Text Classification

- Characteristics of Machine Learning Problems
  - Example representation
  - Concept representation
- 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

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 <u>model</u> 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 <u>example</u> 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

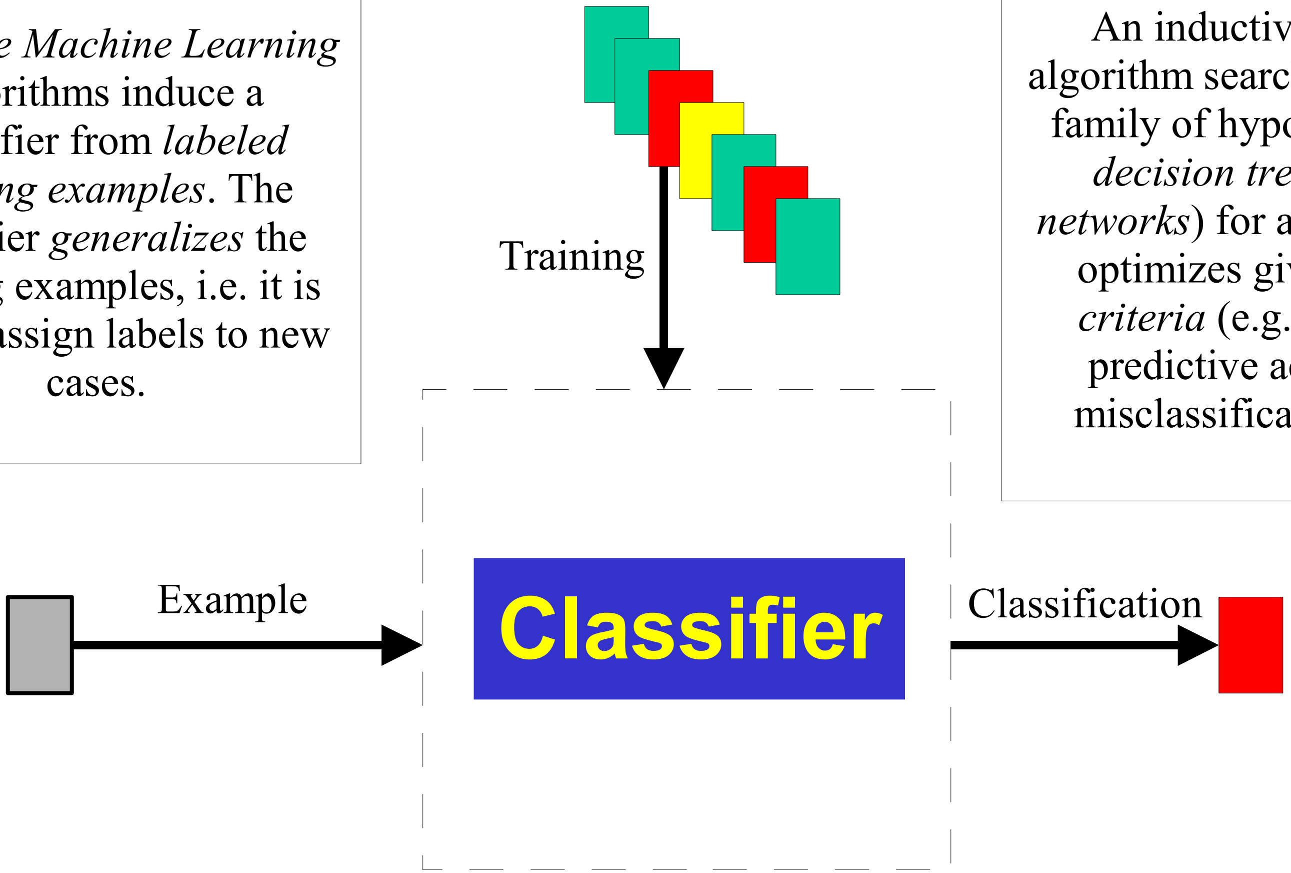
Text Categorization: Assign labels to each document

- 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”

After Manning, Stanford CS276A 2004

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.



