Novel representations and methods in text classification

Manuel Montes, Hugo Jair Escalante

Instituto Nacional de Astrofísica, Óptica y Electrónica, México.

http://ccc.inaoep.mx/~mmontesg/

http://ccc.inaoep.mx/~hugojair/

{*mmontesg, hugojair*}*@inaoep.mx*

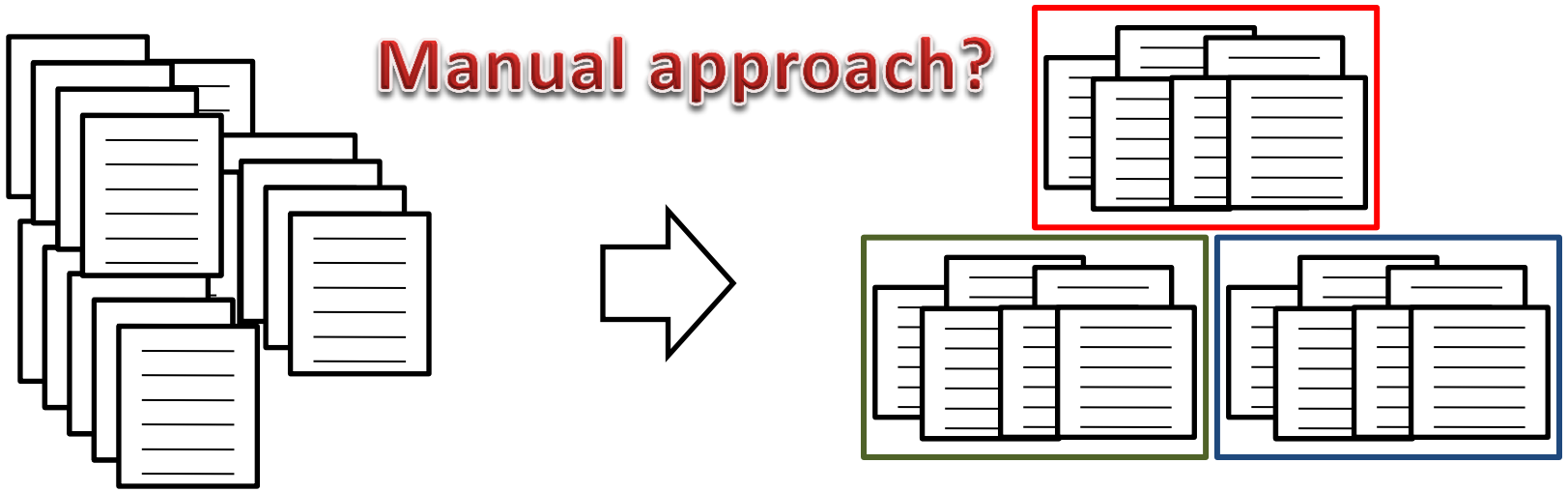
7th Russian Summer School in Information Retrieval
Kazan, Russia, September 2013

Novel represensations and methods in text classification

INTRODUCTION TO TEXT CLASSIFICATION

Text classification

- **Text classification** is the assignment of free-text documents to one or more predefined categories based on their content



Documents (e.g., news articles)

Categories/classes
(e.g., sports, religion, economy)

