classify ~12000 docs according to 23 indications
    - Strong class imbalance
  - Setting: Training data, 3 months for experiments, release of unlabeled test data, each team max 2 submissions
- Entirely free: Implementation used, text preprocessing, parameter tuning …

Results

	Random Forests	SVM	k-Nearest Neighbors	Naive Bayes	Neural Networks
Mit welchen Arten von Features haben Sie experimentiert (z.B. Bag-of-Words, TFIDF, Word Embeddings, ...)	bag of words tfidf ngrams auf char (3-7) und word (1-2) level	BoW, Word/Char n-grams, TF-IDF, Word Embeddings, Titel auf MeSH-Terms untersuchen	Bag-of-Words, N-Grams auf Zeichen- und Tokenebene, TF-IDF, LSA Topic Modelling, Word Embedding)	Bag-of-Words, 2- bis 4-Gramme, Noun Phrases	TF-IDF und Word Embeddings
Welche haben sich bewährt?	tfidf	BoW, TF-IDF, Word Embeddings	TF-IDF, SVD, N-Grams auf Tokenebene	Bag-of-Words, 2- bis 4-Gramme (Noun Phrases haben keine Rolle gespielt)	TF-IDF für das Binäre Problem, WE für Multiclass
Was war die Gesamtzahl Feature in ihrer finalen Konfiguration für die Challenge?	141 000 und 358 000	106490	90	für binary 10000	Binär 4100, Multiclass 200
Haben Sie explizite Feature Selection durchgeführt? Wenn ja - wie?	Chi2	Max. document frequency, min. df, Chi2 test	SVD, LSA Topic Modelling, min_df, max_df	Chi2	Corpusspezifische Stopwords

	Random Forests	SVM	k-Nearest Neighbors	Naive Bayes	Neural Networks	Logistic Regression
Wie wurde gestemmt	Lemma mit Wordnet	Kein Stemming	kein Stemming	Kein Stemming	Kein Stemming	
Wordsemantik? (embeddings, Disambig.)	Synsets aus Wordnet	Word Embeddings	Word Embeddings (Wikipedia)	Speziele bio-Terme mit speziellen DBs	PubMed + PMC Word Embeddings	
Laufzeit Training / Classification	Sekunden	Bis zu einer Stunde	Wenige Minuten	Wenige Minuten	Wenige Minuten	
Tools / libraries	Python, nltk scikitLearn	NLTK, GenSim, Scikitlearn	NLTK, pandas, Gensim	scikit learn , esmre (regexp)	Keras, Gensim, NLTK,	
Überraschendstes Ergebnis	Keine	schlechte Ergebnisse bei Polynomial- oder RBF-Kernels	Accuracy in MC-Competition viel schlechter als bei CV	Schlechte bin Clas. ((overfitting mit 10000 Features?). 80/20 Regel	starke Einfluss der Architektur auf das Multiclass Problem	
Ergebnis Binary	0,958	0,942	0,963	0,931	0,958	0,947
Rank Binary	3	5	1	6	2	4
Ergebnis Multiclass	0,321	0,426	0,395	0,434	0,483	0,467
Rank Multiclass	6	4	5	3	1	2

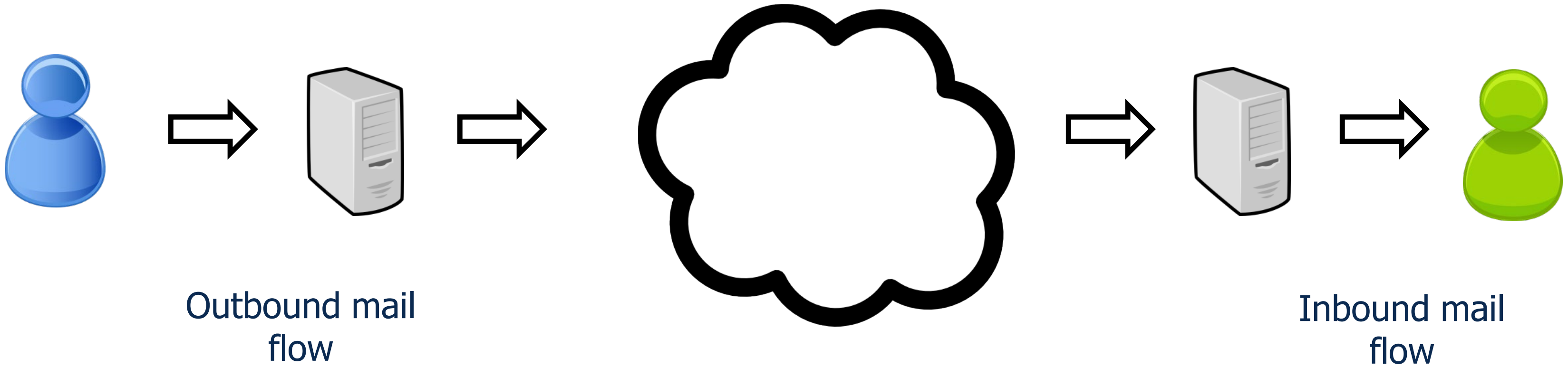
Content of this Lecture

- Classification
- Classification Methods
- Feature Selection
- Case studies
  - Topic classification
  - Competitive Evaluation (Seminar, 2017)
  - 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