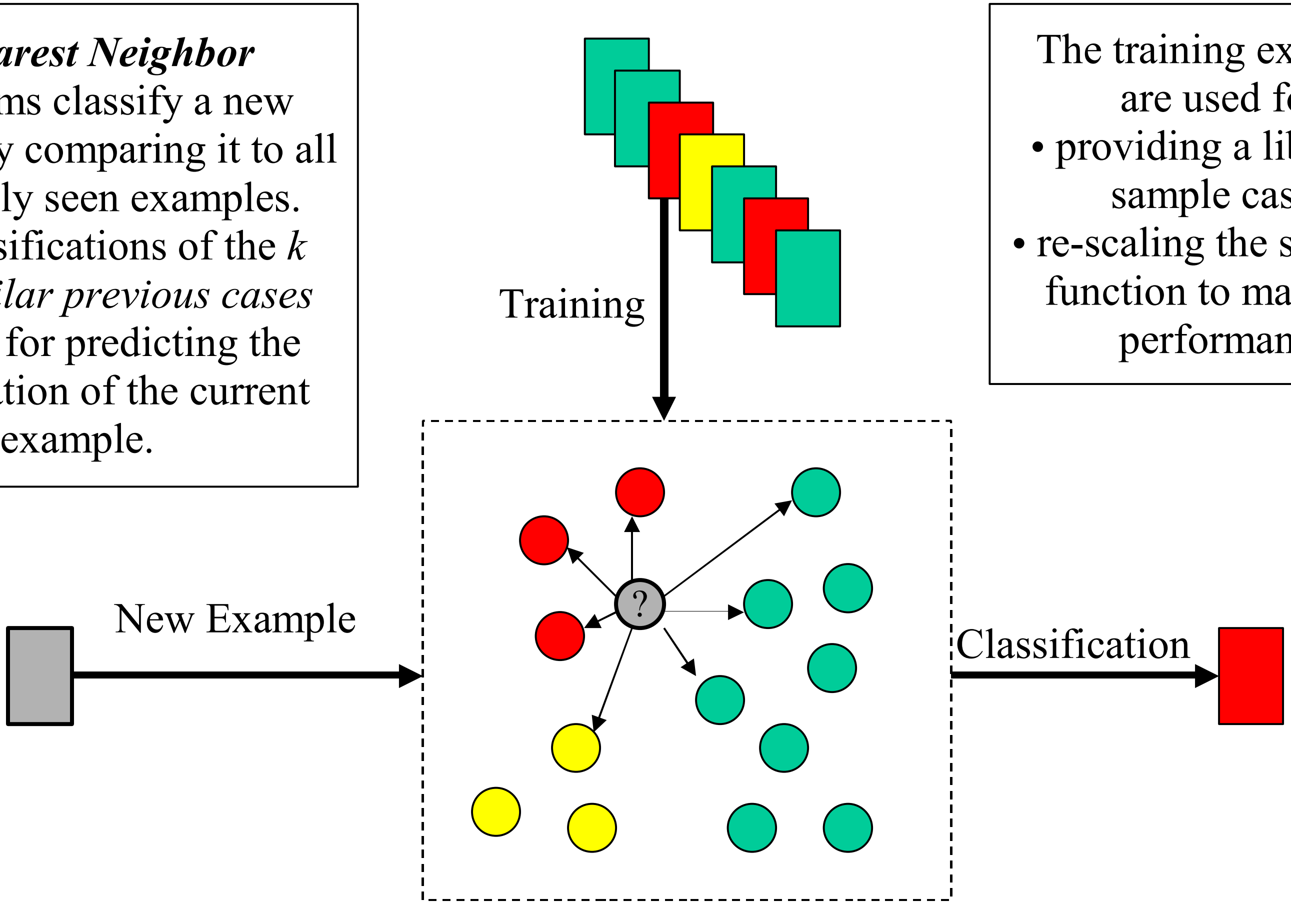
An inductive learning algorithm searches in a given family of hypotheses (e.g., *decision trees, neural networks*) for a member that optimizes given *quality criteria* (e.g., estimated predictive accuracy or misclassification costs).

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

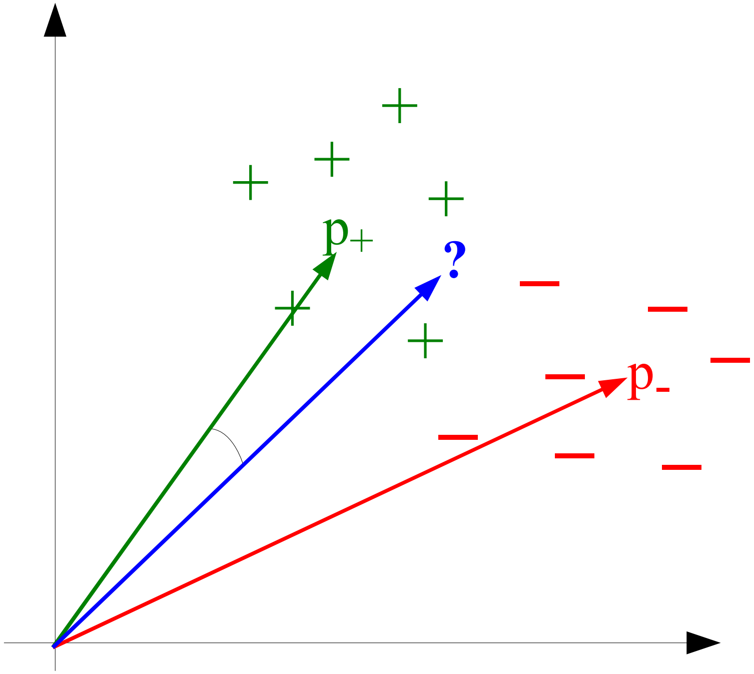
kNN Classifier

- To learn from a training set:
  - Store the training set
- To classify a new document :
  - Compute similarity of document vector 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

Rocchio Classifier

- based on ideas for Rocchio Relevance Feedback
- compute a prototype vector for each class
  - 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)