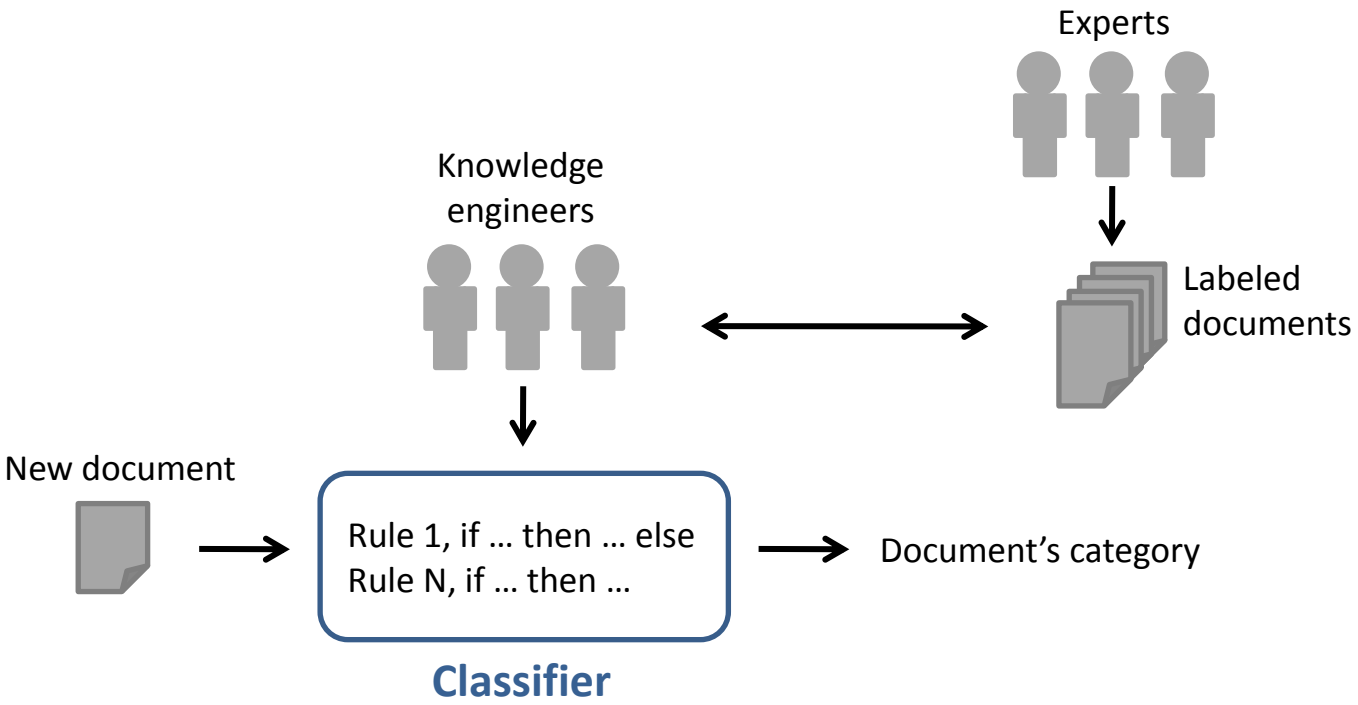
Manual classification

- **Very accurate** when job is done by experts
  - Different to classify news in general categories than biomedical papers into subcategories.
- But difficult and **expensive** to scale
  - Different to classify thousands than millions
- Used by Yahoo!, Looksmart, about.com, ODP, Medline, etc.

Ideas for building an automatic classification system?

How to define a classification function?

Hand-coded rule based systems



- Main approach in the 80s
- Disadvantage → knowledge acquisition bottleneck
  - too time consuming, too difficult, inconsistency issues

Example: filtering spam email

- **Rule-based classifier**

TABLE 1.1. *Average percentage of words or characters in an email message equal to the indicated word or character. We have chosen the words and characters showing the largest difference between* spam *and* email.

	george	you	your	hp	free	hpl	!	our	re	edu	remove
spam	0.00	2.26	1.38	0.02	0.52	0.01	0.51	0.51	0.13	0.01	0.28
email	1.27	1.27	0.44	0.90	0.07	0.43	0.11	0.18	0.42	0.29	0.01

Classifier 1

$$\text{if } (0.2 \cdot \%\texttt{you} - 0.3 \cdot \%\texttt{george}) > 0 \quad \text{then spam else email.}$$

Classifier 2

$$\text{if } (\%\texttt{george} < 0.6) \;\&\; (\%\texttt{you} > 1.5) \quad \text{then spam else email.}$$

Hastie et al. **The Elements of Statistical Learning,** 2007, Springer.

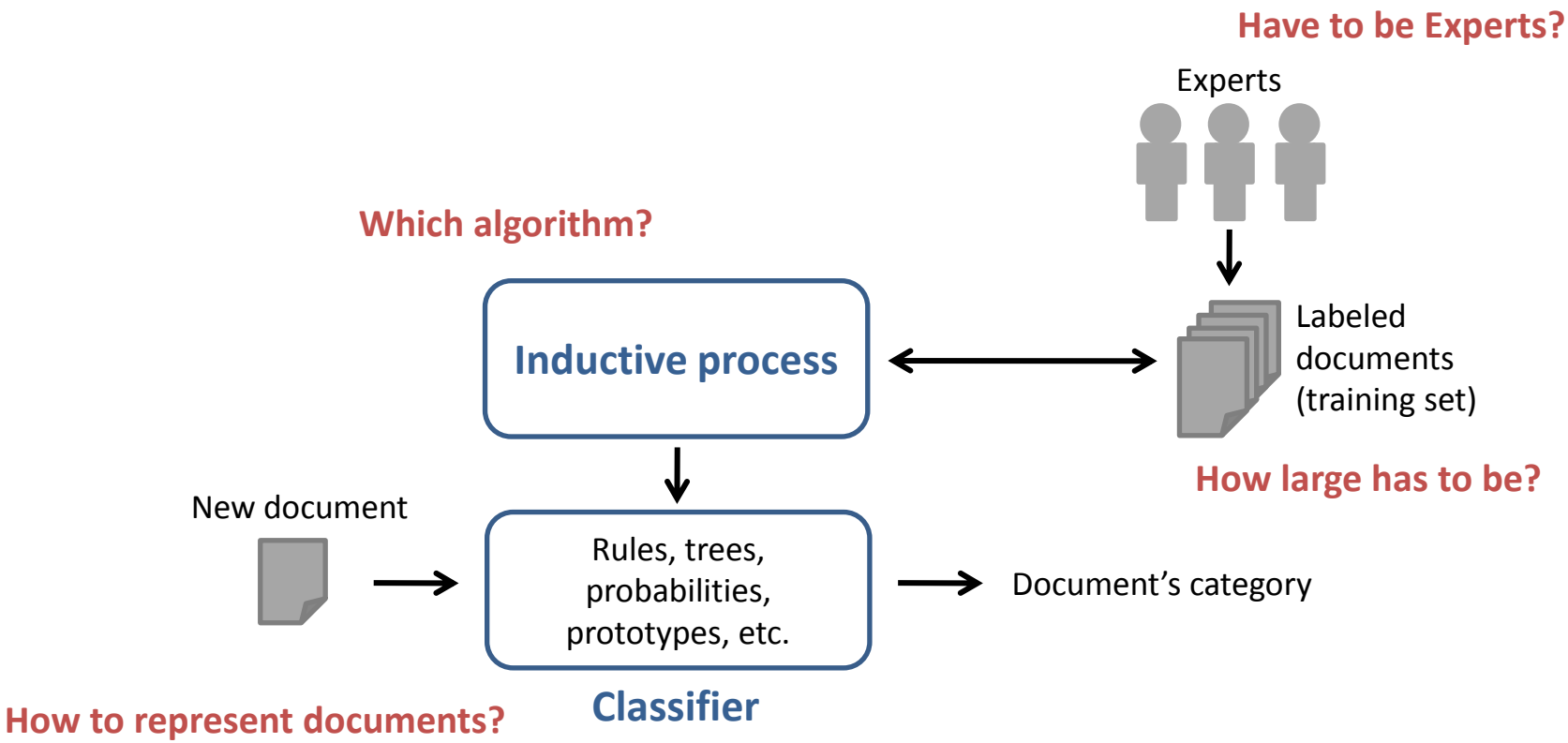
Machine learning approach (1)

- A general inductive process builds a classifier by learning from a set of preclassified examples.
  - Determines the characteristics associated with each one of the topics.

