Text Classification

- Characteristics of Machine Learning Problems
- Text Classification Algorithms
  - k nearest-neighbor algorithm, Rocchio algorithm
  - naïve Bayes classifier
  - Support Vector Machines
  - decision tree and rule learning
- Occam's Razor and Overfitting Avoidance
- Evaluation of classifiers
  - evaluation metrics
  - cross-validation
  - micro- and macro-averaging
- Multi-Label Classification

Type of Training Information

- Supervised Learning:
  - A „teacher“ provides the value for the target function for all training examples (labeled examples)
  - concept learning, classification, regression
- Semi-supervised Learning:
  - Only a subset of the training examples are labeled (labeling examples is expensive!)
- Reinforcement Learning:
  - A teacher provides feedback about the values of the target function chosen by the learner
- Unsupervised Learning:
  - There is no information except the training examples
  - clustering, subgroup discovery, association rule discovery

Example Availability

- Batch Learning
  - The learner is provided with a set of training examples
- Incremental Learning / On-line Learning
  - There is constant stream of training examples
- Active Learning
  - The learner may choose an example and ask the teacher for the relevant training information

Document Representation

- The vector space models allows to transform a text into a document-term table

- In the simplest case
  - Rows:
    - training documents
  - Columns:
    - words in the training documents
  - More complex representation possible

- Most machine learning and data mining algorithms need this type of representation
  - they can now be applied to, e.g., text classification

Example Representation

- Attribute-Value data:
  - Each example is described with values for a fixed number of attributes
    - **Nominal Attributes:**
      - store an unordered list of symbols (e.g., *color*)
    - **Numeric Attributes:**
      - store a number (e.g., *income*)
    - **Other Types:**
      - hierarchical attributes
      - set-valued attributes
  - the data corresponds to a single relation (spreadsheed)
- Multi-Relational data:
  - The relevant information is distributed over multiple relations
    - e.g., `contains_word(Page,Word), linked_to(Page,Page),...`

Bag-of-Words vs. Set-of Words

- **Set-of-Words:** boolean features
  each dimension encodes wether the feature appears in the document or not

- **Bag-of-words:** numeric features
  each dimension encodes how often the feature occurs in the document (possibly normalized)

- Which one is preferable depends on the task and the classifier

Concept Representation

- Most Learners generalize the training examples into an explicit representation
  (called a model, function, hypothesis, concept...)
  - mathematical functions (e.g., polynomial of 3rd degree)
  - logical formulas (e.g., propositional IF-THEN rules)
  - decision trees
  - neural networks

  ....

- Lazy Learning
  - do not compute an explicit model
  - generalize „on demand“ for an example
  - e.g., nearest neighbor classification

A Selection of Learning Techniques

- Decision and Regression Trees
- Classification Rules
- Association Rules
- Inductive Logic Programming
- Neural Networks
- Support Vector Machines
- Statistical Modeling
- Clustering Techniques
- Case-Based Reasoning
- Genetic Algorithms
- ....

Induction of Classifiers

The most „popular“ learning problem:

- Task:
  - learn a model that predicts the outcome of a dependent variable for a given instance
- Experience:
  - experience is given in the form of a data base of examples
  - an example describes a single previous observation
    - *instance:* a set of measurements that characterize a situation
    - *label:* the outcome that was observed in this siutation
- Performance Measure:
  - compare the predicted outcome to the observed outcome
  - estimate the probability of predicting the right outcome in a new situation

Text Classification: Examples

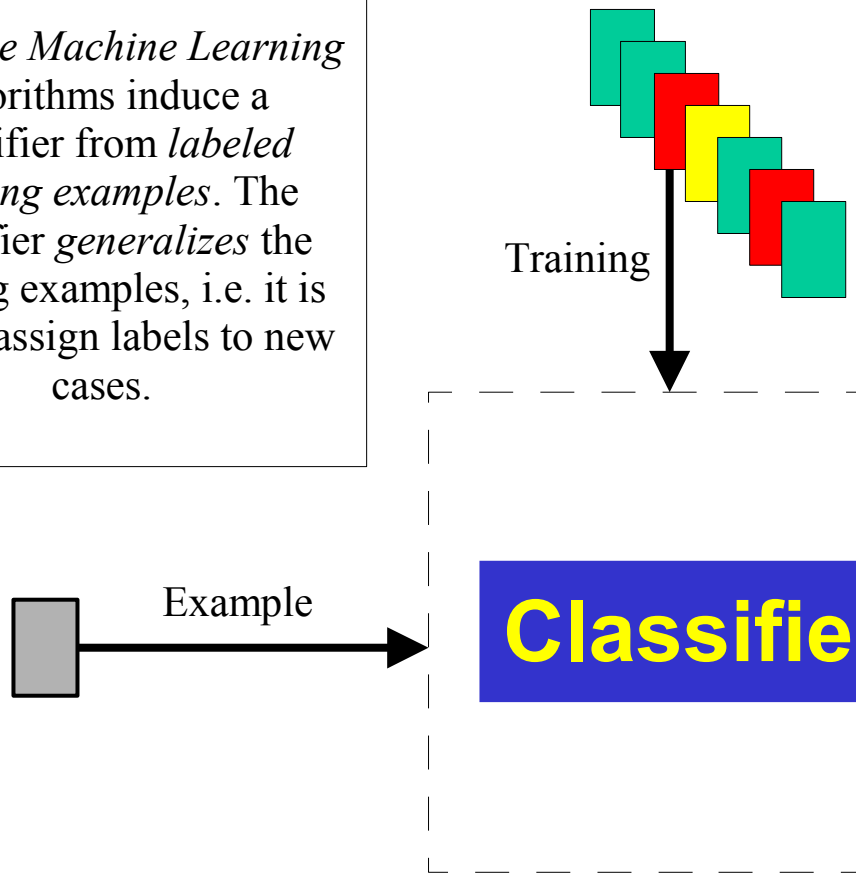
Text Categorization: Assign labels to each document

- Labels are most often topics such as Yahoo-categories
  - *e.g., "finance," "sports," "news::world::asia::business"*
- Labels may be genres
  - *e.g., "editorials" "movie-reviews" "news"*
- Labels may be opinion
  - e.g., *"like", "hate", "neutral"*
- Labels may be binary concepts
  - *e.g., "interesting-to-me" : "not-interesting-to-me"*
  - e.g., *"spam" : "not-spam"*
  - e.g., *"contains adult language" :"doesn't"*

More than one learning task could be defined over the same documents

Induction of Classifiers

Inductive Machine Learning algorithms induce a classifier from *labeled training examples*. The classifier *generalizes* the training examples, i.e. it is able to assign labels to new cases.



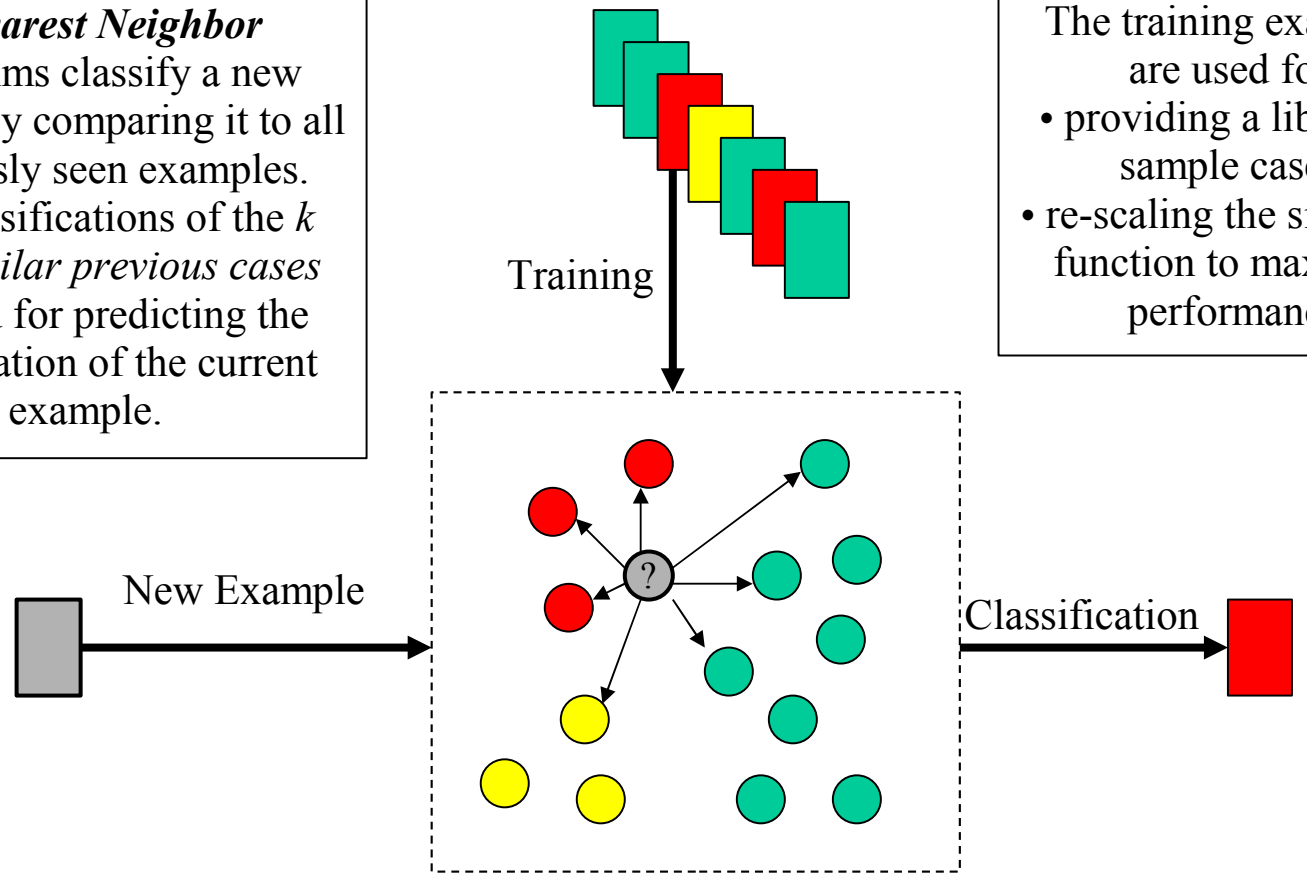
An inductive learning algorithm searches in a given family of hypotheses (e.g., *decision trees, neural networks*) for a member that optimizes given *quality criteria* (e.g., estimated predictive accuracy or misclassification costs).

Induction of Classifiers

- Typical Characteristics
  - attribute-value representation (single relation)
  - batch learning from off-line data (data are available from external sources)
  - supervised learning (examples are pre-classified)
  - numerous learning algorithms for practically all concept representations (decision trees, rules, neural networks, SVMs, statistical models,...)
  - often greedy algorithms (fast processing of large datasets)
  - evaluation by estimating predictive accuracy (on a portion of the available data)

Nearest Neighbor Classifier

K-Nearest Neighbor algorithms classify a new example by comparing it to all previously seen examples. The classifications of the *k most similar previous cases* are used for predicting the classification of the current example.



The training examples are used for
- providing a library of sample cases
- re-scaling the similarity function to maximize performance

kNN Classifier

- To learn from a training set:
  - Store the training set
- To classify a new document :
  - Compute similarity of document vector $\mathbf{q}$ with all available document vectors D (e.g., using cosine similarity)
  - Select the k nearest neighbors (hence the name k-NN)
  - Combine their classifications to a new prediction (e.g., majority, weighted majority,...)
- "Lazy" learning or local learning
  - because no global model is built
  - generalization only happens when it is needed

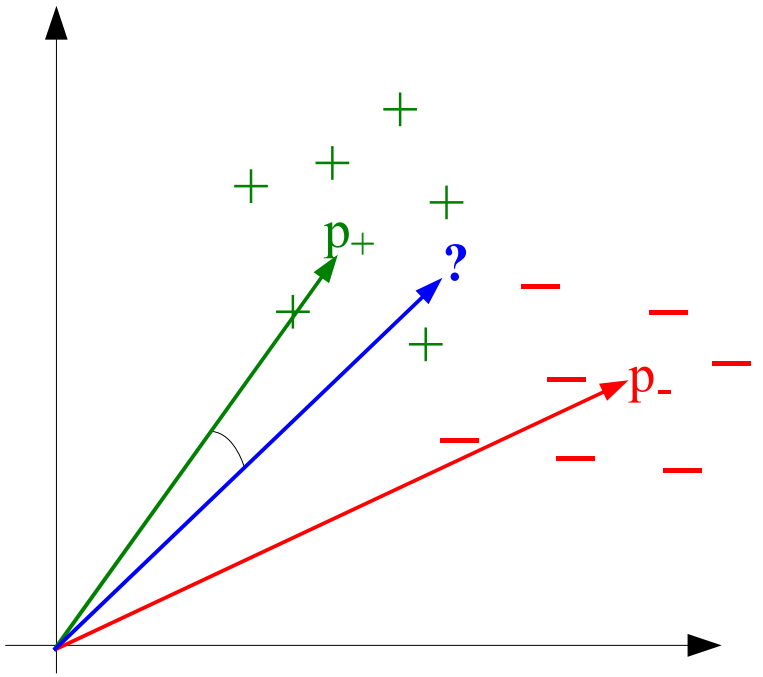
Nearest Neighbor with Inverted Index

- Naively finding nearest neighbors requires comparing the test document $\mathbf{q}$ to $|D|$ documents in collection ($O(|D|)$)
- But determining *k* nearest neighbors is the same as determining the *k* best retrievals using the test document as a query to a database of training documents.
- Use standard vector space inverted index methods to find the *k* nearest neighbors
  - retrieve all documents containing at least one of the words in the query document and rank them
- Testing Time: $O(B \cdot |\mathbf{q}|)$
  - where B is the average number of training documents in which a query-document word appears.
  - Typically $B << |D|$

Rocchio Classifier

- based on ideas for Rocchio Relevance Feedback
- compute a prototype vector $\mathbf{p}_c$ for each class c
  - average the document vectors for each class
- classify a new document according to distance to prototype vectors instead of documents

- assumption:
  - documents that belong to the same class are close to each other (form one cluster)

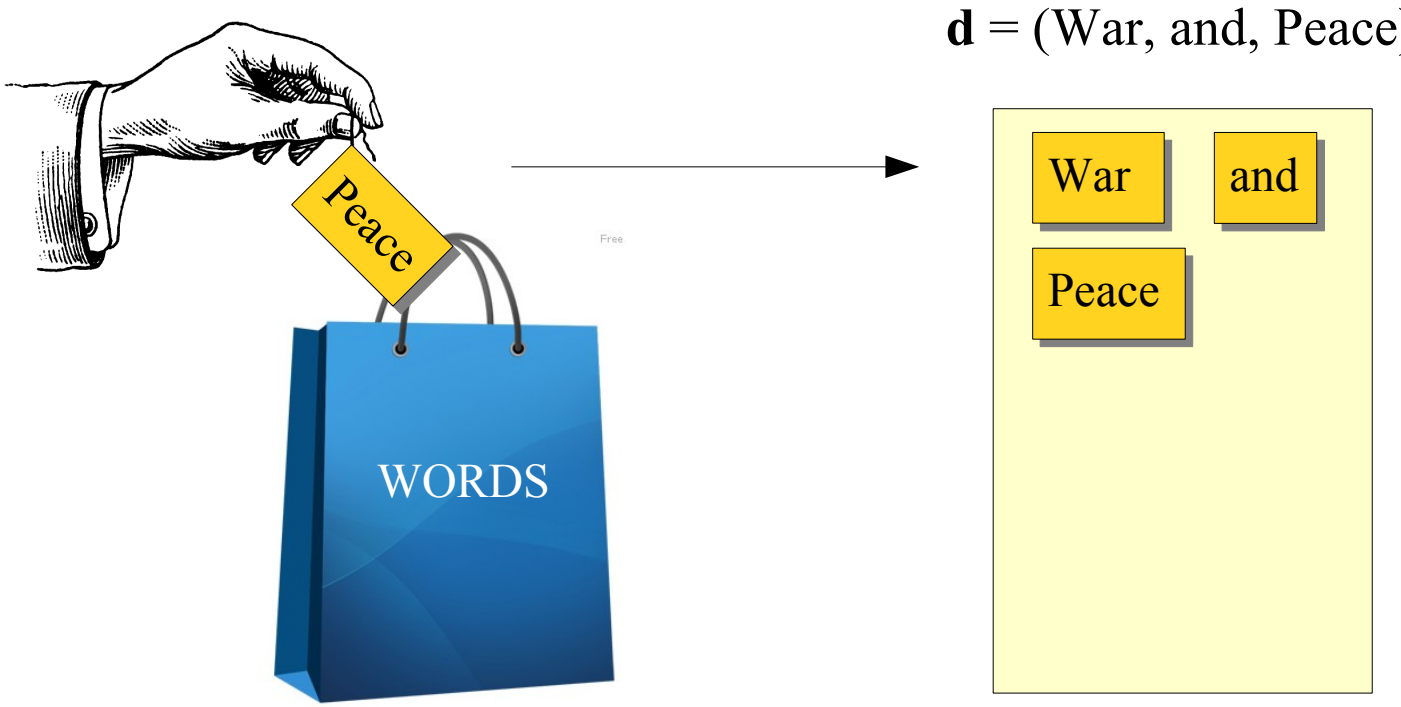


Bag of Words Model

- assumes that the document has been generated by repeatedly drawing one word out of a bag of words
  - like drawing letters out of a Scrabble-bag, but with replacement
- words in the bag may occur multiple times, some more frequently than others
  - like letters in a Scrabble-bag
  - each word w is drawn with a different probability $p(w)$