Probabilistic Document Model

- A document is a sequence of words (tokens, terms, features...)
  - $D=(t_1, t_2, \ldots, t_{|D|})$ where $t_j = w_{i_j} \in W$
  - Assume that a document D has been generated by repeatedly selecting a word w_{i_j} at random
- The probability that a word occurs in a document is dependent on the document's class c
  - $p(t_i|c) \neq p(t_i)$
- **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 $p(c|D)$ that document D belongs to class c

Simple Naïve Bayes Classifier for Text
(Mitchell 1997)

- Bayes Theorem:

$$p(c|D)=\frac{p(D|c)\,p(c)}{p(D)}$$

- $p(D)$ is only for normalization:
  - can be omitted if we only need a ranking of the class and not a probability estimate

$$p(D)=\sum_c p(D|c)\,p(c)$$

- Bayes Classifier:

$$c=arg\,max_c\ \ p(D|c)\,p(c)$$

  - predict class with largest posterior probability
- a document is a sequence of n words

$$p(D|c)=p(t_1,t_2,\ldots.t_n|c)$$

- 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

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

- Naïve Bayes Classifier
  - putting things together:

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

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 $n(D,w)$ times in document D

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

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

- 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

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

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

$$\binom{n}{i_{1,} i_{2,} \ldots i_k} = \frac{n!}{i_1! \cdot i_2! \cdot \ldots \cdot i_k!}$$

$\prod_{w \in D}$ iterates over vocabulary

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

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

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

  - $p(l = |D| \,|\, c)$ 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 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

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

Numerics of Naïve Bayes Models

- Multiply together a large number of small probabilities,
  - Result: extremely small probabilities as answers.
  - Solution: store all numbers as logarithms

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

  - to get back to the probabilities:

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

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

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

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

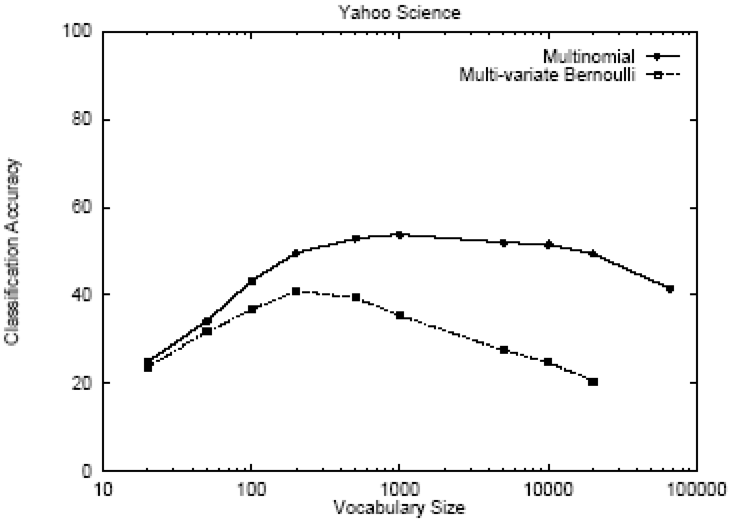


Figure 1: A comparison of event models for different vocabulary sizes on the Yahoo data set. Note that the multi-variate Bernoulli performs best with a small vocabulary and that the multinomial performs best with a larger vocabulary. The multinomial achieves higher accuracy overall.

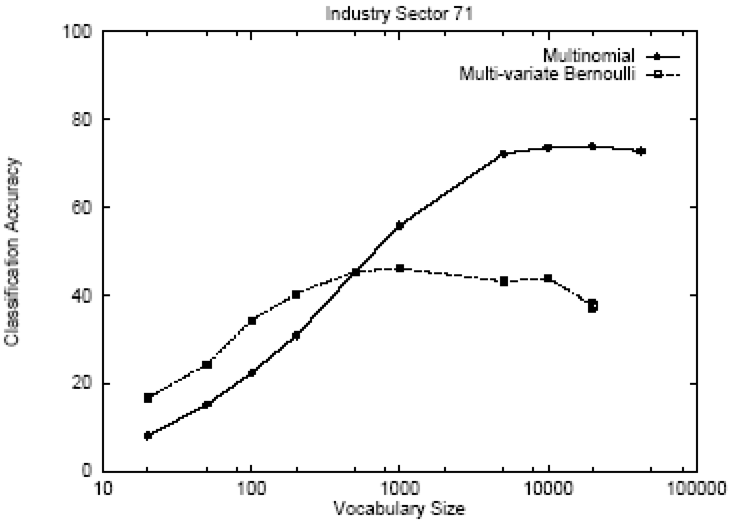


Figure 2: A comparison of event models for different vocabulary sizes on the Industry Sector data set. Note the same trends as seen in the previous figure.