The general text categorization task can be formally defined as the task of approximating an unknown category assignment function $F : D \times C \rightarrow \{0, 1\}$, where D is the set of all possible documents and C is the set of predefined categories. The value of $F(d, c)$ is 1 if the document d belongs to the category c and 0 otherwise. The approximating function $M : D \times C \rightarrow \{0, 1\}$ is called a *classifier*, and the task is to build a classifier that produces results as "close" as possible to the true category assignment function F.

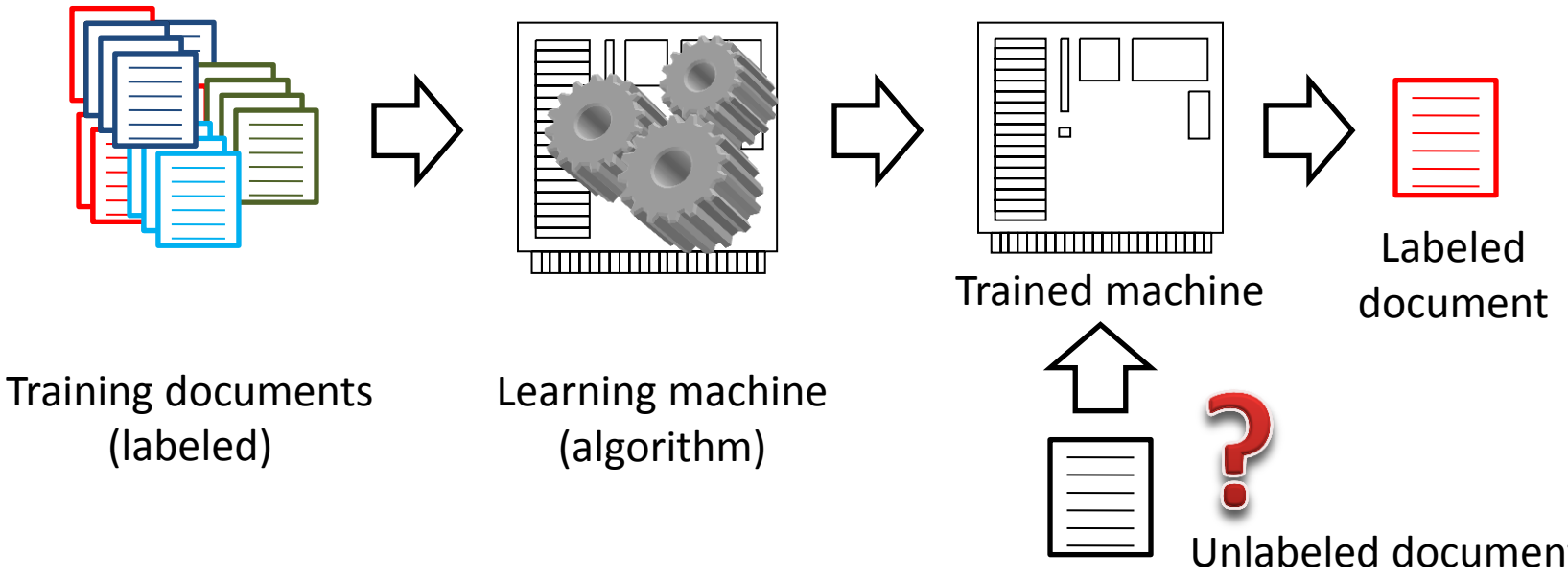
Ronen Feldman and James Sanger, The Text Mining Handbook

Machine learning approach (2)