Probabilistic Document Model

- Repeatedly drawing from the bag of words results in a sequence of randomly drawn words → a document
  - $\mathbf{d} = (t_1, t_2, \ldots, t_{|\mathbf{d}|})$ where $t_j = w_{k_j} \in W$



Class-conditional Probabilities

- Different classes have different bags of words

Sports

Business

Politics

- probabilities of words in different classes are different
  - the sports bag contains more sports words, etc.
  - Formally: $p(w|c_i) \neq p(w|c_j) \neq p(w)$

Independence Assumption

- the probability that a word occurs does not depend on the context (the occurrence or not-occurrence of other words)
  - it only depends on the class of the document
- In other words:
  - Knowing the previous word in the document (or any other word) does not change the probability that a word occurs in position t_i

$$p(t_i = w_{k_i} | t_j = w_{k_j}, c) = p(t_i = w_{k_i} | c)$$

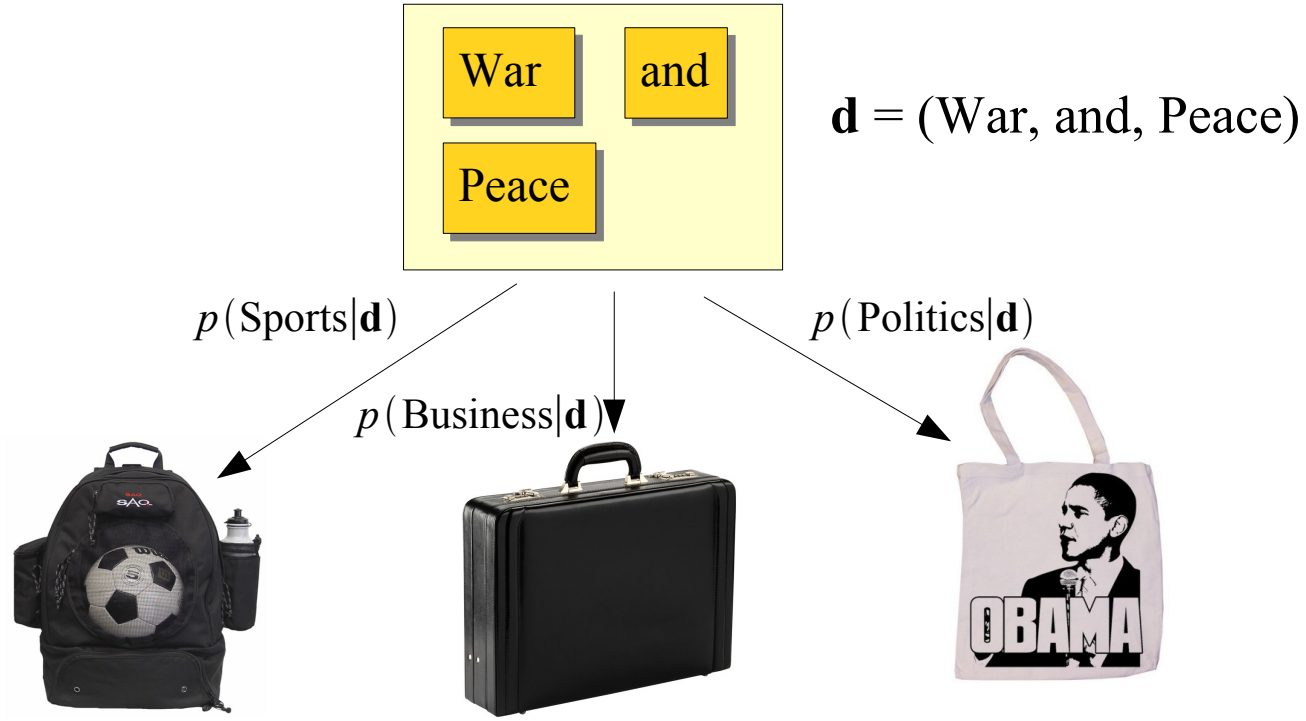
  we will write this shorter as

$$p(t_i | t_j, c) = p(t_i | c)$$

- Important:
  - the independence assumption does not hold in real texts!
  - but it turns out that it can still be used in practice

Probabilistic Text Classification

- Answer the question:
  - From which bag was a given document **d** generated?



- Answer is found by estimating the probabilities $p(c|\mathbf{d})$

Probabilistic Document Model

- A document is a sequence of words (tokens, terms, features...)
  - $\mathbf{d} = (t_1, t_2, \ldots, t_{|\mathbf{d}|})$ where $t_j = w_{i_j} \in W$
  - Assume that a document $\mathbf{d}$ has been generated by repeatedly selecting a word w_{ij} at random
- The probability that a word occurs in a document is dependent on the document's class c
  - $p(t_i|c) \neq p(t_i)$
- (Class-Conditional) **Independence Assumption:** The occurrence of a word in a class is independent of its context
  - $p(t_i|t_j, c) = p(t_i|c)$
- Goal of Classification:
  - Determine the probability $p(c|\mathbf{d})$ that document $\mathbf{d}$ belongs to class c

Bayesian Classification

- Maximum a posteriori classification
  - predict the class c that has the highest probability given the document D

$$c = \arg\max_c p(c|\mathbf{d})$$

  - Problem:
    - we have not seen the document often enough to directly estimate $p(c|\mathbf{d})$
- Bayes Theorem: $p(c|\mathbf{d}) \cdot p(\mathbf{d}) = p(\mathbf{d}|c) \cdot p(c)$
  - equivalently $p(c|\mathbf{d}) = \frac{p(\mathbf{d}|c)\, p(c)}{p(\mathbf{d})}$
- $p(\mathbf{d})$ is only for normalization: $p(\mathbf{d}) = \sum_c p(\mathbf{d}|c)\, p(c)$
  - can be omitted if we only need a ranking of the class and not a probability estimate
- Bayes Classifier:

$$c = \arg\max_c \; p(\mathbf{d}|c)\, p(c)$$

If all prior probabilites $p(c)$ are identical → maximum likelihood prediction

Simple Naïve Bayes Classifier for Text (Mitchell 1997)

- a document is a sequence of n terms

$$p(\mathbf{d}|c) = p(t_1, t_2, ... t_n|c)$$

- Apply Independence Assumption:
  - $p(t_i|c)$ is the probability with which the word $t_i = w_{i_j}$ occurs in documents of class c

$$p(\mathbf{d}|c) = \prod_{i=1}^{|\mathbf{d}|} p(t_i|c)$$

- Naïve Bayes Classifier
  - putting things together:

$$c = \arg\max_c \prod_{i=1}^{|\mathbf{d}|} p(t_i|c)\, p(c)$$

Estimating Probabilities (1)

- Estimate for prior class probability $p(c)$
  - fraction of documents that are of class c
- Word probabilities can be estimated from data
  - $p(t_i|c)$ denotes probability that term $t_i = w_{i_j} \in W$ occurs at a certain position in the document
    - assumption: probability of occurrence is independent of position in text
  - estimated from fraction of document positions in each class on which the term occurs
    - put all documents of class c into a single (virtual) document
    - compute the frequencies of the words in this document

Estimating Probabilities (2)

- Straight-forward approach:
  - estimate probabilities from the frequencies in the training set
  - word w occurs $n(\mathbf{d},w)$ times in document $\mathbf{d}$

$$p(t_i=w|c)=\frac{n_{w,c}}{\sum_{w\in W} n_{w,c}}$$

$$n_{w,c}=\sum_{\mathbf{d}\in c} n(\mathbf{d},w)$$

- Problem:
  - test documents may contain new words
  - those will be have estimated probabilities 0
  - assigned probability 0 for all classes
- Smoothing of probabilities:
  - basic idea: assume a prior distribution on word probabilities
  - e.g., Laplace correction assumes each word occurs at least once in a document

$$p(t_i=w|c)=\frac{n_{w,c}+1}{\sum_{w\in W}(n_{w,c}+1)}=\frac{n_{w,c}+1}{\sum_{w\in W} n_{w,c}+|W|}$$

Full Multinomial Model

Two basic shortcomings of the simple Naïve Bayes:

- If we consider the document as a „bag of words“, many sequences correspond to the same bag of words
  - better estimate:

$$p(\mathbf{d}|c) = \binom{|\mathbf{d}|}{\{n(\mathbf{d},w)_{w\in\mathbf{d}}\}} \prod_{w\in\mathbf{d}} p(w|c)^{n(\mathbf{d},w)}$$

$$\binom{n}{i_1, i_2, \ldots i_k} = \frac{n!}{i_1! \cdot i_2! \cdot \ldots \cdot i_k!}$$

$\prod_{w\in\mathbf{d}}$ iterates over vocabulary

$\prod_{i=1\ldots|\mathbf{d}|}$ iterates over document positions

- we assumed that all documents have the same length
  - a better model will also include the document length $l = |\mathbf{d}|$ conditional on the class

$$p(\mathbf{d}|c) = p(l=|\mathbf{d}|\,|\,c) \binom{|\mathbf{d}|}{\{n(\mathbf{d},w)_{w\in\mathbf{d}}\}} \prod_{w\in\mathbf{d}} p(w|c)^{n(\mathbf{d},w)}$$

  - $p(l=|\mathbf{d}|\,|\,c)$ may be hard to estimate

Binary Model

- a document is represented as a *set of words*
  - model does not take into account document length or word frequencies
  - aka Multi-variate Bernoulli Model
- in this case $p(w|c)$ indicates the probability that a document in class c will mention term w at least once.
  - estimated by fraction of documents in each class in which the term occurs
- the probability of seeing document $\mathbf{d}$ in class c is
  - the product of probabilities for all words occurring in the document
  - times the product of the counter-probabilities of the words that do not occur in the document

$$p(\mathbf{d}|c) = \prod_{t \in \mathbf{d}} p(t|c) \prod_{t \in W, t \notin \mathbf{d}} (1 - p(t|c)) = \prod_{t \in \mathbf{d}} \frac{p(t|c)}{1 - p(t|c)} \underbrace{\prod_{t \in W} (1 - p(t|c))}_{\text{to account for } t \notin \mathbf{d}}$$

Numerics of Naïve Bayes Models

- We need to multiply a large number of small probabilities,
  - Result: extremely small probabilities as answers.
  - Solution: store all numbers as logarithms

$$c = \arg\max_c p(c)\prod_{i=1}^{|\mathbf{d}|} p(t_i|c) = \arg\max_c \underbrace{\left(\log(p(c)) + \sum_{i=1}^{|\mathbf{d}|} \log(p(t_i|c))\right)}_{l_c}$$

  - to get back to the probabilities:

$$p(c|\mathbf{d}) = \frac{e^{l_c}}{\sum_{c'} e^{l_{c'}}} = \frac{1}{1+\sum_{c' \neq c} e^{l_{c'}-l_c}}$$

- Class which comes out at the top wins by a huge margin
  - Sanitizing scores using likelihood ratio LR
    - Also called the logit function

$$\text{logit}(\mathbf{d}) = \frac{1}{1+e^{-LR(\mathbf{d})}}, \quad LR(\mathbf{d}) = \frac{p(c|\mathbf{d})}{1-p(c|\mathbf{d})}$$

Rainbow (McCallum)

- advanced implementation of a Naïve Bayes text classifier with numerous options
  - http://www.cs.umass.edu/~mccallum/bow/rainbow/

Performance analysis

- Multinomial naive Bayes classifier generally outperforms the binary variant
  - but the binary model is better with smaller vocabulary sizes

- K-NN may outperform Naïve Bayes
  - Naïve Bayes is faster and more compact

Graphs taken from Andrew McCallum and Kamal Nigam:A Comparison of Event Models for Naive Bayes Text Classification. AAAI-98 Workshop on "Learning for Text Categorization". http://www.cs.umass.edu/~mccallum/papers/multinomial-aaai98w.ps

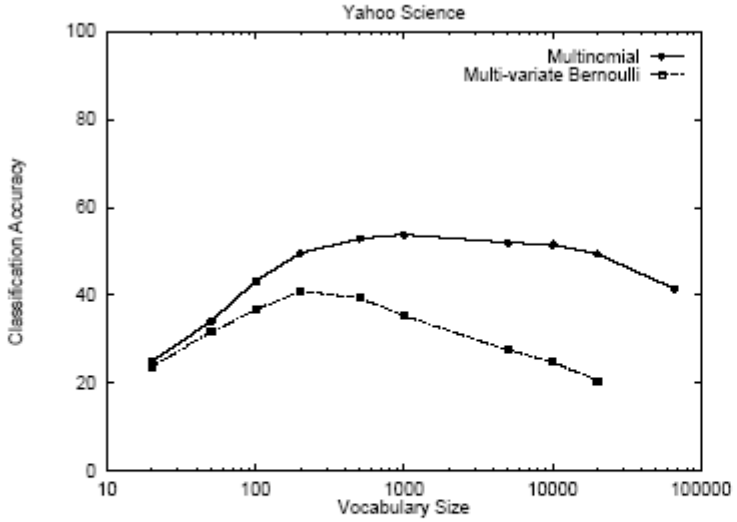


Figure 1: A comparison of event models for different vocabulary sizes on the Yahoo data set. Note that the multi-variate Bernoulli performs best with a small vocabulary and that the multinomial performs best with a larger vocabulary. The multinomial achieves higher accuracy overall.

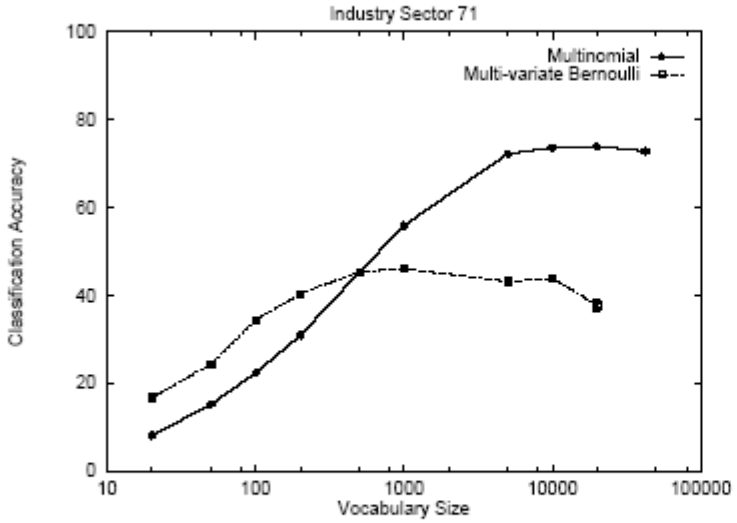


Figure 2: A comparison of event models for different vocabulary sizes on the Industry Sector data set. Note the same trends as seen in the previous figure.

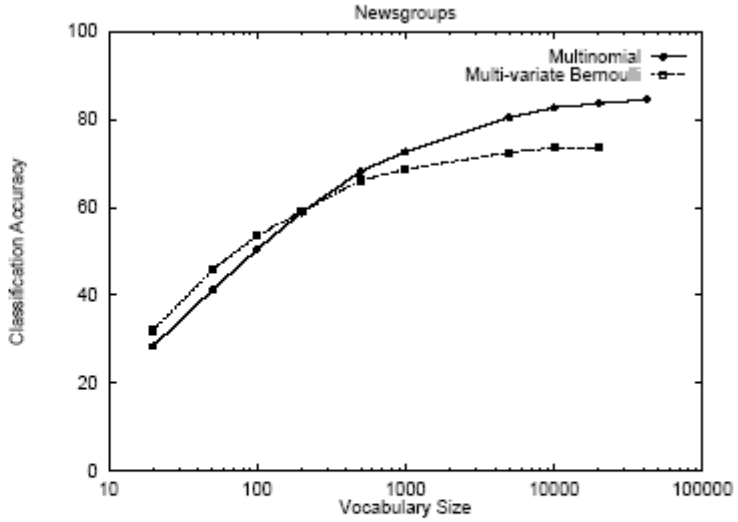


Figure 3: A comparison of event models for different vocabulary sizes on the Newsgroups data set. Here, both data sets perform best at the full vocabulary, but multinomial achieves higher accuracy.