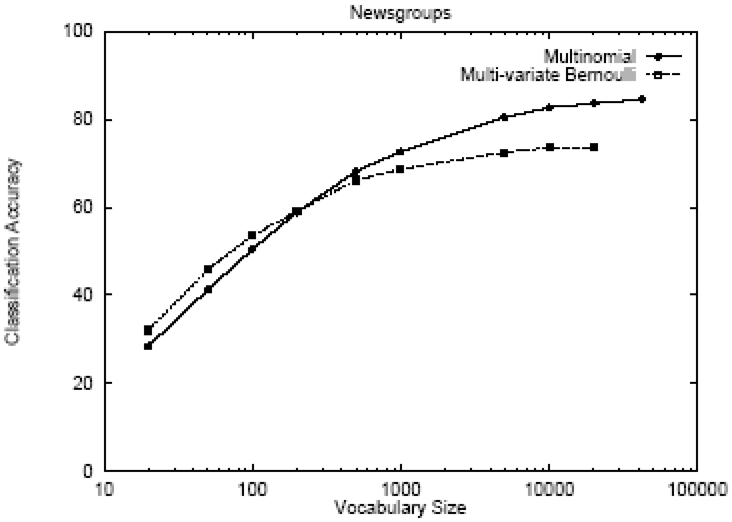


Figure 3: A comparison of event models for different vocabulary sizes on the Newsgroups data set. Here, both data sets perform best at the full vocabulary, but multinomial achieves higher accuracy.

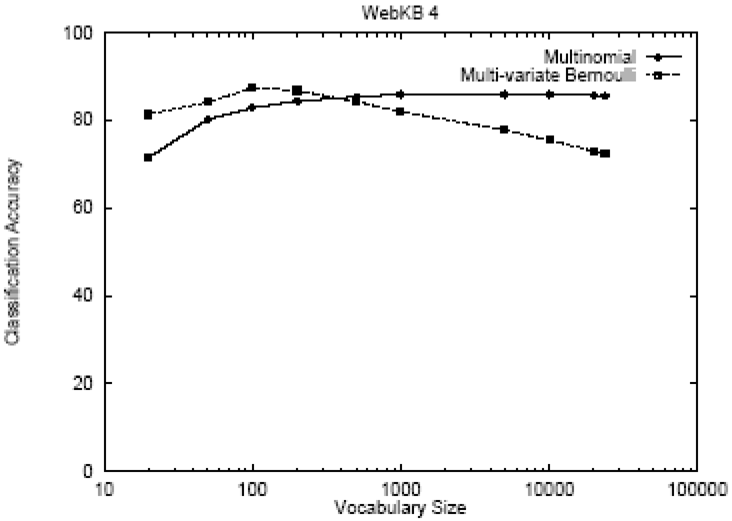


Figure 4: A comparison of event models for different vocabulary sizes on the WebKB data set. Here the two event models achieve nearly equivalent accuracies, but the multi-variate Bernoulli achieves this with a smaller vocabulary.

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

$$\log(p(D|c)) = \log\left(\frac{|D|}{\{n(D,w)_{w\in D}\}}\right) + \sum_{w\in D} n(D,w)\cdot\log p(w|c) = b + d\cdot\alpha_{NB}$$

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

d document vector consisting of term frequencies $n(D,w)$

Fitting a linear decision boundary

- Probabilistic approach
  - fixes the policy that $\alpha_{NB}(w)$ (w-th component of the linear discriminant) 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 α 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