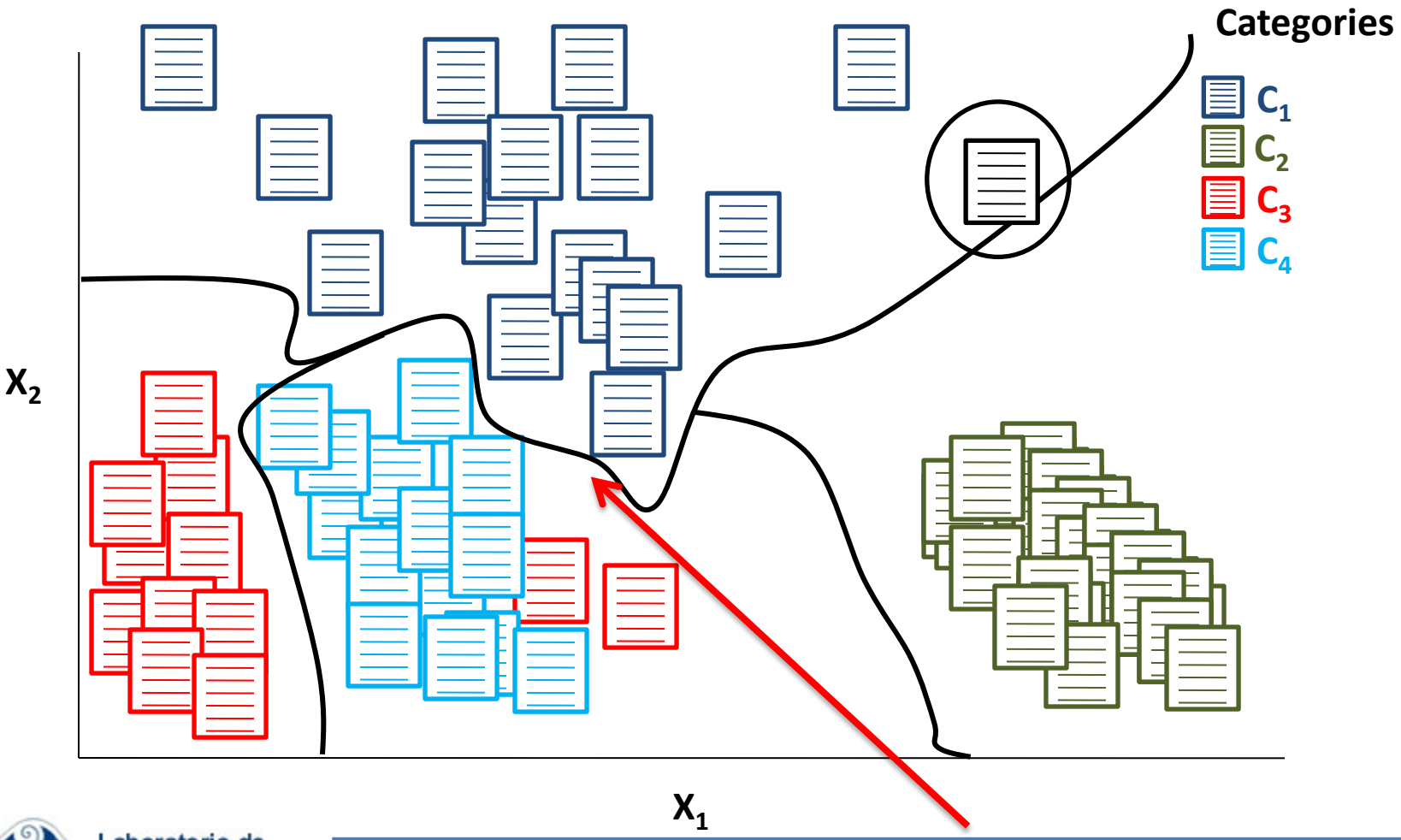


Machine learning approach (3): classification

- To learn a model able to make predictions regarding a variable of interest, using a set of other variables. Example: *text categorization*

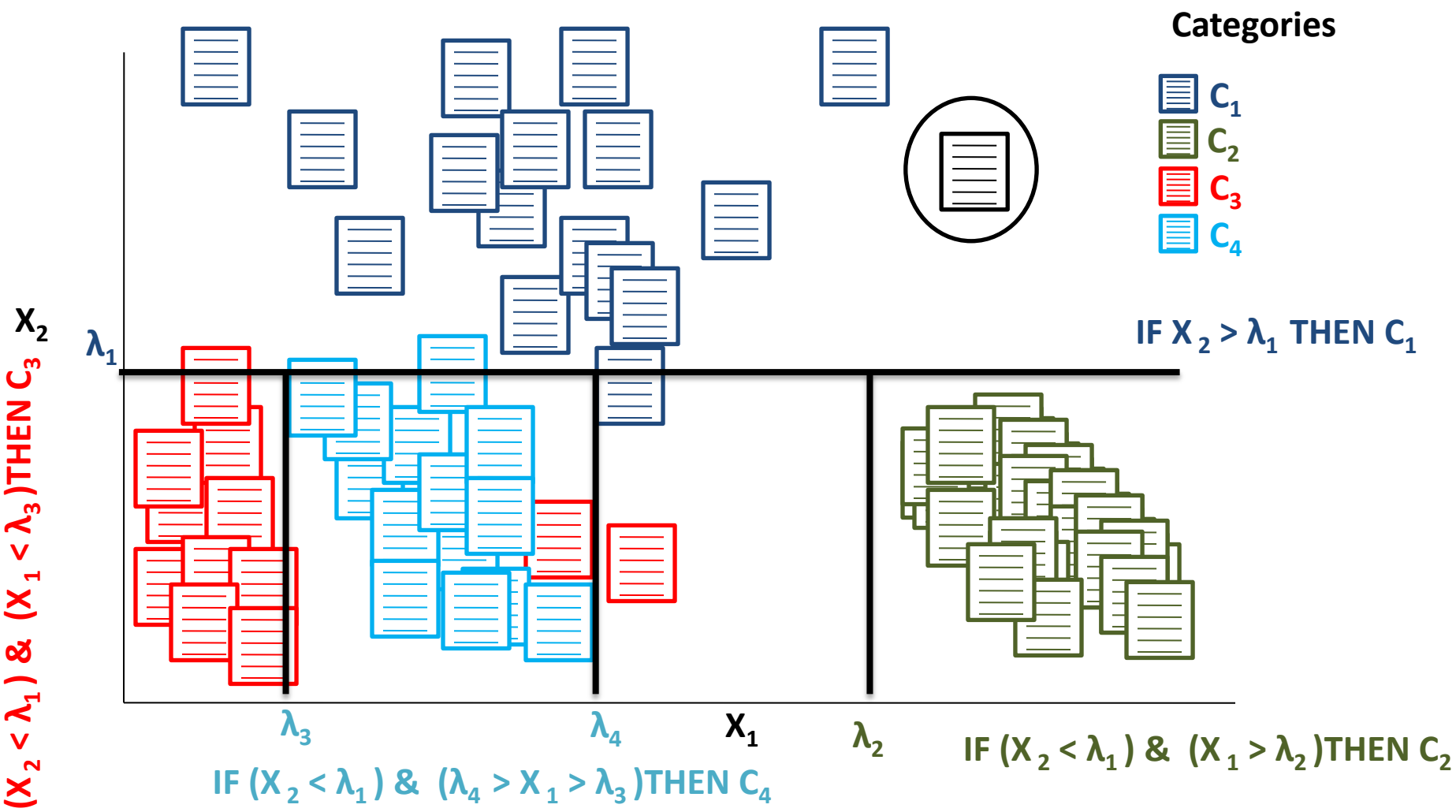


Machine learning approach (4): classification



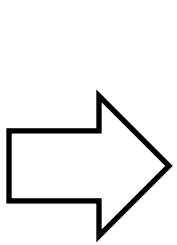
How to learn these functions?