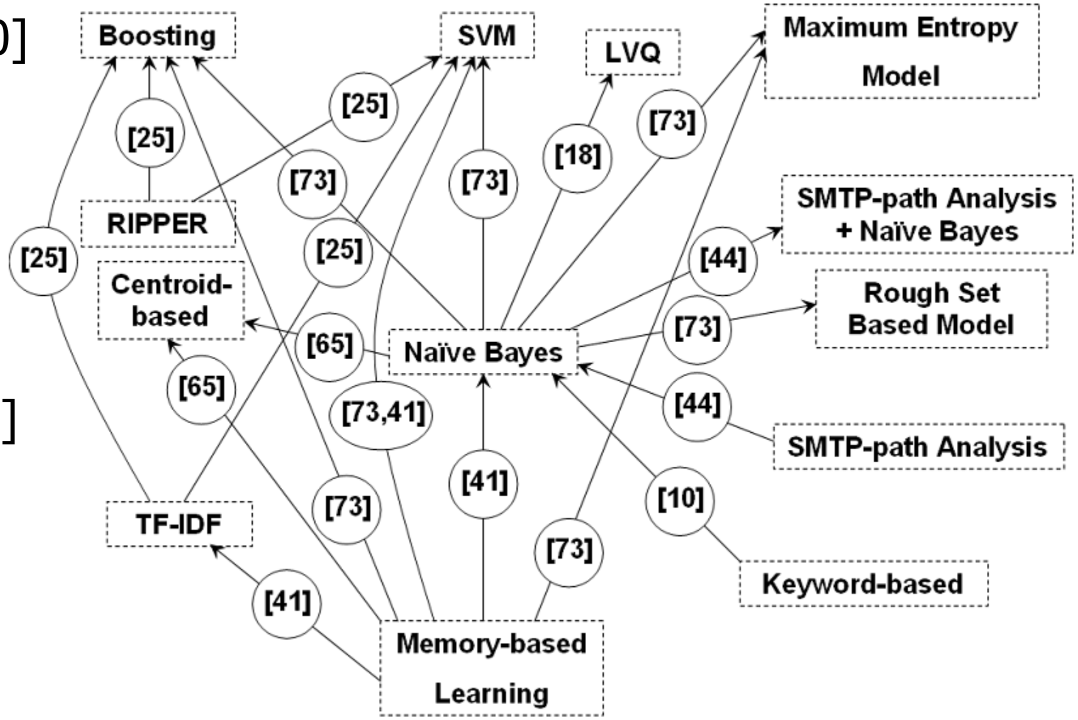


SPAM Detection as a Classification Task

- 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

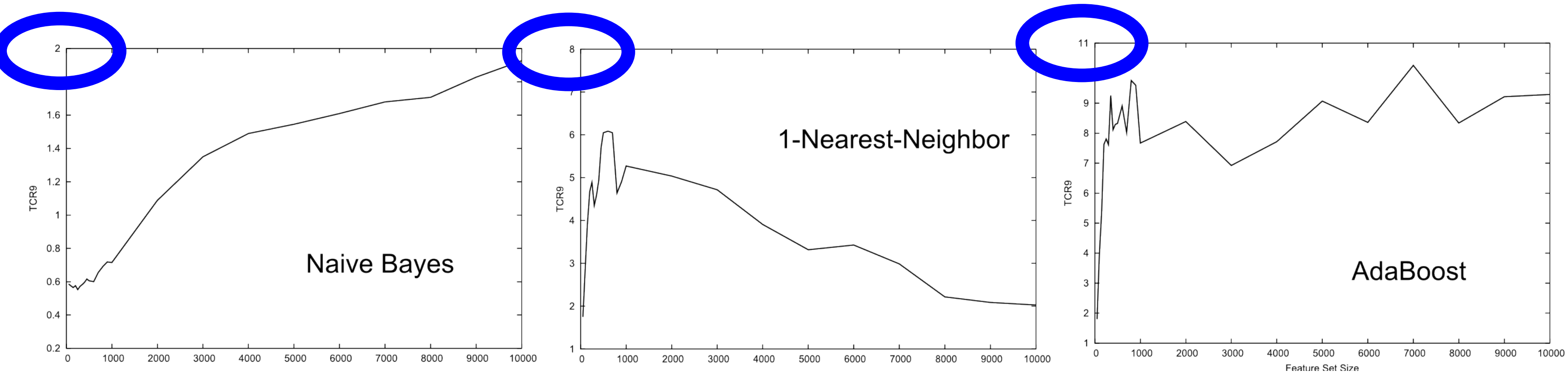
- Rule learning
  [Cohen, 1996]
- k-Nearest-Neighbors
  [Androutsopoulos *et al.*, 2000]
- SVM
  [Kolcz/Alspector, 2001]
- Decision trees
  [Carreras/Marquez, 2001]
- Centroid-based
  [Soonthornphisaj et al., 2002]
- Artificial Neural Networks
  [Clark *et al.*, 2003]
- Logistic regression
  [Goodman/Yih,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

Literature

- Manning / Schütze: Foundations of Statistical Natural Language Processing
- Kelleher, MacNamee, D‘Arcy: Machine Learning for Predictive Data Analysis

Self-Test

- Enumerate different methods for text classification and describe the general framework (supervised learning)
- Describe the Maximum Entropy (NB, kNN, …) method. What role does Iterative Scaling have? Where does "maximum entropy" come into play?
- What is Gaussian Naïve Bayes? Does it have a higher classification complexity than Multinomial Naïve Bayes?
- Describe the Chi^2 feature selection method. On what assumptions is it built?
- Assume the following data: … Build a Naïve Bayes Model and predict the class of the unlabeled instance