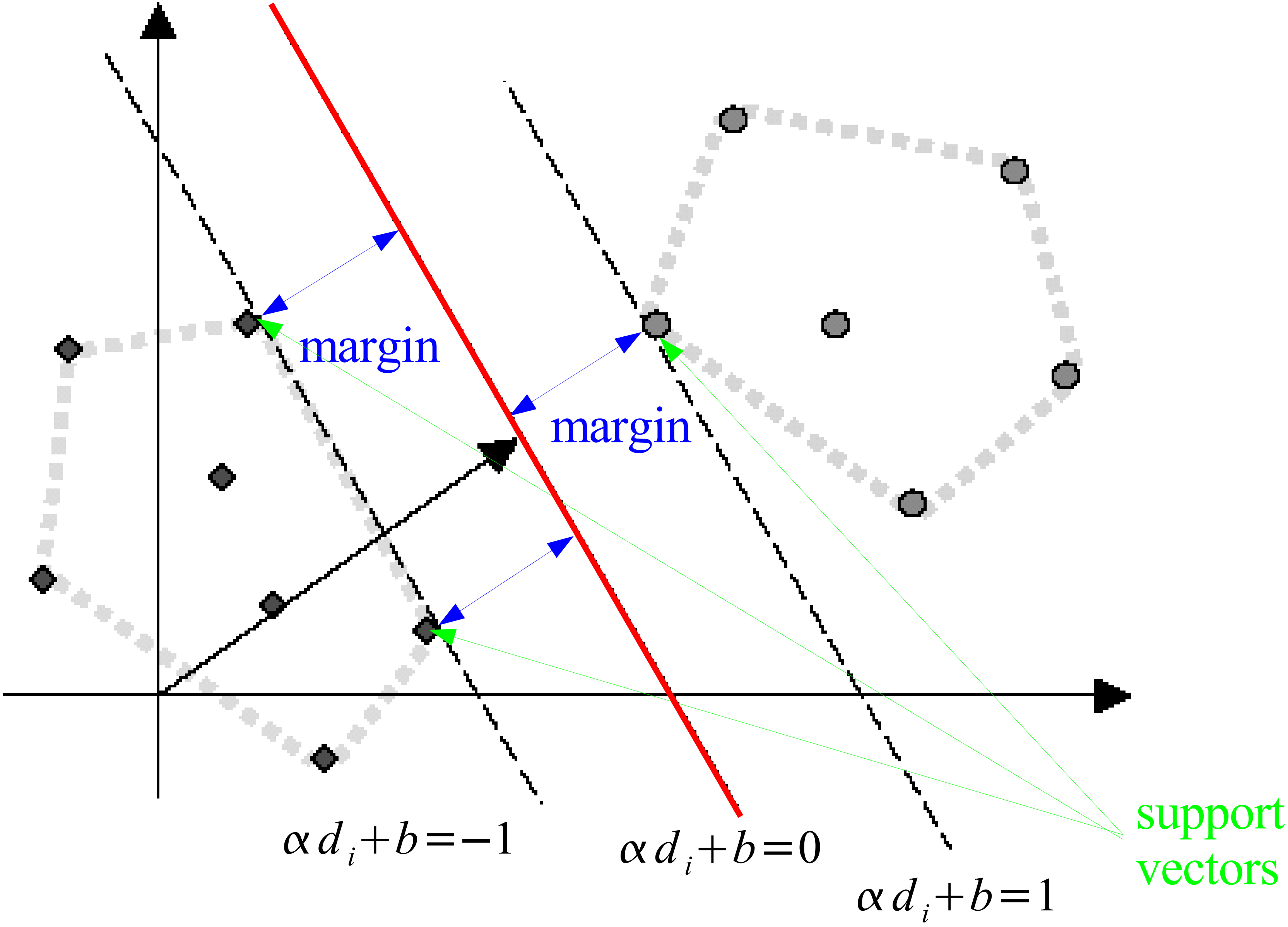
Support vector machines: Basic Idea

- Decision Boundary $\alpha \cdot d + b = 0$
  - Hyperplane that is close to many training data points has a greater chance of misclassifying test instances
  - A hyperplane which passes through a “no-man's land”, has lower chances of misclassifications
- Finding an optimal boundary
  - Goal: Find an α_{SVM} which maximizes the distance of any training point from the hyperplane
  - the closest points to the decision boundary are called **support vectors**
    - they will be put on the planes $\alpha \cdot d_{SV} + b = \pm 1$
  - their distance $1/\|\alpha\|$ to the hyperplane (the **margin**) should be maximized
  - thus: Minimize $\frac{1}{2} \alpha . \alpha \quad (= \frac{1}{2} \| \alpha \|^2)$

    subject to $c_i (\alpha . d_i + b) \geq 1 \ \forall\, i = 1, \ldots..n$

$$c_i = \begin{cases} +1 & if\ c = + \\ -1 & if\ c = - \end{cases}$$

Illustration of the SVM Optimization Problem



SVMs: non separable classes

- Classes in the training data not always separable.
- Introduce fudge variables ξ_i

Minimize $\frac{1}{2}\alpha . \alpha + C\sum_{i} \xi_i$

subject to $c_i(\alpha . d_i + b) \geq 1 - \xi_i \quad \forall\ i = 1,....,\ n.$

and $\xi_i \geq 0 \qquad \forall\ i = 1,........n$

Dual Representation and Kernel Trick

- The optimization problem can be formulated in a different way (the so-called *dual representation*)

dot product of document vectors

$$\text{Maximize} \qquad \sum_{\mathrm{i}} \lambda_{\mathrm{i}} - \frac{1}{2}\sum_{i,j} \lambda_i \lambda_j c_i c_j (d_i . d_j)$$

$$\text{subject to} \qquad \sum_{\mathrm{i}} \mathrm{c_i} \lambda_i = 0$$

$$\text{and} \qquad 1 \leq \lambda_{\mathrm{i}} \leq \mathrm{C} \qquad \forall\ \mathrm{i} = 1,\ldots\ldots..\mathrm{n}$$

- regular SVMs can only find a linear decision boundary
- Non-linearity can be achieved by replacing the dot-product $<d_i,d_j>$ with a function $k(d_i,d_j)$
  - k is also called a kernel
  - note relation to nearest neighbor algorithms!

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

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

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

- 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

Overfitting Avoidance

- 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)