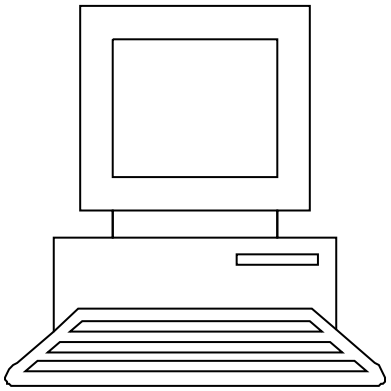
Machine learning approach (5): classification



Document representation

- Represent the content of *digital* documents in a way that they can be processed by a computer





Before representing documents: Preprocessing

- Eliminate information about style, such as html or xml tags.
  - For some applications this information may be useful. For instance, only index some document sections.

- Remove stop words
  - Functional words such as articles, prepositions, conjunctions are not useful (do not have an own meaning).

- Perform stemming or lemmatization
  - The goal is to reduce inflectional forms, and sometimes derivationally related forms.

am, are, is → be
car, cars, car's → car

Document representation

- Transform documents, which typically are strings of characters, into a representation suitable for the learning algorithm:
  - **Codify/represent/transform documents into a vector representation**

- The most common used document representation is the *bag of words (***BOW***)* approach
  - Documents are represented by the set of words that they contain
  - Word order is not captured by this representation
  - There is no attempt for understanding their content
  - The vocabulary of all of the different words in all of the documents is considered as the base for the vector representation

Document representation

Terms in the vocabulary
(*Basic units expressing document's content*)

	t_1	t_1	…	$t_{\|V\|}$
d_1				
d_2				
:		$w_{i,j}$		
d_m				

Documents in the corpus
(*one vector/row per document*)

V: Vocabulary from the collection *(i.e., et of all different words that occur in the corpus)*

Weight indicating the contribution of word *j* in document *i*.

Which words are good features?
How to select/extract them?
How to compute their weights?

Document representation

- **Simplest BOW-based representation:** Each document is represented by a binary vector whose entries indicate the presence/absence of terms from the vocabulary (**Boolean/binary weighting**)

Document	Content
Syllabus.txt	Advanced topics on text mining
Evaluation.txt	Homework, reports (text)
Students.txt	Graduate (Advanced)
Description.txt	Studying topics on text mining

Obtain the BOW representation with Boolean weighting for these documents

Term weighting
[extending the Boolean BOW]

- **Two main ideas:**
  - The importance of a term increases proportionally to the number of times it appears in the document.
    - It helps to *describe* document's content.

Local

  - The general importance of a term decreases proportionally to its occurrences in the entire collection.
    - Common terms are not good to *discriminate* between different classes

Global

Does the order of words matters?

Term weighting – main approaches

- **Binary weights:**
  - $w_{i,j}$ = 1 iff document d_i contains term t_j , otherwise 0.

- **Term frequency (tf):**
  - $w_{i,j}$ = (no. of occurrences of t_j in d_i)

- **tf x idf weighting scheme:**
  - $w_{i,j} = tf(t_j, d_i) \times idf(t_j)$, where:
    - $tf(t_j, d_i)$ indicates the ocurrences of t_j in document d_i
    - $idf(t_j) = \log [N/df(t_j)]$, where $df(t_j)$ is the number of documets that contain the term t_j.

These methods do not use the information of the classes, why?

Term weighting – main approaches

- **Binary weights:**
  - $w_{i,j}$ = 1 iff document d_i contains term t_j , otherwise 0.

- **Term frequency (tf):**
  - $w_{i,j}$ = (no. of occurrences of t_j in d_i)

- **tf x idf weighting scheme:**
  - $w_{i,j} = tf(t_j, d_i) \times idf(t_j)$, where:
    - $tf(t_j, d_i)$ indicates the ocurrences of t_j in document d_i
    - $idf(t_j) = \log [N/df(t_j)]$, where $df(t_j)$ is the number of documets that contain the term t_j.

$$w_{i,j} = \frac{w_{i,j}}{\sqrt{\sum_{k=1}^{M} (w_{i,k})^2}}$$

Normalization?

Extended document representations

- Document representations that capture information not considered by the BOW formulation

- Examples:
  - Based on distributional term representations
  - Locally weighted bag of words
  - Bag of concepts
  - Concise semantic analysis
  - Latent semantic indexing
  - Topic modeling
  - …

The topic of this course

Dimensionality issues

- A central problem in text classification is the high dimensionality of the feature space.
  - There is one dimension for each unique word found in the collection → can reach hundreds of thousands
  - Processing is extremely costly in computational terms
  - Most of the words (features) are irrelevant to the categorization task

How to select/extract relevant features?

How to evaluate the relevancy of the features?

Out of the scope of this course

What is a learning algorithm?