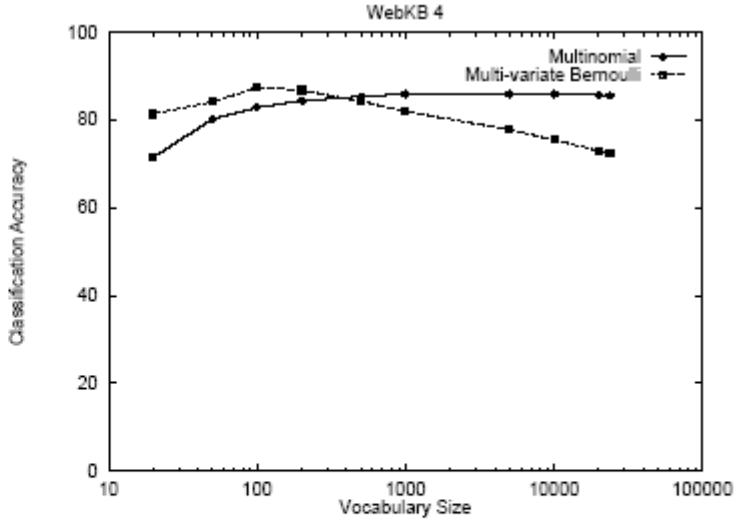


Figure 4: A comparison of event models for different vocabulary sizes on the WebKB data set. Here the two event models achieve nearly equivalent accuracies, but the multi-variate Bernoulli achieves this with a smaller vocabulary.

NB: Decision boundaries

- Bayesian classier partitions the multidimensional term space into regions
  - Within each region, the probability of one class is higher than others
  - On the boundaries, the probability of two or more classes are exactly equal

- 2-class NB has a linear decision boundary
  - easy to see in the logarithmic representation of the multinomial version

$$\log(p(\mathbf{d}|c)) = \log\left(\begin{matrix}|\mathbf{d}|\\ \{n(\mathbf{d},w)_{w\in\mathbf{d}}\}\end{matrix}\right) + \sum_{w\in\mathbf{d}} n(\mathbf{d},w)\cdot\log p(w|c) = b + \mathbf{d}\cdot\mathbf{w}_{NB}$$

$\mathbf{w}_{NB}$ weight vector: weight of word w is $\log(p(w|c))$

$\mathbf{d}$ document vector consisting of term frequencies $n(\mathbf{d},w)$

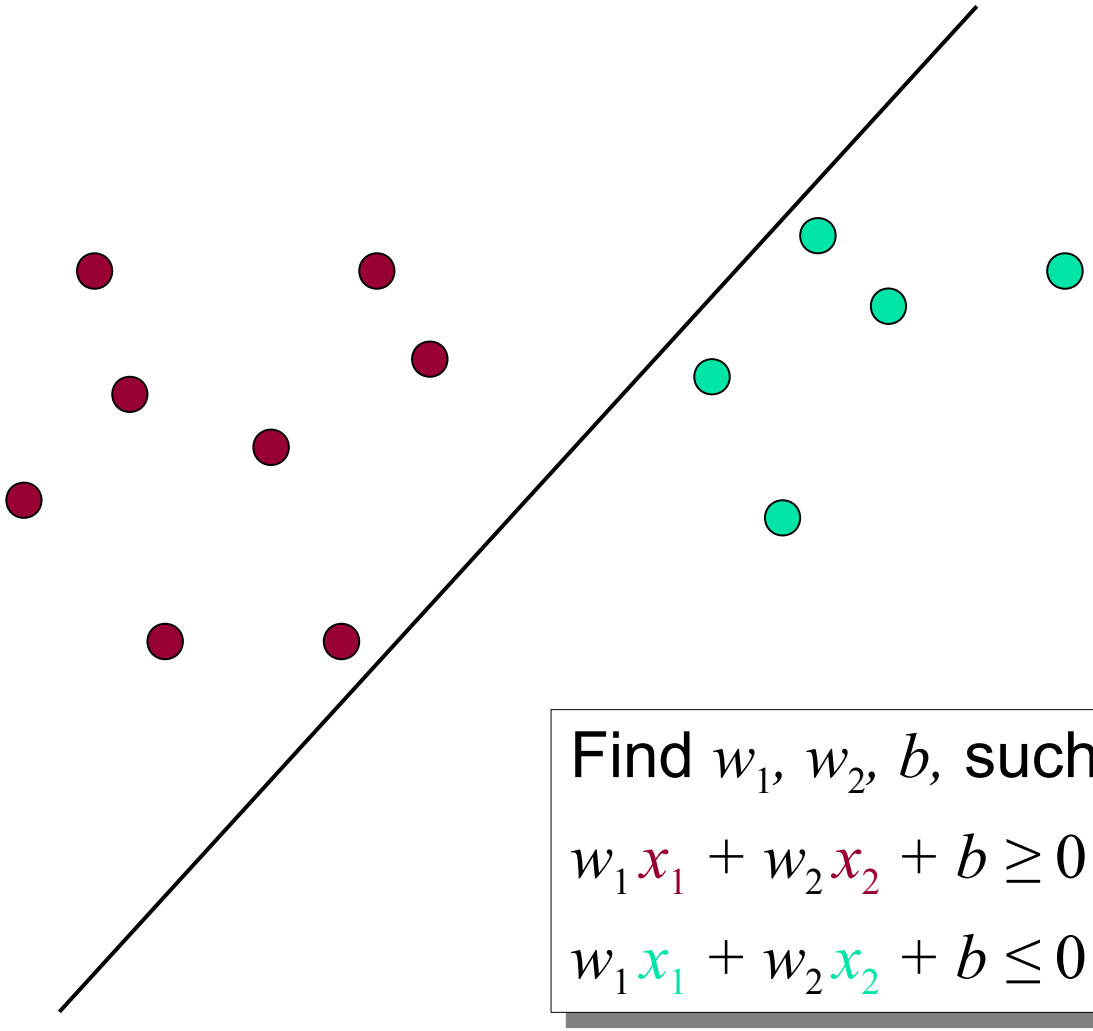
Fitting a linear decision boundary

- Probabilistic approach
  - fixes the policy that $\mathbf{w}_{NB}(w)$ (the component of the linear discriminant corresponding to term w) depends only on the statistics of term w in the corpus.
  - Therefore it cannot pick from the entire set of possible linear discriminants

- Discriminative approach
  - try to find a weight vector $\mathbf{w}$ so that the discrimination between the two classes is optimal
  - statistical approaches:
    - perceptrons (neural networks with a single layer)
    - logistic regression
  - most common approach in text categorization

    → support vector machines

Finding a Linear Decision Boundary

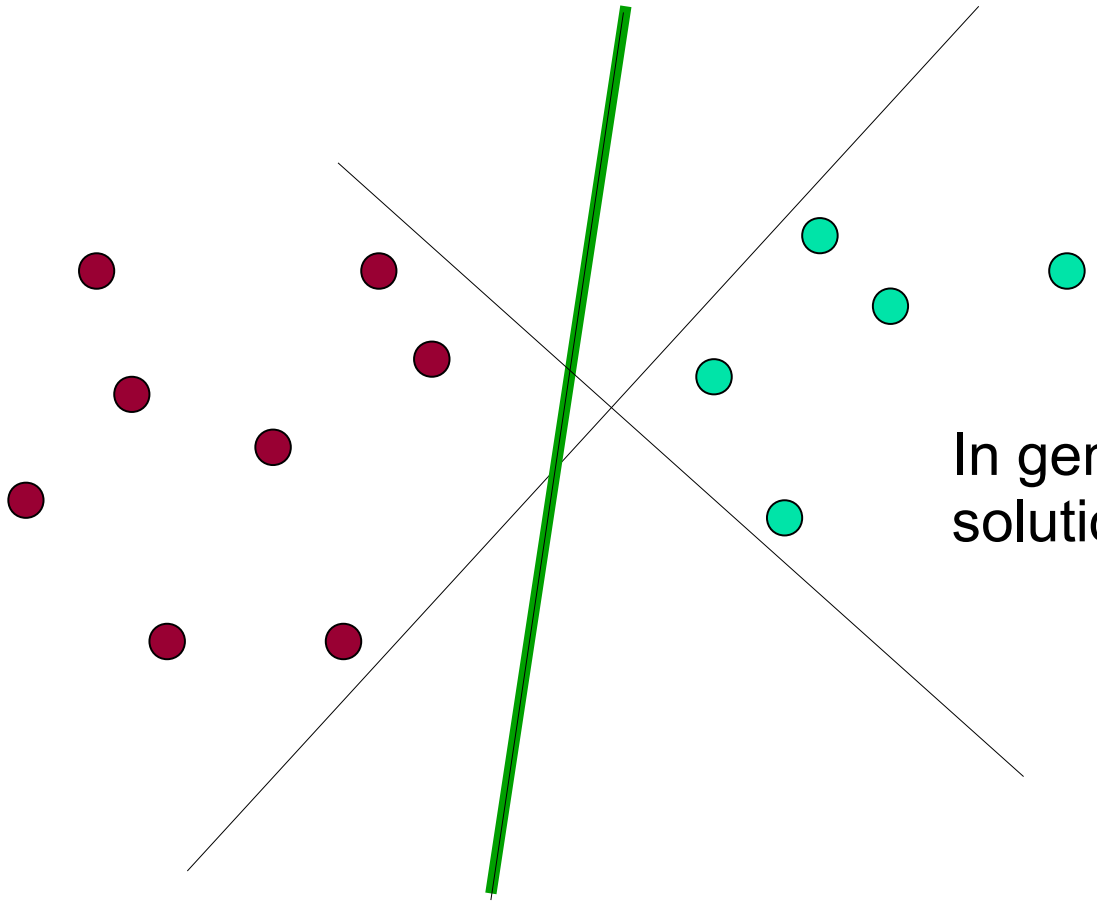


Find w_1, w_2, b, such that

$w_1 x_1 + w_2 x_2 + b \geq 0$ for red points

$w_1 x_1 + w_2 x_2 + b \leq 0$ for green points

Which Hyperplane?

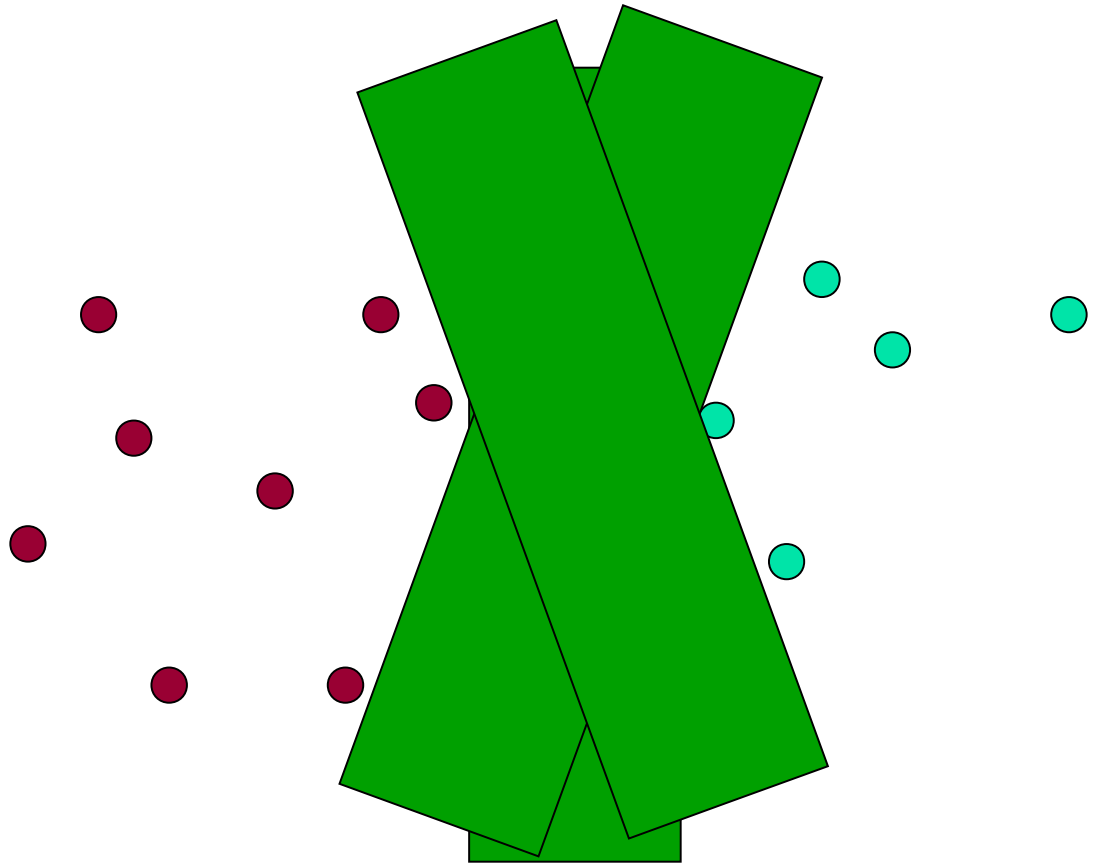


In general, many possible solutions for $\mathbf{w} = (w_1, w_2), b$

- Intuition 1: If there are no points near the decision surface, then there are no very uncertain classifications

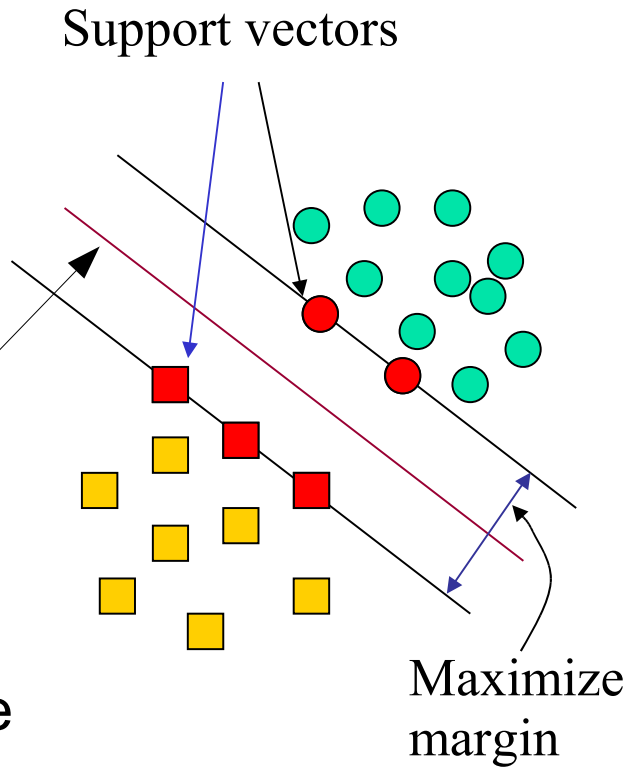
Support Vector Machines: Intuition

- Intuition 2: If you have to place a fat separator between classes, you have less choices, and so overfitting is not so easy



Support Vector Machine (SVM)

- SVMs maximize the *margin* around the separating hyperplane.
  - a.k.a. large margin classifiers
- The decision function is fully specified by a subset of training samples, the *support vectors*.