Evaluation of Learned Models

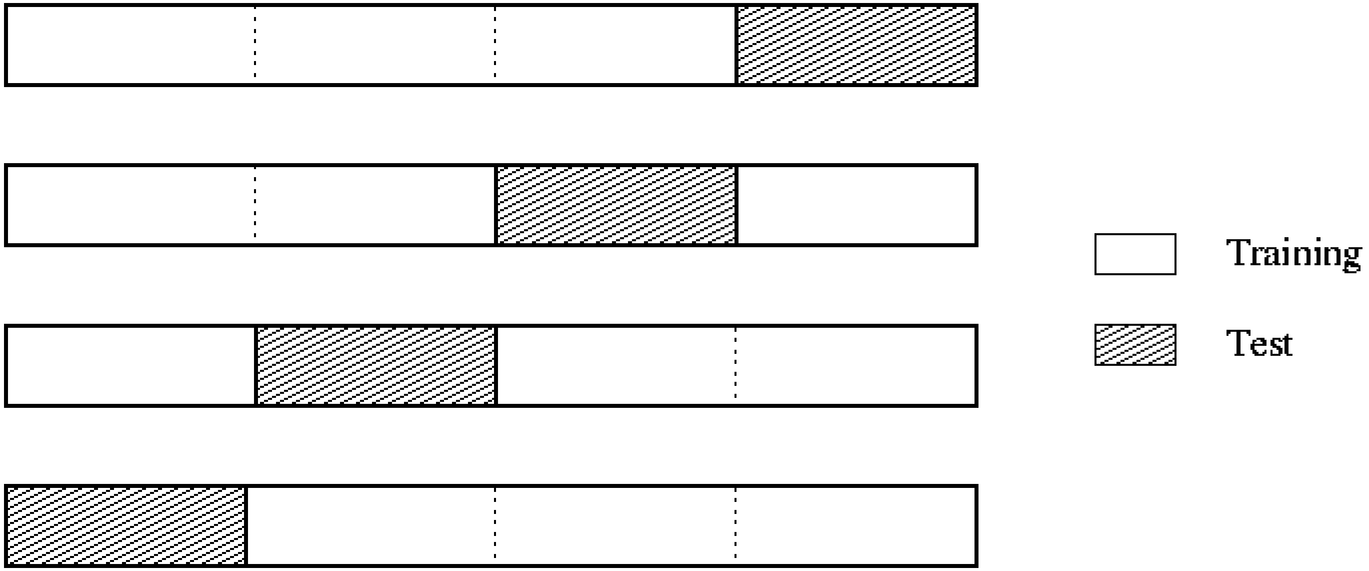
- Validation through experts
  - a domain experts evaluates the plausibility of a learned model
    - + subjective, time-intensive, costly
    - – but often the only option (e.g., clustering)
- Validation on data
  - evaluate the accuracy of the model on a separate dataset drawn from the same distribution 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
	$\bar{n}_A$	$\bar{n}_B$	$\bar{n}_C$	$\bar{n}_D$	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