- A function:

$$f : \Re^d \rightarrow C \qquad C = \{C_1, \ldots, C_K\}$$

$$f : (\Re^d, C) \rightarrow \{0,1\}$$

- Given:

$$D = \{(\mathbf{x}_i, y_i)\}_{1,\ldots,N}$$

$$\mathbf{x}_i \in \Re^d ; y_i \in C$$

Out of the scope of this course

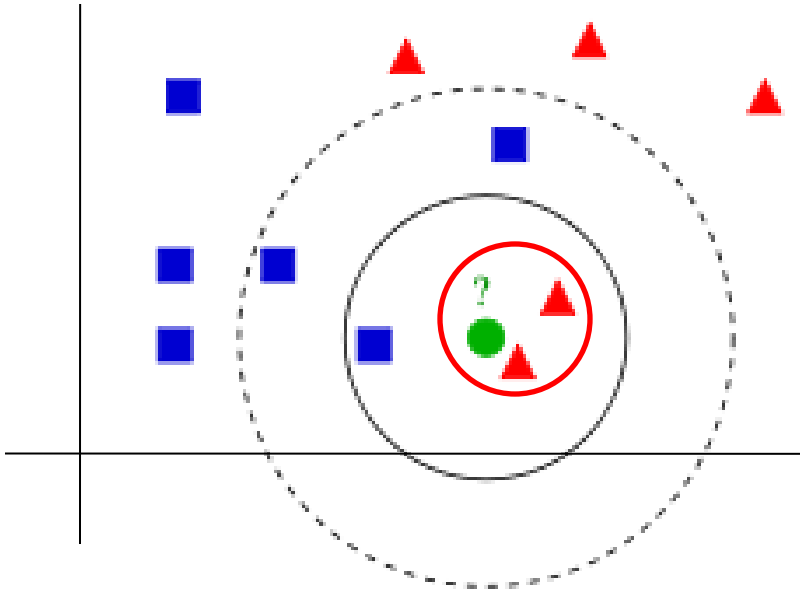
Classification algorithms

- Popular classification algorithms for TC are:
  - **K-Nearest Neighbors**
    - *Example-based* approach
  - **Centroid-based classification**
    - *Prototype*-based approach
  - **Support Vector Machines**
    - *Kernel*-based approach
  - **Naïve Bayes**
    - *Probabilistic* approach

KNN: K-nearest neighbors classifier

Positive examples

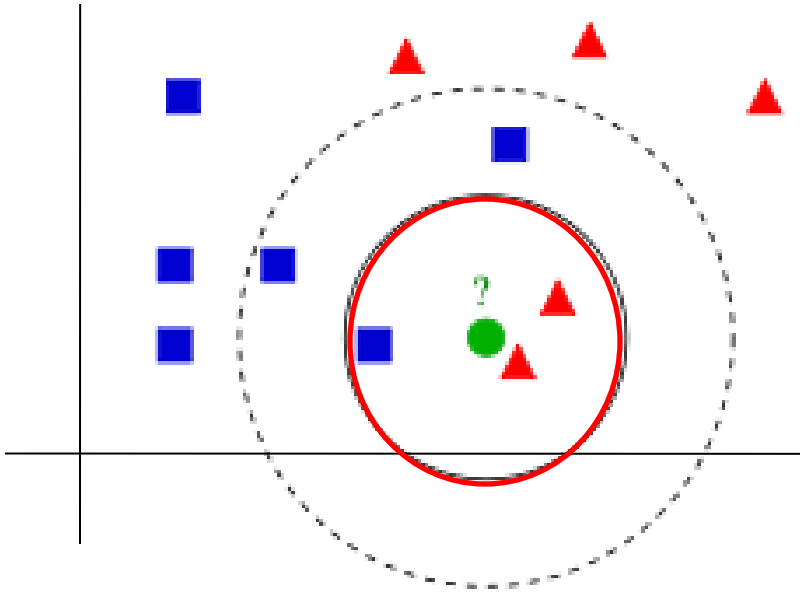
Negative examples



KNN: K-nearest neighbors classifier

Positive examples

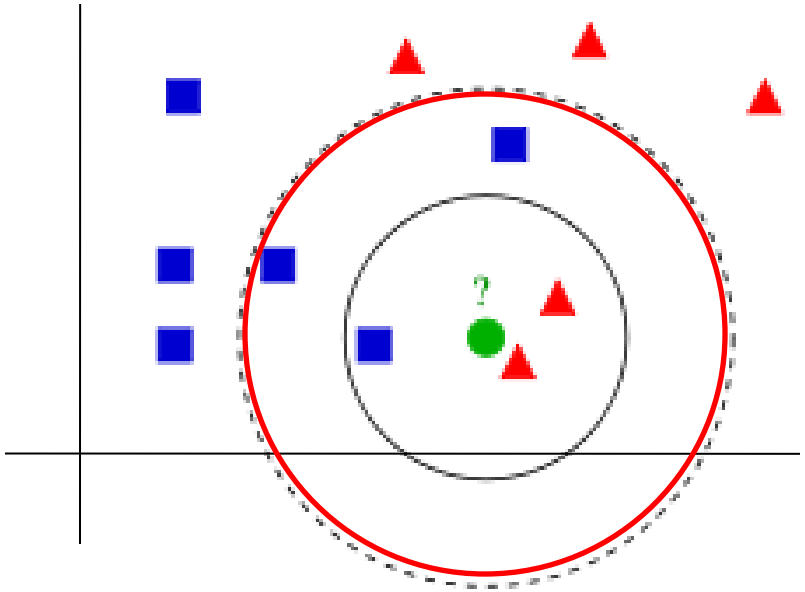
Negative examples



KNN: K-nearest neighbors classifier

Positive examples

Negative examples



KNN – the algorithm

- Given a new document *d*:
  1. Find the *k* most similar documents from the training set.
     - Common similarity measures are the cosine similarity and the Dice coefficient.
  2. Assign the class to *d* by considering the classes of its *k* nearest neighbors
     - Majority voting scheme
     - Weighted-sum voting scheme