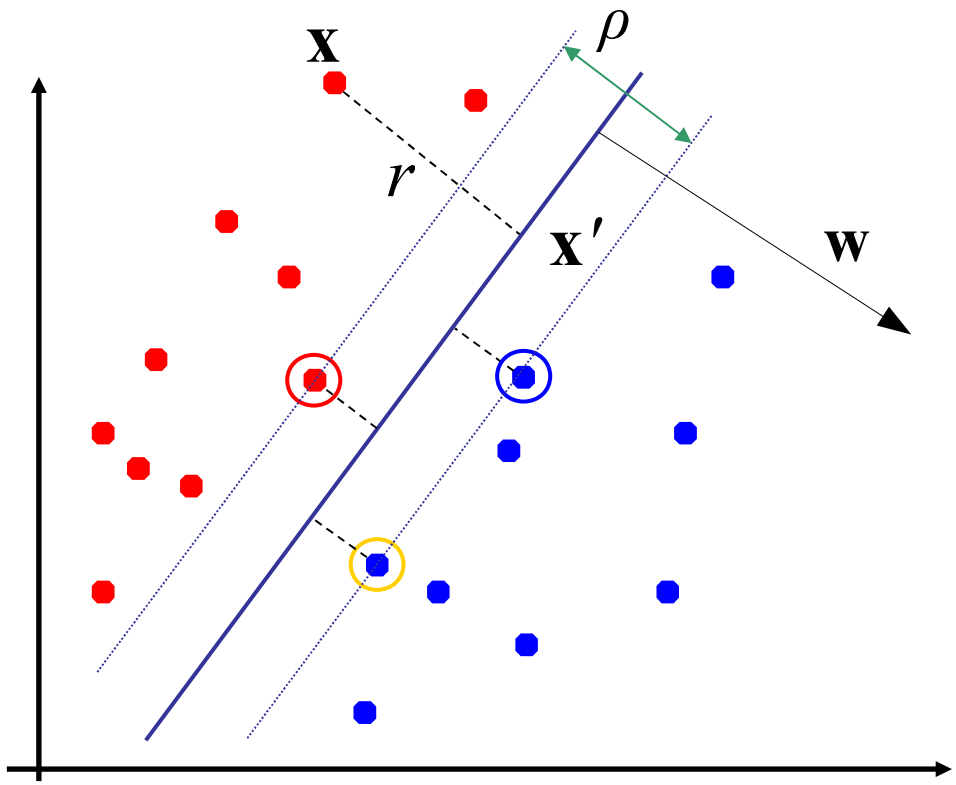
$$\mathbf{w}^T \cdot \mathbf{x}_i + b = 0$$

- Formalization
  - $\mathbf{w}$: normal vector to decision hyperplane
  - $\mathbf{x}_i$: i-th data point
  - y_i: class of data point i ($+1$ or -1) NB: Not 1/0
  - Classifier is:

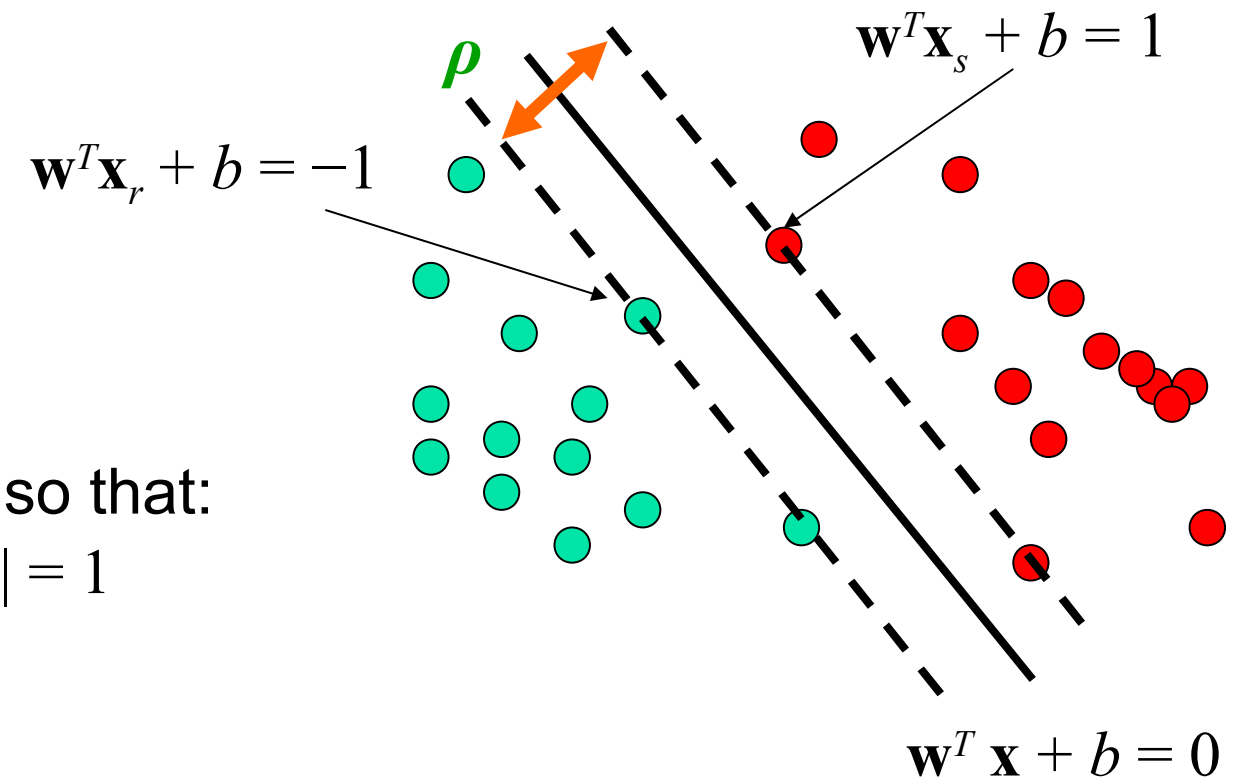
$$f(\mathbf{x}_i) = \text{sign}(\mathbf{w}^T\mathbf{x}_i + b)$$

Geometric Margin

- Distance from example to the separator is $r = y\dfrac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|}$
- Examples closest to the hyperplane are support vectors.
- Margin ρ of the separator is the width of separation between support vectors of classes.



Linear Support Vector Machine (SVM)



- Assumption:
  - $\mathbf{w}$ is normalized so that:

    $\min_{i=1,\ldots,n} |\mathbf{w}^T\mathbf{x}_i + b| = 1$

- This implies:

  $\mathbf{w}^T(\mathbf{x}_r - \mathbf{x}_s) = 2$

  $\rho = ||\mathbf{x}_r - \mathbf{x}_s||_2 = 2/||\mathbf{w}||_2$

Linear SVM Mathematically

- Only the direction of w is important, i.e., we can choose it so that the closest points to the hyperplane have the value 1.
- If all data is at least value 1, the following two constraints follow for a training set $\{(\mathbf{x_i}, y_i)\}$

$$\mathbf{w}^T\mathbf{x_i} + b \geq 1 \quad \text{if } y_i = +1$$

$$\mathbf{w}^T\mathbf{x_i} + b \leq -1 \quad \text{if } y_i = -1$$

  - For support vectors, the inequalities become equalities, which can be rewritten as

$$y_i \cdot (\mathbf{w}^T \mathbf{x}_i + b) = 1$$

- Then, since each example's distance from the hyperplane is

$$r = y\frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|} \quad \rightarrow \text{ the margin is } \rho = \frac{2}{\|\mathbf{w}\|}$$

(the margin is twice the distance r to the support vectors)

Linear SVMs Mathematically (cont.)

- Then we can formulate the *quadratic optimization problem:*

Find $\mathbf{w}$ and b such that the margin

$$\rho = \frac{2}{\|\mathbf{w}\|} \text{ is maximized;}$$

and for all $\{(\mathbf{x_i}, y_i)\}$

$$\mathbf{w^T x_i} + b \geq +1 \quad \text{if } y_i = 1;$$
$$\mathbf{w^T x_i} + b \leq -1 \quad \text{if } y_i = -1$$

- A better formulation ($\max 1/\|\mathbf{w}\| = \min \|\mathbf{w}\| = \min \mathbf{w}^T\mathbf{w}$):

Find $\mathbf{w}$ and b such that

$\mathbf{\Phi(w)} = \frac{1}{2}\, \mathbf{w}^T\mathbf{w}$ is minimized;

and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i\,(\mathbf{w^T x_i} + b) \geq 1$

Solving the Optimization Problem

- This is now
  - optimizing a *quadratic* function
  - subject to *linear* constraints

Find $\mathbf{w}$ and b such that

$\Phi(\mathbf{w}) = \frac{1}{2}\, \mathbf{w}^T\mathbf{w}$ is minimized;

and for all $\{(\mathbf{x}_i, y_i)\}: y_i\, (\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

- Quadratic optimization problems are a well-known class of mathematical programming problems
  - many (rather intricate) algorithms exist for solving them
- The solution involves constructing a *dual problem*
  - where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:

Find $\alpha_1 \ldots \alpha_n$ such that

$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$ is maximized

and

(1) $\Sigma\alpha_i y_i = 0$

(2) $\alpha_i \geq 0$ for all α_i

The Optimization Problem Solution

- The solution has the form:

$$\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x_i} \qquad b = y_k - \mathbf{w}^T\mathbf{x_k} \text{ for any } \mathbf{x_k} \text{ such that } \alpha_k \neq 0$$

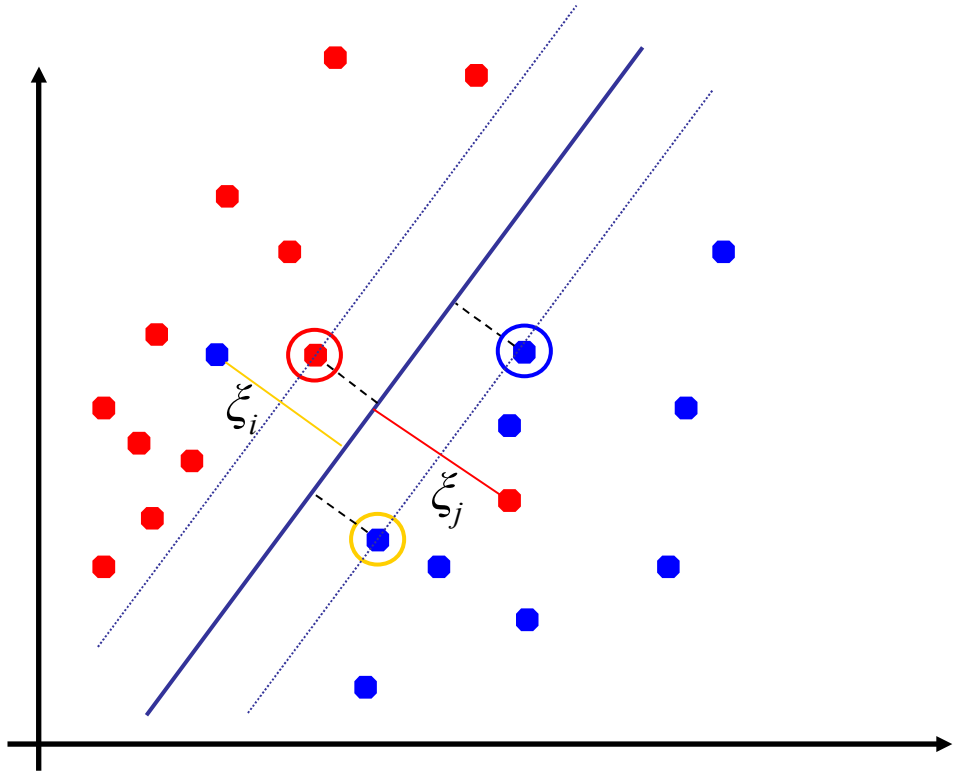
  - $\alpha_i \neq 0$ indicates that corresponding $\mathbf{x}_i$ is a support vector.
- Then the classifying function will have the form:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \Sigma\alpha_i y_i \mathbf{x_i}^T\mathbf{x} + b$$

  - Notice that it relies on an inner product between the test point $\mathbf{x}$ and the support vectors $\mathbf{x}_i$ – we will return to this later.
  - Also keep in mind that solving the optimization problem involved computing the inner products $\mathbf{x}_i^T\mathbf{x}_i$ between all pairs of training points.

Soft Margin Classification

- If the training set is not linearly separable, *slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples.
- Allow some errors
  - Let some points be moved to where they belong, at a cost
- Still, try to minimize training set errors, and to place hyperplane "far" from each class (large margin)



Soft Margin Classification Mathematically

- The old formulation:

Find $\mathbf{w}$ and b such that

$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w}$ is minimized;

and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

- The new formulation incorporating slack variables:

Find $\mathbf{w}$ and b such that

$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ is minimized

and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$

  - Parameter C can be viewed as a way to control overfitting – a regularization term

Soft Margin Classification – Solution

- The dual problem for soft margin classification:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x_i}^T\mathbf{x_j}$ is maximized

and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \le \alpha_i \le C$ for all α_i

  - NOTE: Neither slack variables ξ_i nor their Lagrange multipliers appear in the dual problem!

- Solution to the dual problem is:

$\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x_i}$
$b = y_k(1 - \xi_k) - \mathbf{w}^T\mathbf{x_k}$
where $k = \text{argmax}_k\ \alpha_k$

But $\mathbf{w}$ not needed explicitly for classification!

$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x_i}^T\mathbf{x} + b$

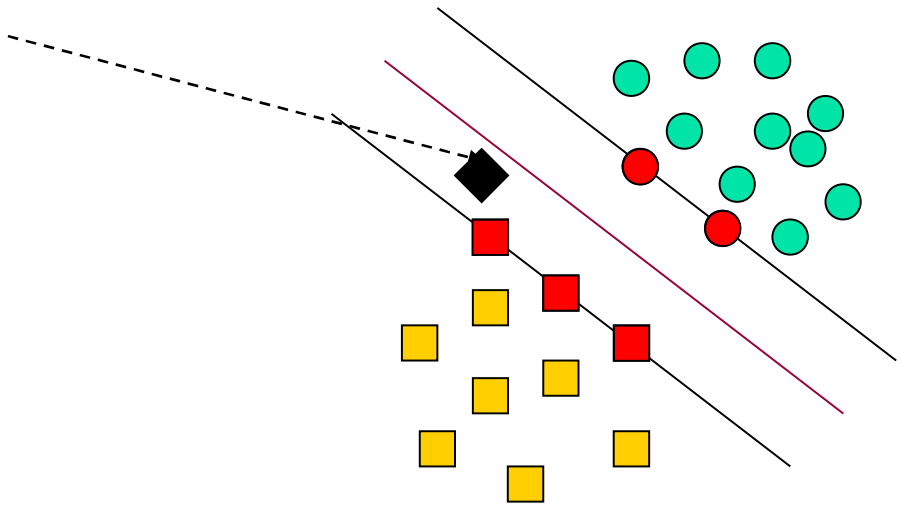
Classification with SVMs

- Given a new point (x_1,x_2), we can score its projection onto the hyperplane normal:
  - In 2 dims: $\text{score} = w_1x_1+w_2x_2+b.$
  - in general: $\text{score} = \mathbf{w}^T\mathbf{x} + b = \Sigma\alpha_i y_i \mathbf{x_i}^T\mathbf{x} + b$

sum runs over all support vectors (all α_i other are 0)

score ≥ 0: yes

score < 0: no



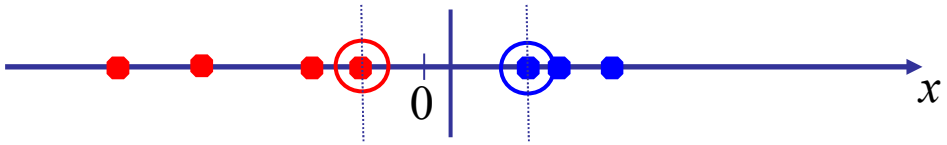
Linear SVMs: Summary

- The classifier is a separating hyperplane.
- Most "important" training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points $\mathbf{x}_i$ are support vectors with non-zero Lagrangian multipliers α_i.
- Both in the dual formulation of the problem and in the solution training points appear only inside inner products:

Foto by Burr Settles, http://www.cs.cmu.edu/~bsettles/

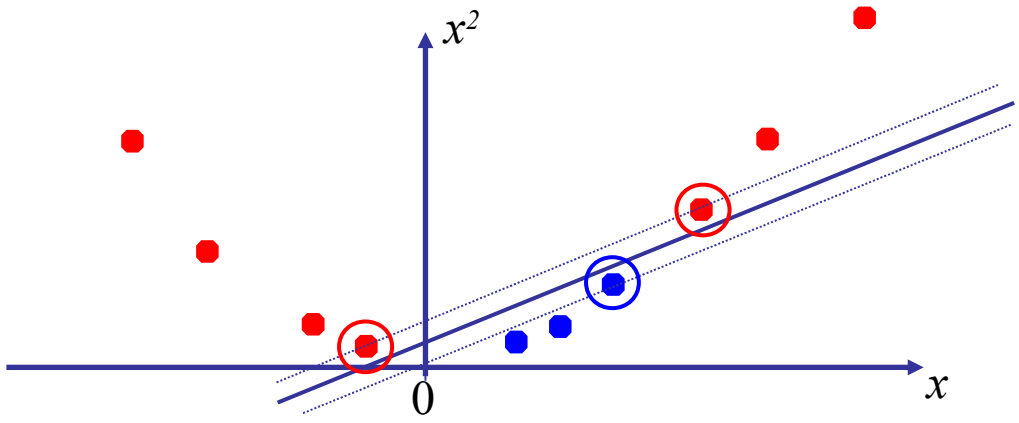
Non-linear SVMs

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



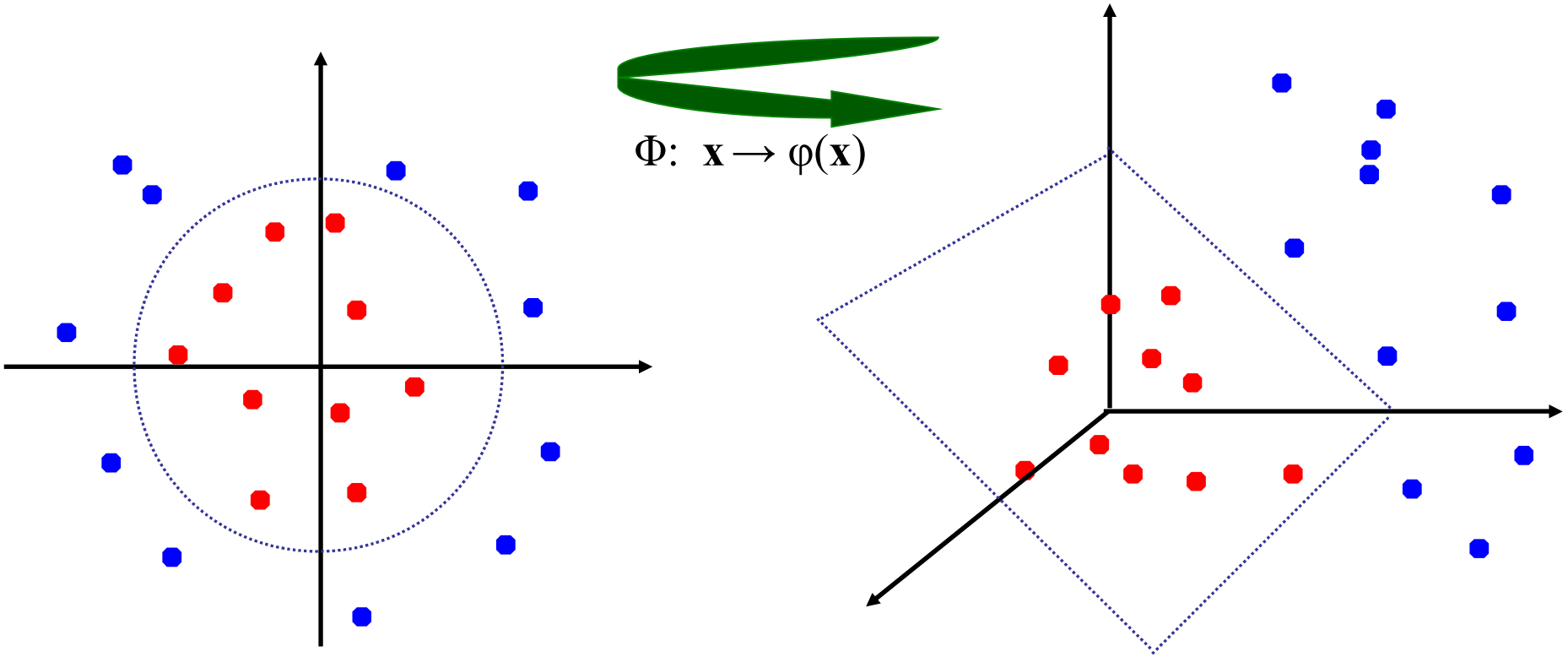
- But what are we going to do if the dataset is just too hard?

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



Non-linear SVMs: Feature spaces

- **General idea:** the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



The "Kernel Trick"

- The linear classifier relies on an inner product between vectors $K(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^T\mathbf{x}_j$
- If every datapoint is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$, the inner product becomes:

$$K(\mathbf{x}_i,\mathbf{x}_j)= \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$$

- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.
- Example:

  2-dimensional vectors $\mathbf{x}=[x_1\ x_2]$; let $K(\mathbf{x}_i,\mathbf{x}_j)=(1+\mathbf{x}_i^T\mathbf{x}_j)^2$,

  Need to show that $K(\mathbf{x}_i,\mathbf{x}_j)= \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$:

$$K(\mathbf{x}_i,\mathbf{x}_j) = (1+\mathbf{x}_i^T\mathbf{x}_j)^2 = 1+ x_{i1}^2x_{j1}^2 + 2\,x_{i1}x_{j1}\,x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} =$$

$$= [1\ \ x_{i1}^2\ \sqrt{2}\,x_{i1}x_{i2}\ \ x_{i2}^2\ \sqrt{2}x_{i1}\ \sqrt{2}x_{i2}]^T\,[1\ \ x_{j1}^2\ \sqrt{2}\,x_{j1}x_{j2}\ \ x_{j2}^2\ \sqrt{2}x_{j1}\ \sqrt{2}x_{j2}]$$

$$= \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j) \qquad \text{where } \varphi(\mathbf{x}) = [1\ \ x_1^2\ \sqrt{2}\,x_1x_2\ \ x_2^2\ \ \sqrt{2}x_1\ \sqrt{2}x_2]$$

Kernels

- Why use kernels?
  - Make non-separable problem separable.
  - Map data into better representational space
- Common kernels
  - Linear:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \cdot \mathbf{x}_j$$

  - Polynomial:

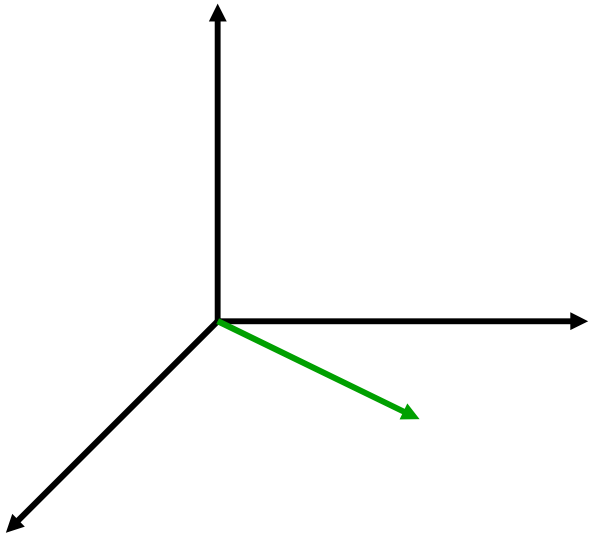
$$K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \cdot \mathbf{x}_j)^d$$

  - Radial basis function (infinite dimensional space)

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}$$

High Dimensional Data

- Pictures like the one at right are misleading!
  - Documents are zero along almost all axes
  - Most document pairs are very far apart
    - (i.e., not strictly orthogonal, but only share very common words and a few scattered others)



- In classification terms:
  - virtually all document sets are separable, for almost any classification

- This is part of why linear classifiers are quite successful in text classification
  → SVMs with linear Kernels are usually sufficient!

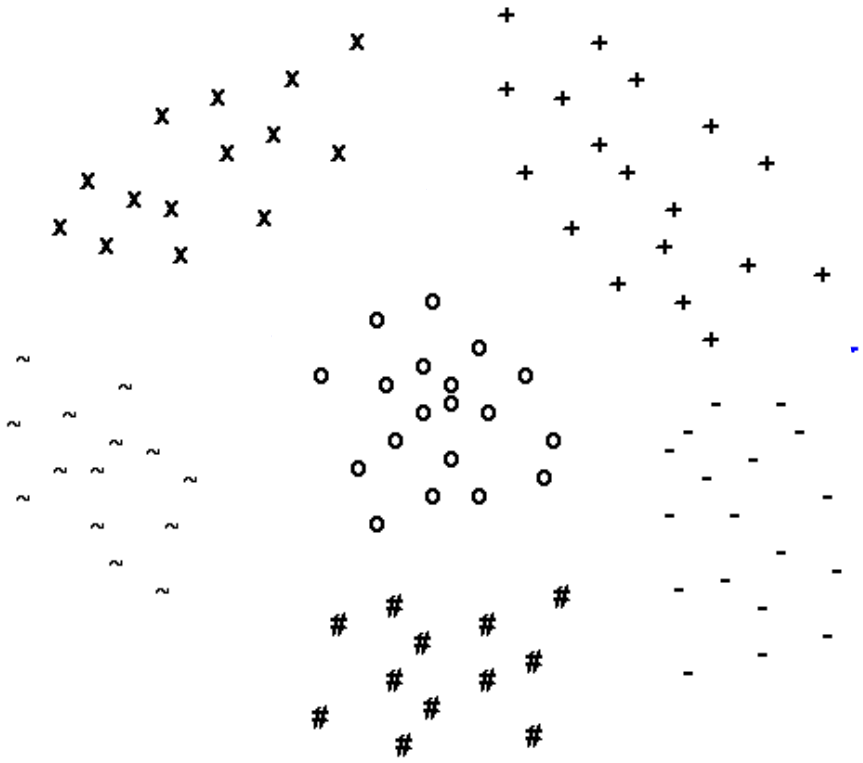
Performance

- Comparison with other classifiers
  - Amongst most accurate classifier for text
  - Better accuracy than naive Bayes and decision tree classifier,
- Different Kernels
  - Linear SVMs suffice for most text classification tasks
  - standard text classification tasks have classes almost separable using a hyperplane in feature space
    - becaue of high dimensionality of the feature space
- Computational Efficiency
  - requires to solve a quadratic optimization problem.
    - Working set: refine a few λ at a time holding the others fixed.
  - overall quadratic run-time
    - can be reduced by clever selection of the working set

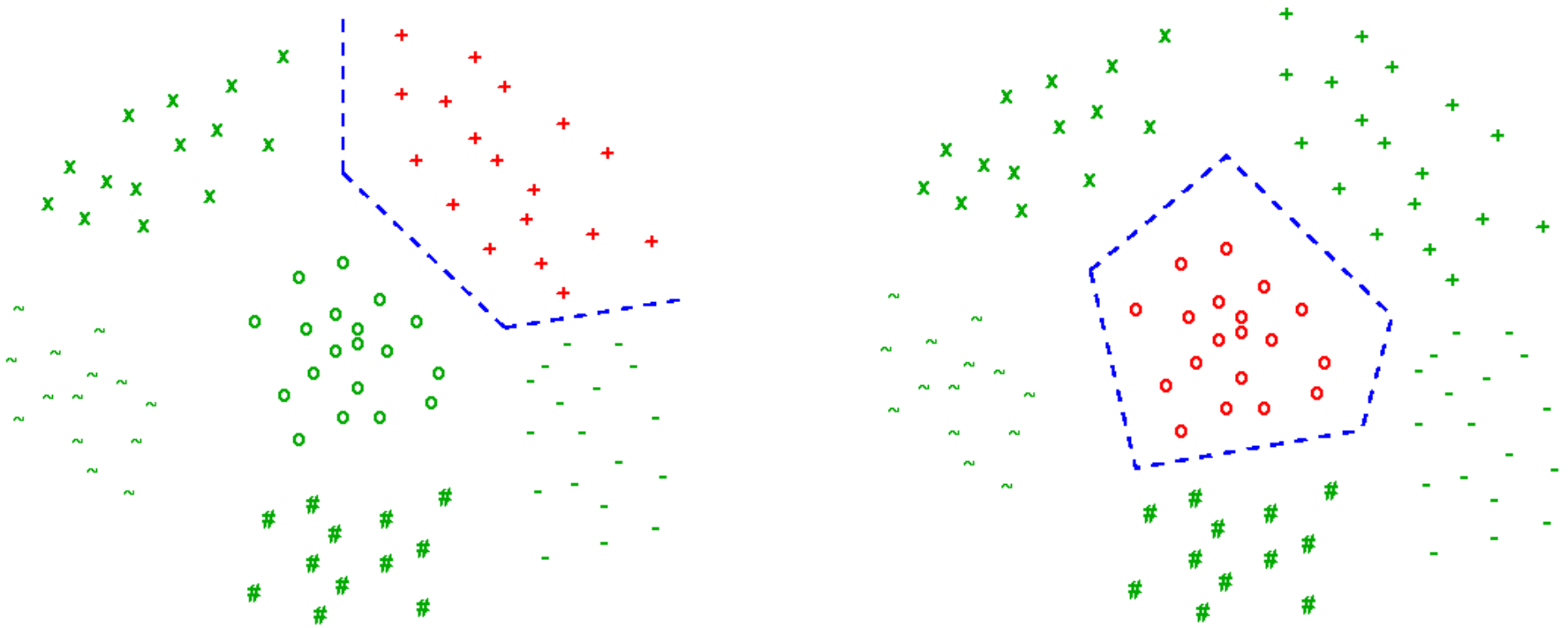
Multi-Class Classification

Many problems have $c > 2$ classes not just two

- naïve Bayes, k-NN, Rocchio can handle multiple classes naturally
- Support Vector Machines need to be extended
  - SVM learns a hyperplane that separates the example space into two regions
  - Simple idea:
    Learn multiple such surfaces and combine them



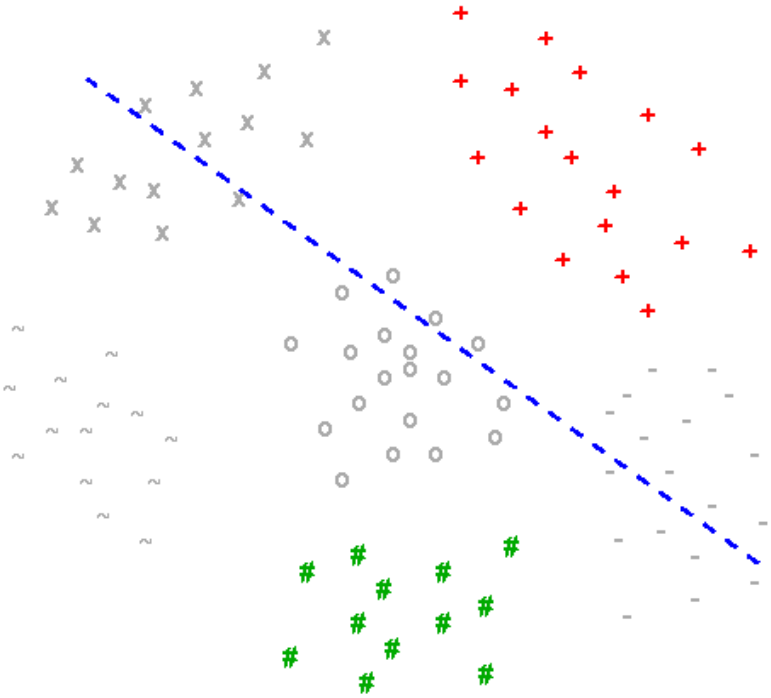
One-against-all

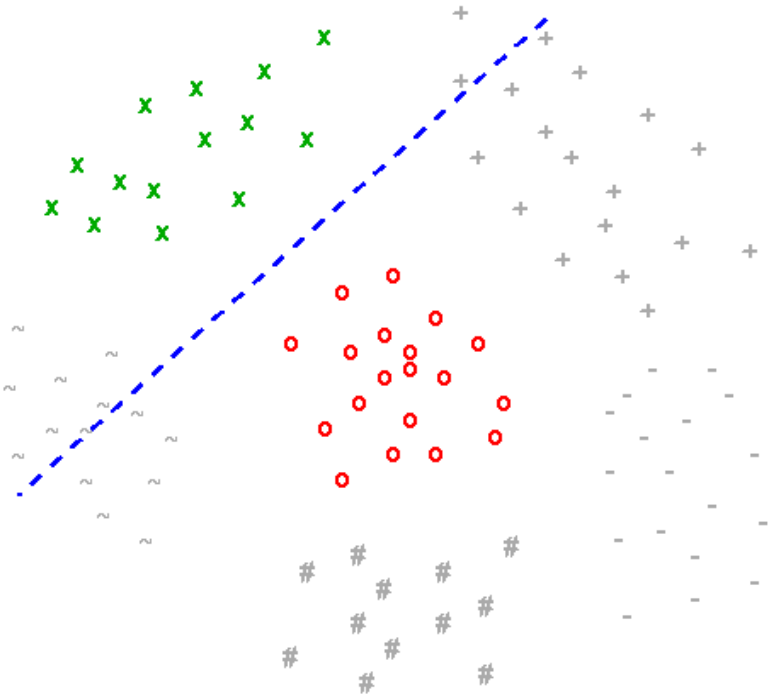


- c binary problems, one for each class
- label examples of class positive, all others negative
- predict class with the highest response value
  - e.g., closest to to decision boundary
    (not trivial for SVMs because of different scales of hyperplanes

Pairwise Classification

- $c(c\text{-}1)/2$ problems
- each class against each other class





- smaller training sets
- simpler decision boundaries
- larger margins

Aggregating Pairwise Predictions

- Aggregate the predictions $P(C_i > C_j)$ of the binary classifiers into a final ranking by computing a score s_i for each class I

  - Voting: count the number of predictions for each class
    (number of points in a tournament)

$$s_i = \sum_{j=1}^{c} \delta\{P(C_i > C_j) > 0.5\} \qquad \delta\{x\} = \begin{cases} 1 & \text{if } x = \text{true} \\ 0 & \text{if } x = \text{false} \end{cases}$$

  - Weighted Voting: weight the predictions by their probability

$$s_i = \sum_{j=1}^{c} P(C_i > C_j)$$

- General Pairwise Coupling problem:
  - Given $P(C_i > C_j) = P(C_i | C_i, C_j)$ for all i, j
  - Find $P(C_i)$ for all i
  - Can be turned into an (underconstrained) system of linear equations

Rule-based Classifiers

- A classifier basically is a function that computes the output (the *class*) from the input (the *attribute values*)
- Rule learning tries to represent this function in the form of (a set of) IF-THEN rules

  $\texttt{IF (att}_i \texttt{ = val}_{iI}\texttt{) AND (att}_j \texttt{ = val}_{jJ}\texttt{) THEN class}_k$

- Coverage
  - A rule is said to ***cover*** an example if the example satisfies the conditions of the rule.
- Correctness
  - ***completeness***: Each example should be covered by (at least) one rule
  - ***consistency***: For each example, the predicted class should be identical to the true class.