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

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

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

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

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

Micro-Averaging

Predicted

True

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

Predicted

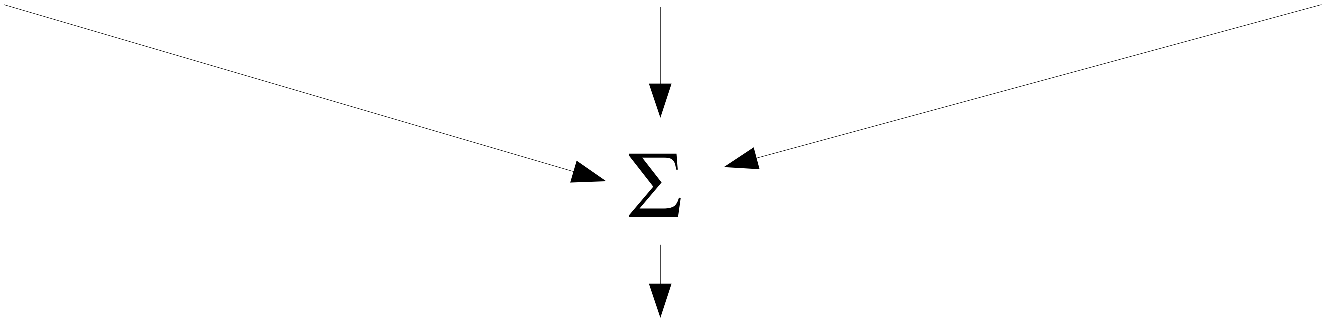
True

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

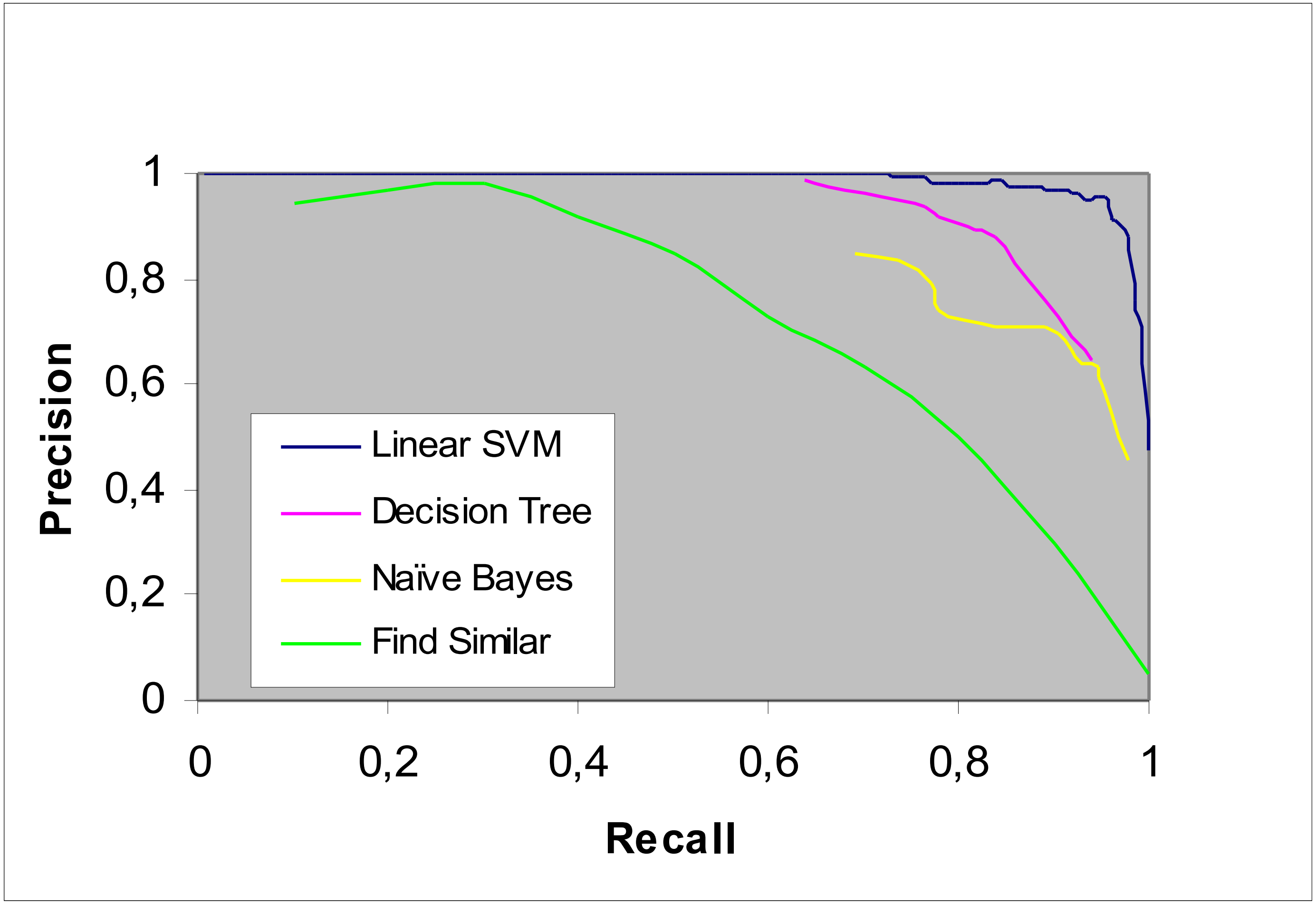
Benchmark Datasets

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

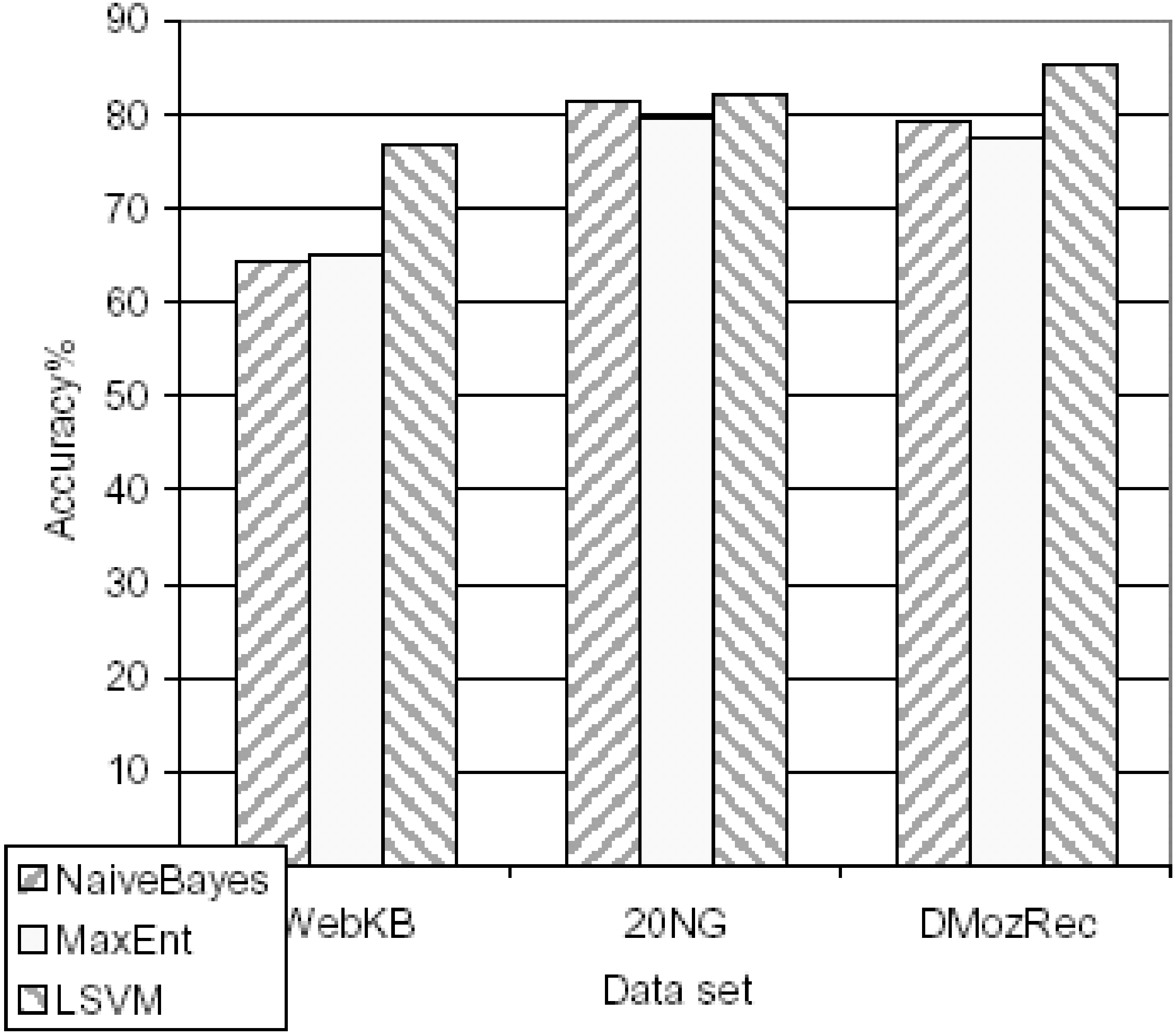
- Reuters-21578
  - 10700 labeled documents
  - 10% documents with multiple class labels
- OHSUMED
  - 348566 abstracts from medical journals
- 20 newsgroups
  - 18800 labeled USENET postings
  - 20 leaf classes, 5 root level classes
  - more recent 19 newsgroups
- WebKB
  - 8300 documents in 7 academic categories.
- Industry sectors
  - 10000 home pages of companies from 105 industry sectors
  - Shallow hierarchies of sector names