Common similarity measures

- Dice coefficient

$$s(d_i, d_j) = \frac{2\sum_{k=1}^{n}(w_{ki} \times w_{kj})}{\sum_{k=1}^{m} w_{ki}^2 + \sum_{k=1}^{m} w_{kj}^2} \qquad s(A, B) = \frac{2|A \cap B|}{|A| + |B|}$$

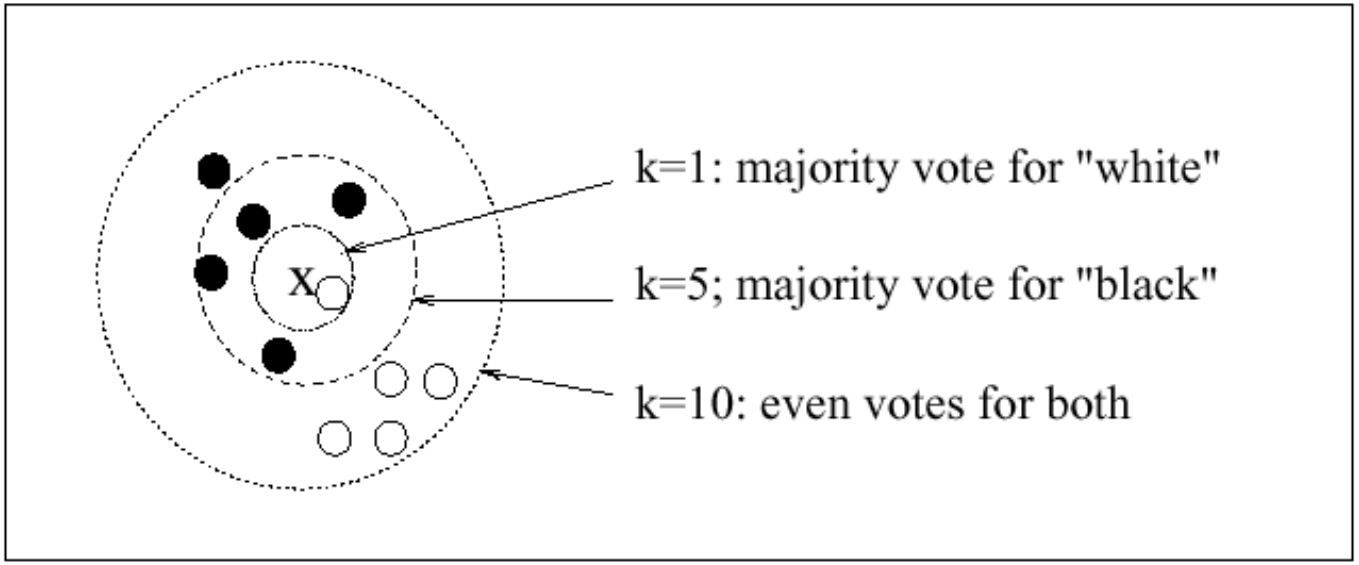
- Cosine measure

$$s(d_i, d_j) = \frac{\sum_{k=1}^{n}(w_{ki} \times w_{kj})}{\sqrt{\sum_{k=1}^{m} w_{ki}^2} \times \sqrt{\sum_{k=1}^{m} w_{kj}^2}} \qquad s(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \times \|B\|}$$

w_{ki} indicates the weight of word *k* in document *i*

Selection of *K*

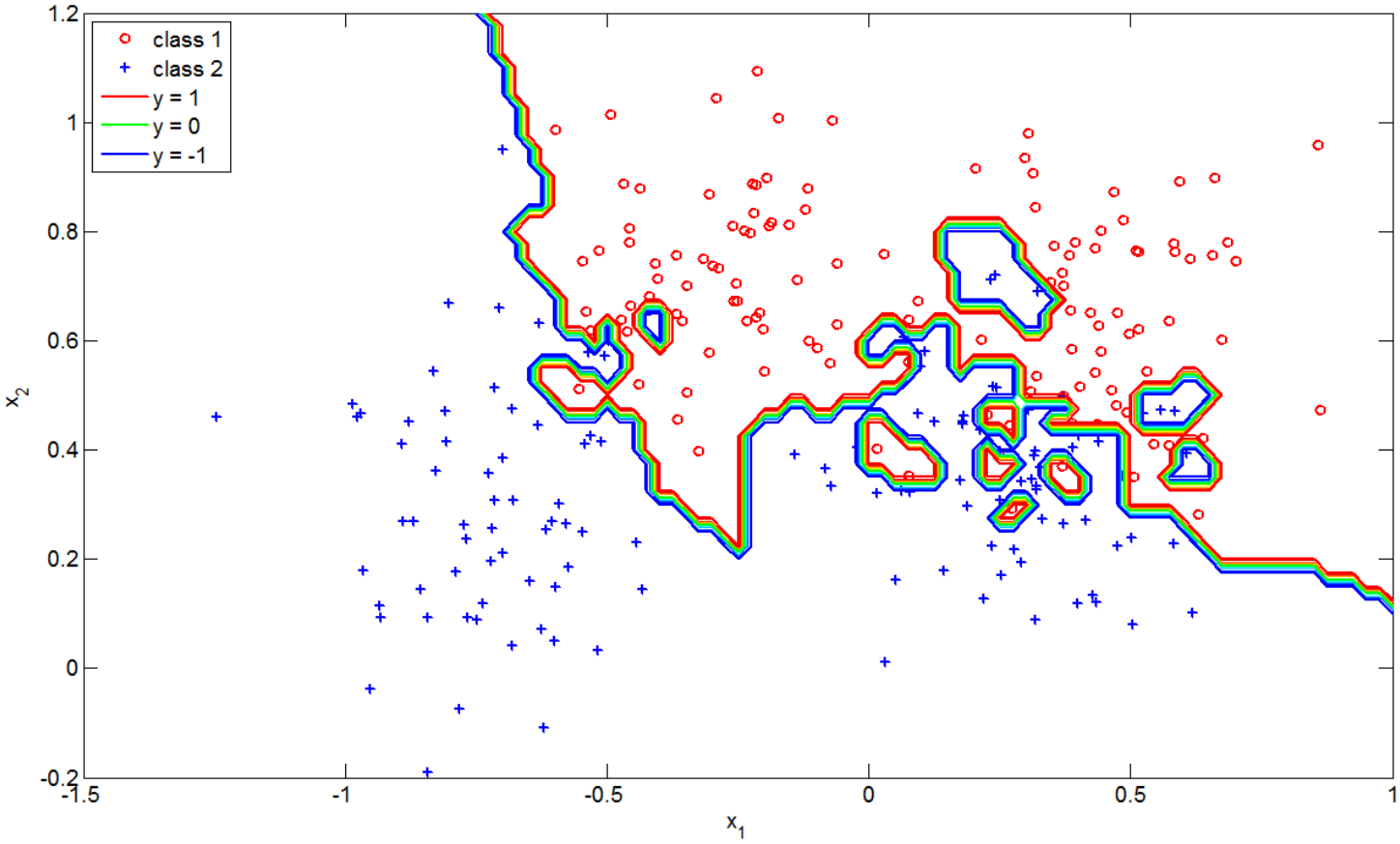
K-Nearest Neighbor using a *majority* voting scheme