Separate-and-Conquer Strategy

- Learn rules for each class separately
  - use the biggest class as the default class
- To learn rules for one class:
  - Add rules to a theory until all examples of a class are covered (*completeness*)
  - remove the covered examples
- To learn a single rule:
  - Add conditions to the rule that
    - Cover as many examples p from the class as possible
    - Exclude as many examples n from other classes as possible
    - E.g., maximize $\frac{p}{(p+n)}$ or better the Laplace estimate $\frac{(p+1)}{(p+n+2)}$

Set-valued Features

- Use binary conditions of the form $t_i \in D$
- Efficient representation of binary conditions by listing all words that occur
  (vector-based representation also lists those that do not occur)
- Several, separate set-valued features are possible (thus it is an extension of the vector-space model)
  - this allows conditions of the form, e.g., $t_i \in title(D)$
- Useful for tasks with
  - more than one text-based features
  - combining regular features with text-based features
  - e.g. seminar announcements, classifying e-mails

Occam's Razor

> Entities should not be multiplied beyond necessity.
>
> *William of Ockham (1285 - 1349)*

- Machine Learning Interpretation:
  - Among theories of (approximately) equal quality on the *training* data, simpler theories have a better chance to be more accurate on the *test* data
  - It is desirable to find a trade-off between *accuracy* and *complexity* of a model
- (Debatable) Probabilistic Justification:
  - There are more complex theories than simple theories. Thus a simple theory is less likely to explain the observed phenomena by chance.

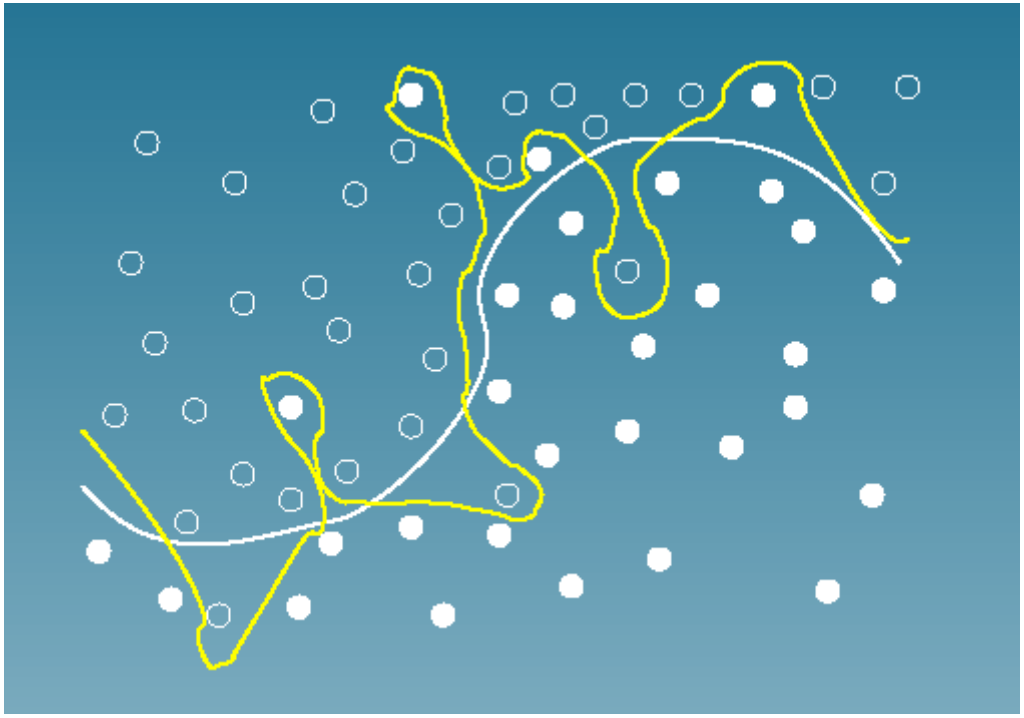
Overfitting

- Overfitting
  - Given
    - a fairly general model class (e.g., rules)
    - enough degrees of freedom (e.g., no length restriction)
  - you can always find a model that explains the data
- Such concepts do not generalize well!
- Particularly bad for noisy data
  - Data often contain errors due to
    - inconsistent classification
    - measurement errors
    - missing values

→ Capacity control

Capacity Control

- Choose the right complexity of a classifier



Overfitting Avoidance in SVMs

- Choose simpler model classes
  - Linear kernels or polynomial kernels with a low degree d
- Choose a lower regularization parameter C
  - High values of C force better fit to the data
  - Low values of C allow more freedom in selecting the slack variables
- Note:
  - Overfitting Avoidance in SVMs is also known as Capacity Control or Regularization

Overfitting Avoidance in Rule Learning

- Choose a simpler model class
  - restrict number of conditions in a rule
  - demand minimum coverage for a rule
- Pruning
  - simplify a theory after it has been learned
- Reduced Error Pruning
  1. Reserve part of the data for validation
  2. Learn a rule set
  3. Simplify rule set by deleting rules and conditions as long as this does not decrease accuracy on the validation set
- Incremental REP
  - Do this after each individual rule is learned

RIPPER (Cohen, 1995)

Efficient algorithm for learning classification rules

- covering algorithm (aka *separate-and-conquer*)
- incremental pruning of rules (I-REP)
- set-valued features support text mining

The Compress Algorithm

- Simple, elegant algorithm capturing a Minimum-Description Length Idea:
  1. Put all documents of one class into a separate directory
  2. `compress/zip` each directory into file `<class_i>.zip`
- To classify a new document:
  1. Tentatively assign the document to each class (by adding it to the respective directories)
  2. compress/zip each directory into file `<class_i>_new.zip`
  3. assign document to the class for which the *distance measure* `|<class_i>.zip|-|<class_i>_new.zip|` is minimal
- Benedetto et al. (Phys. Rev. Letters 2002) report results for
  - language recognition (100% accuracy for 10 EC languages)
  - authorship determination (93.3% for 11 Italian authors)
  - document clustering (similarity tree of European languages)

Language Tree generated with Variant of Compress-Algorithm

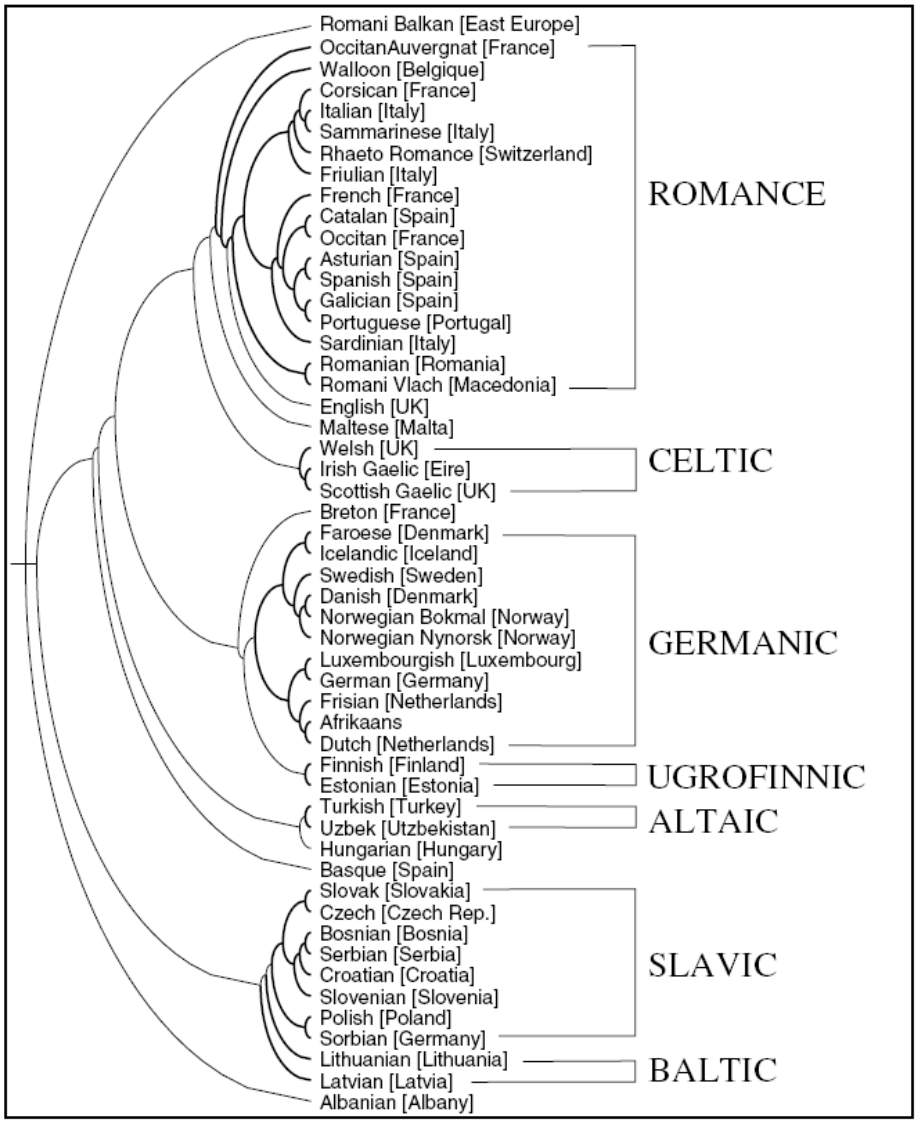


FIG. 1. Language Tree: This figure illustrates the phylogenetic-like tree constructed on the basis of more than 50 different versions of "The Universal Declaration of Human Rights." The tree is obtained using the Fitch-Margoliash method applied to a distance matrix whose elements are computed in terms of the relative entropy between pairs of texts. The tree features essentially all the main linguistic groups of the Euro-Asiatic continent (Romance, Celtic, Germanic, Ugro-Finnic, Slavic, Baltic, Altaic), as well as a few isolated languages such as the Maltese, typically considered an Afro-Asiatic language, and the Basque, classified as a non-Indo-European language, and whose origins and relationships with other languages are uncertain. Notice that the tree is unrooted, i.e., it does not require any hypothesis about common ancestors for the languages. What is important is the relative positions between pairs of languages. The branch lengths do not correspond to the actual distances in the distance matrix.

Dario Benedetto, Emanuele Caglioti, and Vittorio Loreto, Language Trees and Zipping, *Physical Review Letters* 88, 2002

Evaluation of Learned Models

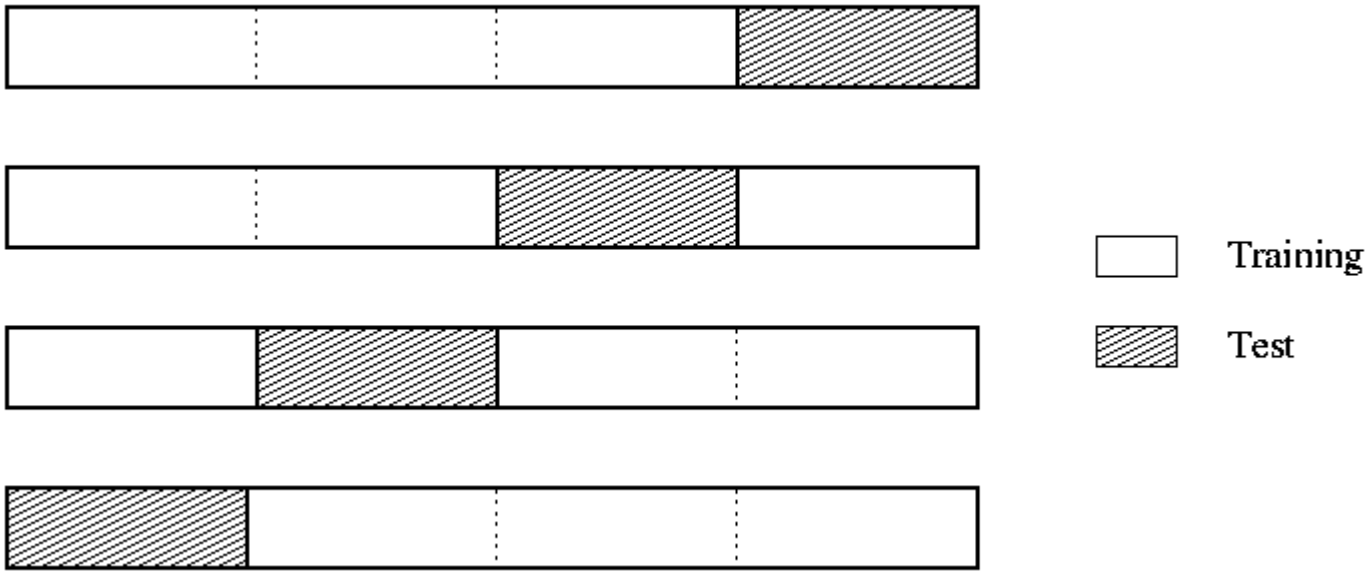
- Validation through experts
  - a domain experts evaluates the plausibility of a learned model
    - + subjective, time-intensive, costly
    - – but often the only option (e.g., clustering)
- Validation on data
  - evaluate the accuracy of the model on a separate dataset drawn from the same distribution as the training data
    - – labeled data are scarce, could be better used for training
    - + fast and simple, off-line, no domain knowledge needed, methods for re-using training data exist (e.g., cross-validation)
- On-line Validation
  - test the learned model in a fielded application
    - + gives the best estimate for the overall utility
    - – bad models may be costly

Out-of-Sample Testing

- Performance cannot be measured on training data
  - overfitting!
- Reserve a portion of the available data for testing
- Problem:
  - waste of data
  - labelling may be expensive

Cross-Validation

- split dataset into n (usually 10) partitions
- for every partition p
  - use other n-1 partitions for learning and partition p for testing
- average the results



Evaluation

- In Machine Learning:
  Accuracy = percentage of correctly classified examples
- Confusion Matrix:

	Classified as +	**Classified as -**	
Is +	a	c	a+c
Is -	b	d	b+d
	a+b	c+d	n

$$recall = \frac{a}{(a+c)}$$

$$precision = \frac{a}{(a+b)}$$

$$accuracy = \frac{(a+d)}{n}$$

Evaluation for Multi-Class Problems

- for multi-class problems, the confusion matrix has many more entries:

classified as

true class	A	B	C	D	
A	$n_{A,A}$	$n_{B,A}$	$n_{C,A}$	$n_{D,A}$	n_A
B	$n_{A,B}$	$n_{B,B}$	$n_{C,B}$	$n_{D,B}$	n_B
C	$n_{A,C}$	$n_{B,C}$	$n_{C,C}$	$n_{D,C}$	n_C
D	$n_{A,D}$	$n_{B,D}$	$n_{C,D}$	$n_{D,D}$	n_D
	$\overline{n}_A$	$\overline{n}_B$	$\overline{n}_C$	$\overline{n}_D$	n