Sample Results

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



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



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, and University Web pages from WebKB.

Sample Results

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

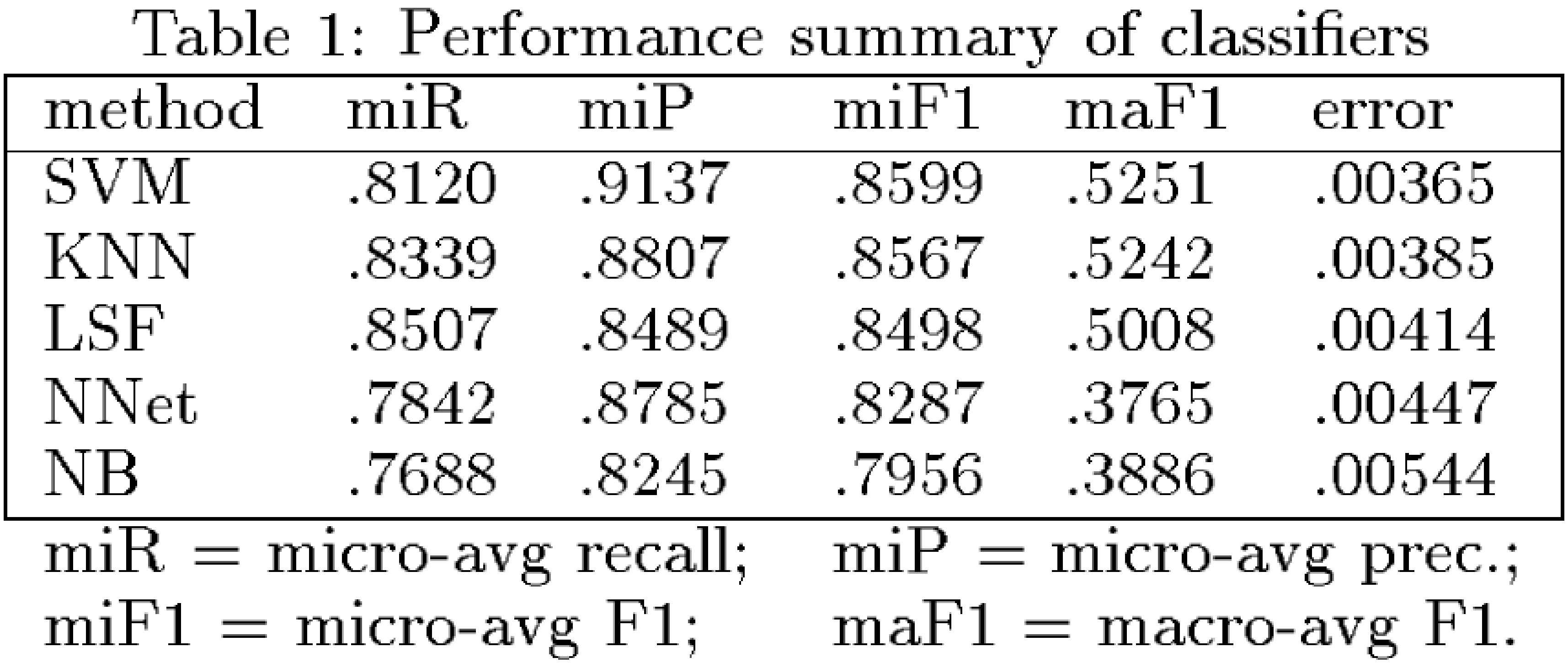


Table 1: Performance summary of classifiers

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