- accuracy is defined analogously to the two-class case:

$$accuracy = \frac{n_{A,A} + n_{B,B} + n_{C,C} + n_{D,D}}{n}$$

Recall and Precision for Multi-Class Problems

- For multi-class text classification tasks, recall and precision can be defined for each category separately
- Recall of Class X:
  - How many documents of class X have been recognized as class X?
- Precision of Class X:
  - How many of our predictions for class X were correct?
- Predictions for Class X can be summarized in a 2x2 table
  - z.B:

$$X = A, \overline{X} = \{B, C, D\}$$

	classified X	~~classified not X~~	
is X	$n_{X,X}$	$n_{\overline{X},X}$	n_X
~~is not X~~	$n_{X,\overline{X}}$	$n_{\overline{X},\overline{X}}$	$n_{\overline{X}}$
	$\overline{n}_X$	$\overline{n}_{\overline{X}}$	n

Micro- and Macro-Averaging

- To obtain a single overall estimate for recall and precision
  - we have to combine the estimates for the individual classes
- Two strategies:
  - **Micro-Averaging:**
    - add up the 2x2 contingency tables for each class
    - compute recall and precision from the summary table
  - **Macro-Averaging:**
    - compute recall and precision for each contingency table
    - average the recall and precision estimates
- Basic difference:
  - Micro-Averaging prefers large classes
    - they dominate the sums
  - Macro-Averaging gives equal weight to each class
    - r/p on smaller classes counts as much as on larger classes

Macro-Averaging

True \ Predicted	C1	~~C1~~	
C1	15	5	20
~~C1~~	10	70	80
	25	75	100

True \ Predicted	C2	~~C2~~	
C2	20	10	30
~~C2~~	12	58	70
	32	68	100

True \ Predicted	C3	~~C3~~	
C3	45	5	50
~~C3~~	5	45	50
	50	50	100

$$prec(c1) = \frac{15}{25} = 0.600 \qquad prec(c2) = \frac{20}{32} = 0.625 \qquad prec(c3) = \frac{45}{50} = 0.900$$

$$avg.\ prec = \frac{prec(c1) + prec(c2) + prec(c3)}{3} = 0.708$$

$$recl(c1) = \frac{15}{20} = 0.750 \qquad recl(c2) = \frac{20}{30} = 0.667 \qquad recl(c3) = \frac{45}{50} = 0.900$$

$$avg.\ recl = \frac{recl(c1) + recl(c2) + recl(c3)}{3} = 0.772$$

Micro-Averaging

Predicted

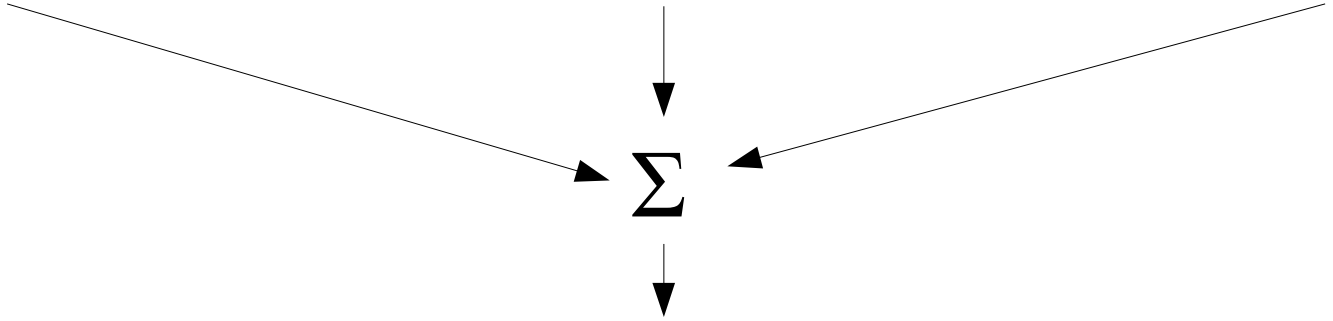
True	C1	~~C1~~	
C1	15	5	20
~~C1~~	10	70	80
	25	75	100

Predicted

True	C2	~~C2~~	
C2	20	10	30
~~C2~~	12	58	70
	32	68	100

Predicted

True	C3	~~C3~~	
C3	45	5	50
~~C3~~	5	45	50
	50	50	100



Predicted

True	C	~~C~~	
C	80	20	100
~~C~~	27	173	200
	107	193	300

$$avg.\ prec = \frac{80}{107} = 0.748$$

$$avg.\ recl = \frac{80}{100} = 0.800$$

Micro-Averaged estimates are in this case higher because the performance on the largest class (C3) was best

Benchmark Datasets

Publicly available Benchmark Datasets facilitate standardized evaluation and comparisons to previous work

- Reuters-21578
  - 12,902 labeled documents
  - 10% documents with multiple class labels
- OHSUMED
  - 348,566 abstracts from medical journals
- 20 newsgroups
  - 18,800 labeled USENET postings
  - 20 leaf classes, 5 root level classes
  - more recent 19 newsgroups
- WebKB
  - 8300 documents in 7 academic categories.
- Industry sectors
  - 10,000 home pages of companies from 105 industry sectors
  - Shallow hierarchies of sector names

Reuters-21578 Dataset

- Most (over)used data set
- originally 21578 documents, not all of them are useful
- 9603 training, 3299 test articles (ModApte split)
- 118 categories
  - Multilabel Classification: An article can be in more than one category
  - Simple approach: Learn 118 binary category distinctions
- Average document: about 90 types, 200 tokens
- Average number of classes assigned
  - 1.24 for docs with at least one category
- Only about 10 out of 118 categories are large

Common categories
(#train, #test)

- Earn (2877, 1087)
- Acquisitions (1650, 179)
- Money-fx (538, 179)
- Grain (433, 149)
- Crude (389, 189)
- Trade (369,119)
- Interest (347, 131)
- Ship (197, 89)
- Wheat (212, 71)
- Corn (182, 56)

Reuters-21578 Sample Document

<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET" OLDID="12981" NEWID="798">

<DATE> 2-MAR-1987 16:51:43.42</DATE>

<TOPICS><D>livestock</D><D>hog</D></TOPICS>

<TITLE>AMERICAN PORK CONGRESS KICKS OFF TOMORROW</TITLE>

<DATELINE> CHICAGO, March 2 - </DATELINE><BODY>The American Pork Congress kicks off tomorrow, March 3, in Indianapolis with 160 of the nations pork producers from 44 member states determining industry positions on a number of issues, according to the National Pork Producers Council, NPPC.

Delegates to the three day Congress will be considering 26 resolutions concerning various issues, including the future direction of farm policy and the tax law as it applies to the agriculture sector. The delegates will also debate whether to endorse concepts of a national PRV (pseudorabies virus) control and eradication program, the NPPC said.

A large trade show, in conjunction with the congress, will feature the latest in technology in all areas of the industry, the NPPC added. Reuter

&#3;</BODY></TEXT></REUTERS>

Reuters – Accuracy with different Algorithms

	Rocchio	NBayes	Trees	LinearSVM
earn	92,9%	95,9%	97,8%	98,2%
acq	64,7%	87,8%	89,7%	92,8%
money-fx	46,7%	56,6%	66,2%	74,0%
grain	67,5%	78,8%	85,0%	92,4%
crude	70,1%	79,5%	85,0%	88,3%
trade	65,1%	63,9%	72,5%	73,5%
interest	63,4%	64,9%	67,1%	76,3%
ship	49,2%	85,4%	74,2%	78,0%
wheat	68,9%	69,7%	92,5%	89,7%
corn	48,2%	65,3%	91,8%	91,1%
Avg Top 10	64,6%	81,5%	88,4%	91,4%
Avg All Cat	61,7%	75,2%	na	86,4%

Results taken from S. Dumais et al. 1998

Reuters - SVM with different Kernels

	Bayes	Rocchio	C4.5	k-NN	SVM (poly) degree d = 1	2	3	4	5	SVM (rbf) width γ = 0.6	0.8	1.0	1.2
earn	95.9	96.1	96.1	97.3	98.2	98.4	**98.5**	98.4	98.3	**98.5**	98.5	98.4	98.3
acq	91.5	92.1	85.3	92.0	92.6	94.6	**95.2**	95.2	95.3	95.0	95.3	95.3	**95.4**
money-fx	62.9	67.6	69.4	78.2	66.9	72.5	75.4	74.9	**76.2**	74.0	75.4	**76.3**	75.9
grain	72.5	79.5	89.1	82.2	91.3	93.1	**92.4**	91.3	89.9	**93.1**	91.9	91.9	90.6
crude	81.0	81.5	75.5	85.7	86.0	87.3	88.6	**88.9**	87.8	**88.9**	89.0	88.9	88.2
trade	50.0	77.4	59.2	77.4	69.2	75.5	76.6	77.3	**77.1**	76.9	78.0	**77.8**	76.8
interest	58.0	72.5	49.1	74.0	69.8	63.3	67.9	73.1	**76.2**	74.4	75.0	**76.2**	76.1
ship	78.7	83.1	80.9	79.2	82.0	85.4	86.0	**86.5**	86.0	**85.4**	86.5	87.6	87.1
wheat	60.6	79.4	85.5	76.6	83.1	84.5	85.2	**85.9**	83.8	**85.2**	85.9	85.9	85.9
corn	47.3	62.2	87.7	77.9	86.0	86.5	85.3	**85.7**	83.9	**85.1**	85.7	85.7	84.5
microavg.	**72.0**	**79.9**	**79.4**	**82.3**	84.2	85.1	85.9	86.2	85.9	86.4	86.5	86.3	86.2
					combined: **86.0**					combined: **86.4**			

Fig. 2. Precision/recall-breakeven point on the ten most frequent Reuters categories and microaveraged performance over all Reuters categories. k-NN, Rocchio, and C4.5 achieve highest performance at 1000 features (with $k = 30$ for k-NN and $\beta = 1.0$ for Rocchio). Naive Bayes performs best using all features.

Reuters – Micro F1 vs. Macro F1

- Results of five Text Classification Methods on the REUTERS-21578 benchmark

Table 1: Performance summary of classifiers

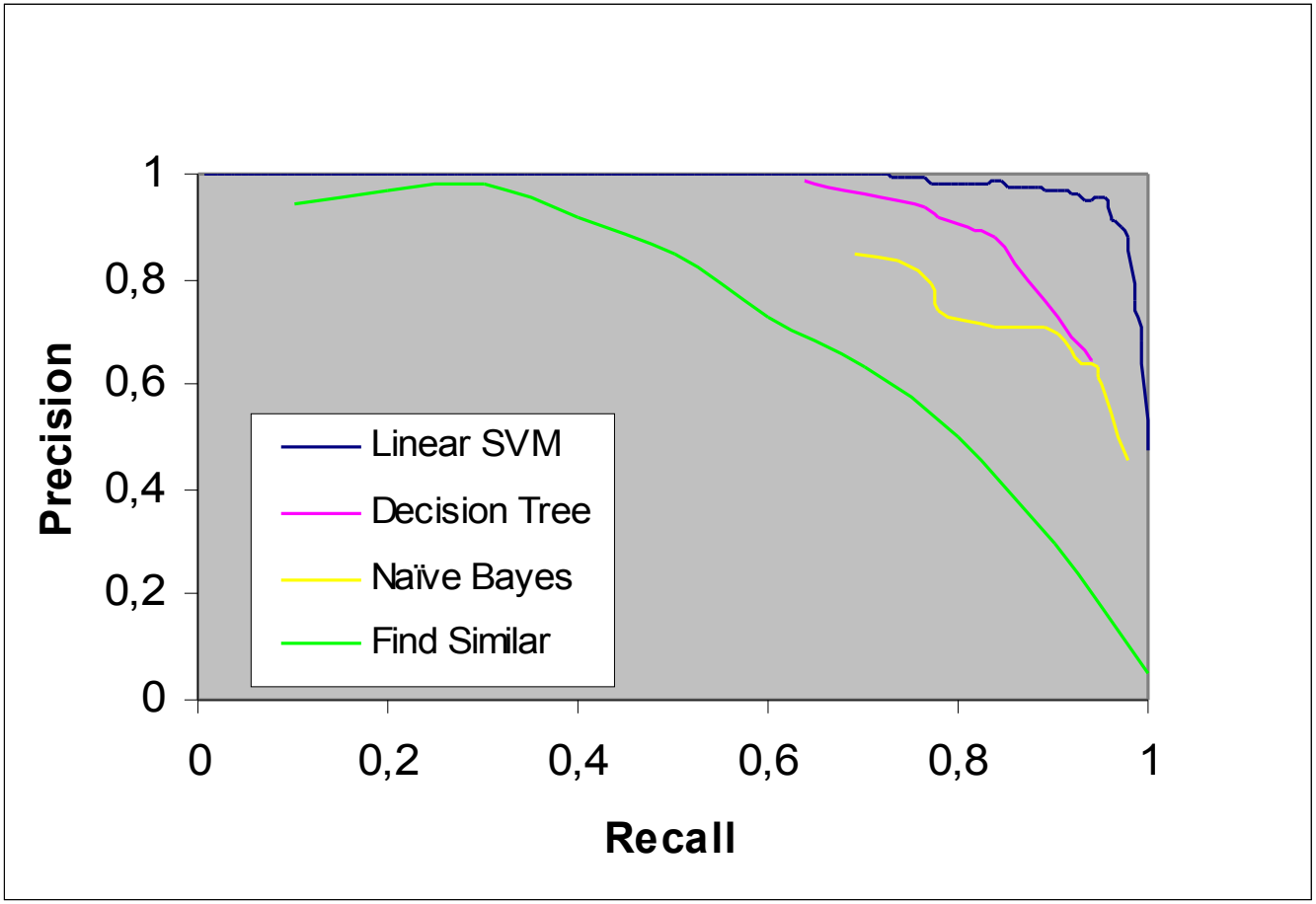
method	miR	miP	miF1	maF1	error
SVM	.8120	.9137	.8599	.5251	.00365
KNN	.8339	.8807	.8567	.5242	.00385
LSF	.8507	.8489	.8498	.5008	.00414
NNet	.7842	.8785	.8287	.3765	.00447
NB	.7688	.8245	.7956	.3886	.00544

miR = micro-avg recall; miP = micro-avg prec.;
miF1 = micro-avg F1; maF1 = macro-avg F1.

Source:Yang & Liu, SIGIR 1999

Reuters – Recall/Precision Curve

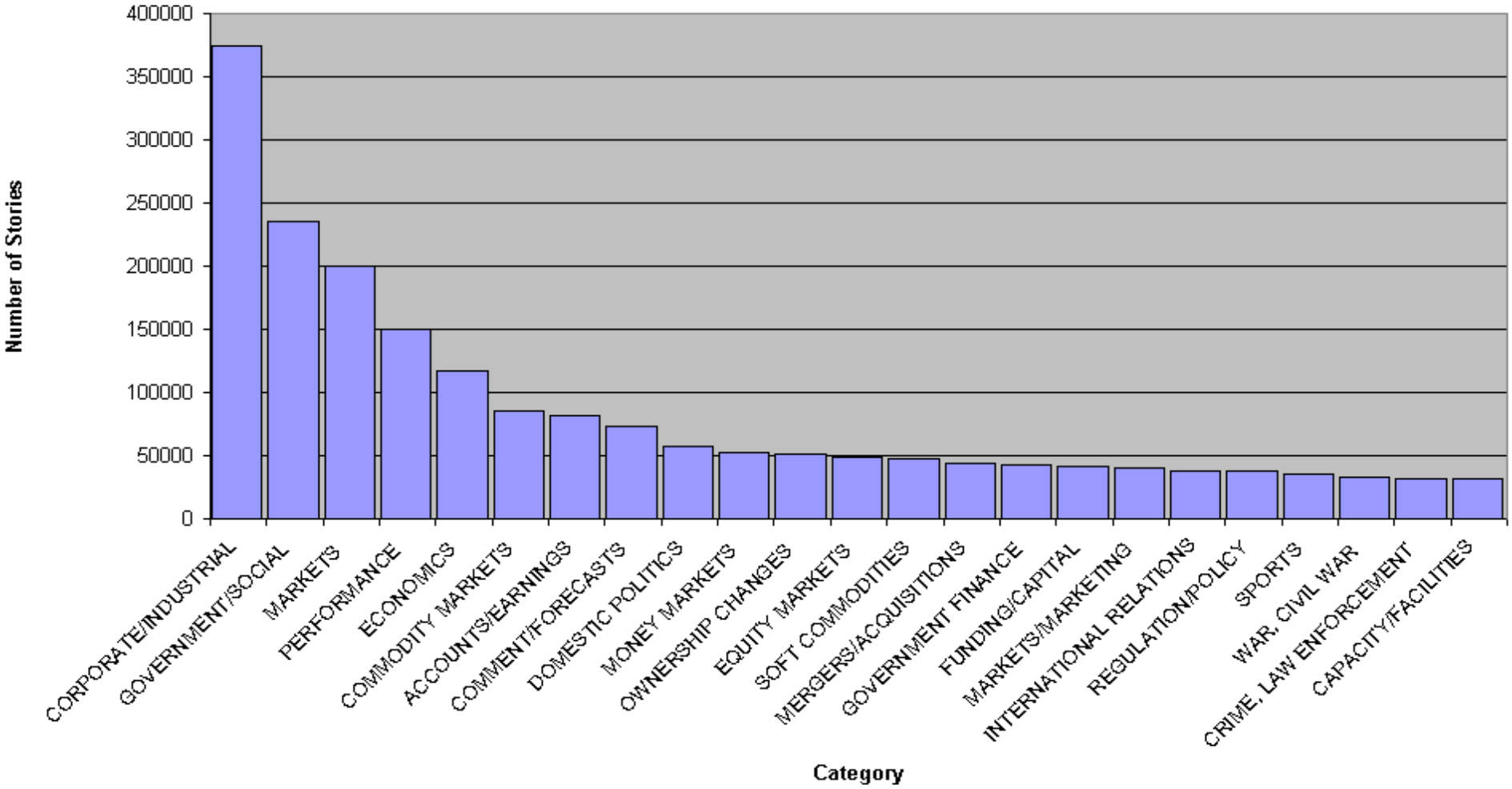
- Comparison of Linear SVM, Decision Tree, (Binary) Naive Bayes, and a version of nearest neighbor on one Reuters category



Graph taken from S. Dumais, LOC talk, 1999.

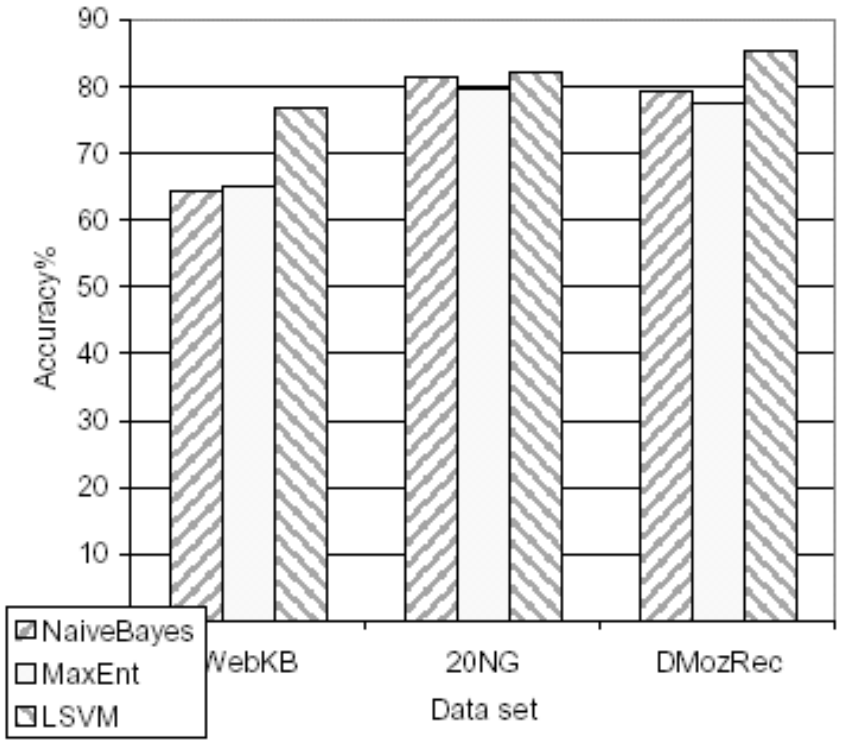
New Reuters: RCV1: 810,000 docs

- Top topics in Reuters RCV1



Multiple Datasets

- Comparison of accuracy across three classifiers:
  - Naive Bayes, Maximum Entropy and Linear SVM
- using three data sets:
  - 20 newsgroups
  - the Recreation sub-tree of the Open Directory
  - University Web pages from WebKB.



Multi-Label Classification

Multilabel Classification:

- there might be multiple label λ_i associated with each example
  - e.g., keyword assignments to texts

- Relevant labels R for an example
  - those that should be assigned to the example
- Irrelevant labels $I = L \setminus R$ for an example
  - those that should not be assigned to the examples

→ loss functions for classification can be adapted for multi-label classification

Hamming Loss

- average % of misclassified labels per example (R as I or I as R)

$$\text{HamLoss}(\hat{R}, R) = \frac{1}{c} \cdot |\hat{R} \Delta R|$$

- corresponds to 0/1 loss (accuracy, error) for classification problems

Multi-Label Classification Algorithms

Binary Relevance Classification (BR)

- Frequently taken approach to multi-label classification
- train one independent binary classifier for each label (one-vs-all)

- **Disadvantage:**
  - Each label is considered in isolation
  - no correlations between labels can be modeled

Multilabel Multiclass Perceptrons (MMP) (Crammer & Singer, JMLR-03)

- variant of BR that does not train perceptrons independently
- instead, they are trained that they collectively optimize the ranking of the labels

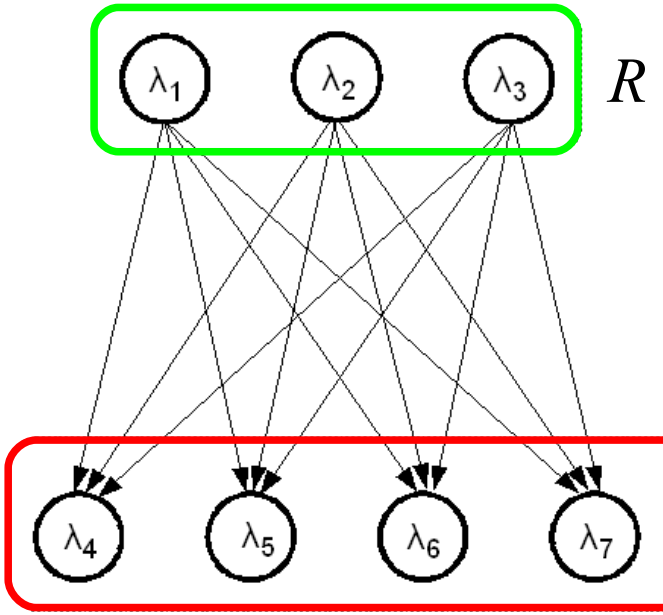
- **Disadvantage:**
  - additional methods are needed to convert rankings into multi-label classifications

Preference Learning

- Can we adapt Pairwise Classification to Multi-Label problems?
  - Multi-Class classification is a special case where only one label is relevant, i.e., for each example, $R = \{ \lambda_i \}$
- Relation to Preference Learning:
  - We can say that for each training example, we know that the relevant label is preferred over all other labels ($\lambda_i > \lambda_j$)
  - Pairwise classification learns a binary classifier $C_{i,j}$ for each pair of labels $\{ \lambda_i, \lambda_j \}$, which indicates whether $\lambda_i > \lambda_j$ or $\lambda_i < \lambda_j$
  - Predicted is the most preferred object (the one that receives the most votes from the binary classifiers)
  - Actually, we can produce a ranking of all labels according to the number of votes

Pairwise Multi-Label Ranking

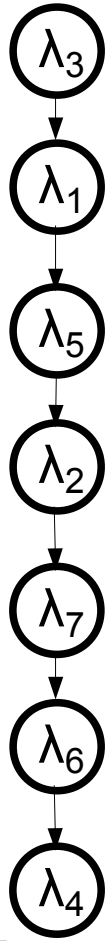
- Tranformation of Multi-Label Classification problems into preference learning problems is straight-forward



R relevant labels

$|R| \cdot |I|$ preferences

I irrelevant labels



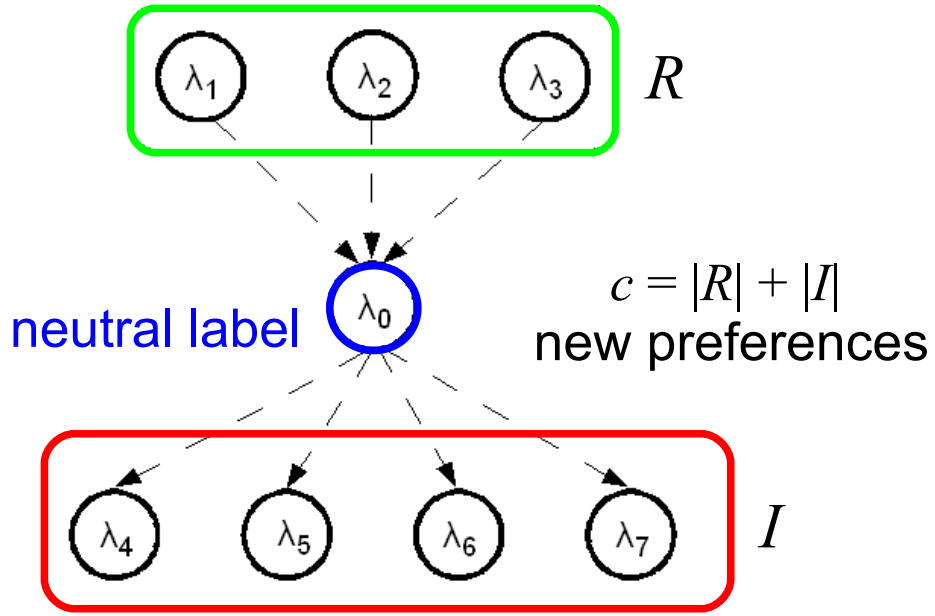
- at prediction time, the pairwise ensemble predicts a label ranking

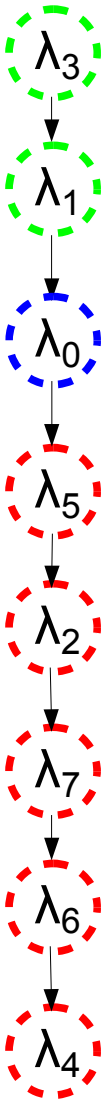
Problem:

- Where to draw boundary between relevant and irrelevant labels?

Calibrated Multi-Label PC

- Key idea:
  - introduce a neutral label into the preference scheme
  - the neutral label is
    - less relevant than all relevant classes
    - more relevant than all irrelevant classes

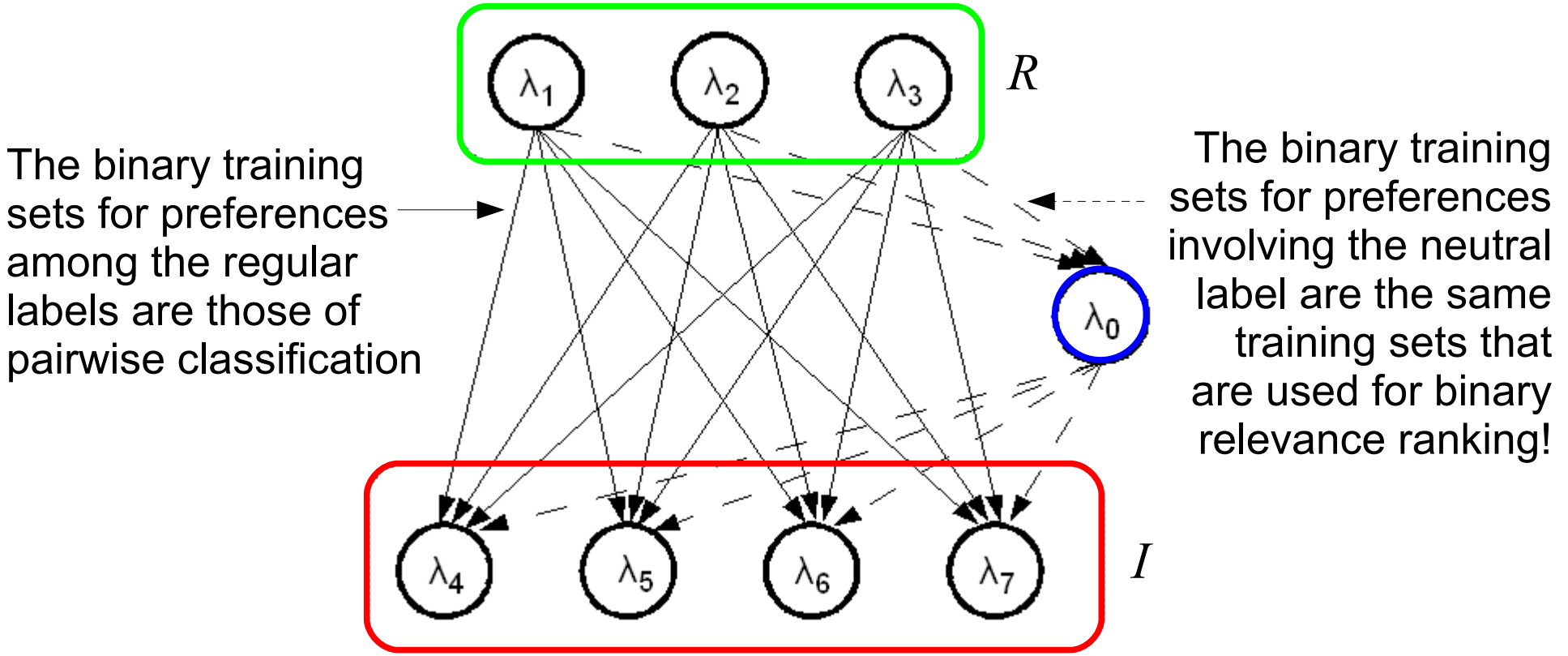




  - at prediction time, all labels that are ranked above the neutral label are predicted to be relevant

Calibrated Multi-Label PC

- Effectively, Calibrated Multi-Label Pairwise Classification (CMLPC) combines BR and PC



Results: Reuters CV1 Text Classification Task

- On REUTERS-CV1 text classification dataset
  - 804,414 Reuters newswire articles (535,987 train, 268,427 test)
  - 103 different labels, ≈ 3 labels per example
  - base learner: perceptrons

	Ranking Loss Functions			Multi-Label Loss Function
	AvgPrec	RankLoss	OneError	HamLoss
BR	88.23 %	2.529 %	5.02 %	1.26 %
MMP	92.82 %	0.687 %	3.75 %	—
MLPC	93.67 %	0.478 %	2.96 %	—
CMLPC	**93.81** %	**0.472** %	**2.90** %	1.03 %

- both pairwise methods outperform the one-against-all variants
  - BR is regular binary relevance ranking
  - MMP is an improvent that can optimize multi-label loss functions (Crammer & Singer, JMLR-03)
- the calibrated version outperforms the uncalibrated version
  - small difference, but statistically significant

EuroVOC Classification of EC Legal Texts

- Eur-Lex database
  - ≈ 20,000 documents
  - ≈ 4,000 labels in EuroVOC descriptor
  - ≈ 5 labels per document
- Pairwise modeling approach learns ≈8,000,000 perceptrons
  - memory-efficient dual representation necessary

Title and reference

Council Directive 91/250/EEC of 14 May 1991 on the legal protection of computer programs

Classifications

EUROVOC descriptor

- *data-processing law, computer piracy, copyright, software, approximation of laws*

Directory code

- 17.20.00.00 *Law relating to undertakings* / Intellectual property law

Subject matter

- *Internal market, Industrial and commercial property*

Text

COUNCIL DIRECTIVE of 14 May 1991 on the legal protection of computer programs (91/250/EEC)

THE COUNCIL OF THE EUROPEAN COMMUNITIES,

Having regard to the Treaty establishing the European Economic Community and in particular Article 100a thereof, . . .

- **Results:**
  - average precision of pairwise method is almost 50%

→ on average, the 5 relevant labels can be found within the first 10 labels of the ranking of all 4000 labels

  - one-against-all methods (BR and MMP) had a precision < 30%.
    - but were more efficient (even though the pairwise approach used less arithmetic operations)