How to select a good value for *K*?

Decision surface of KNN

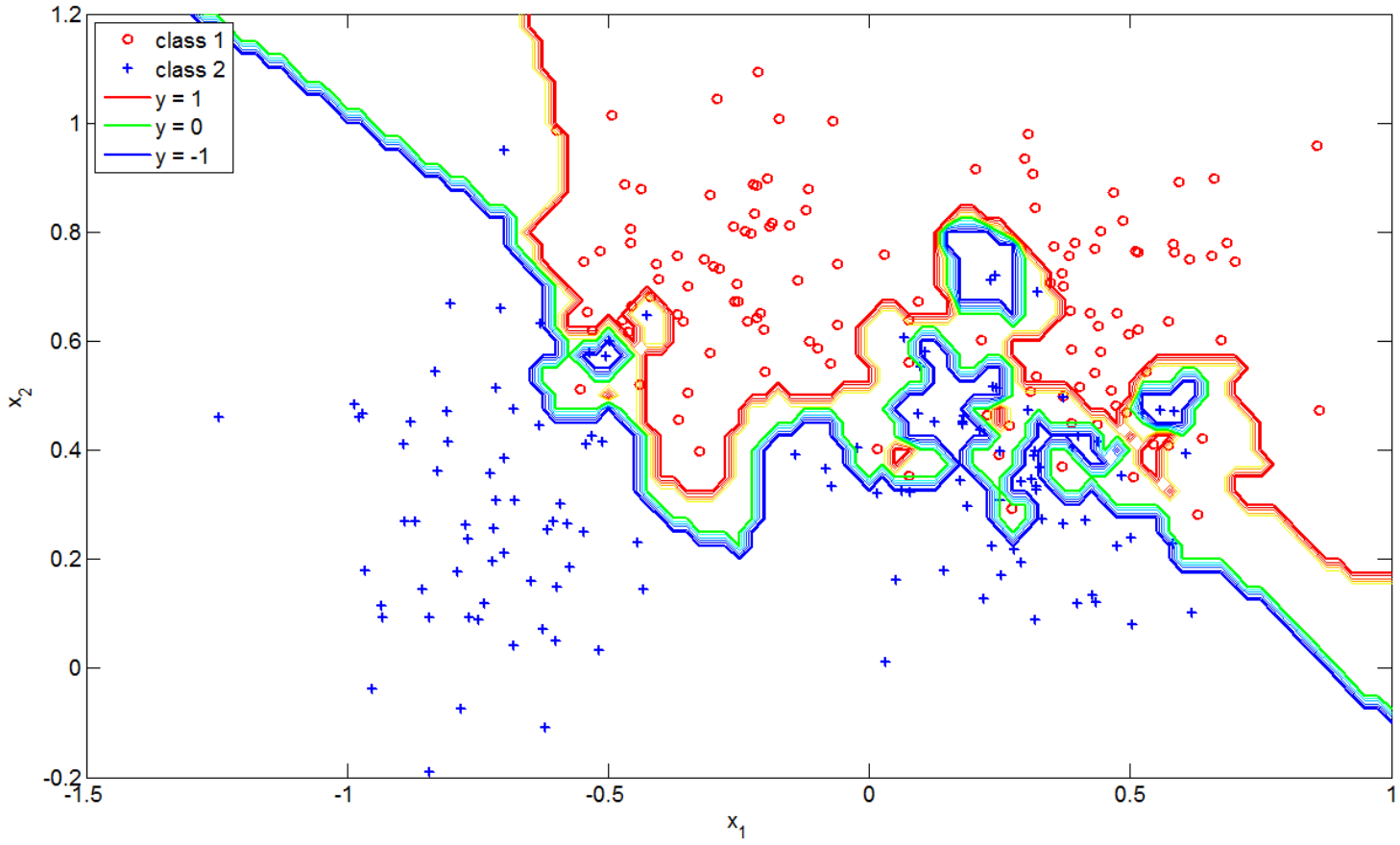
http://clopinet.com/CLOP



K=1

Decision surface of KNN

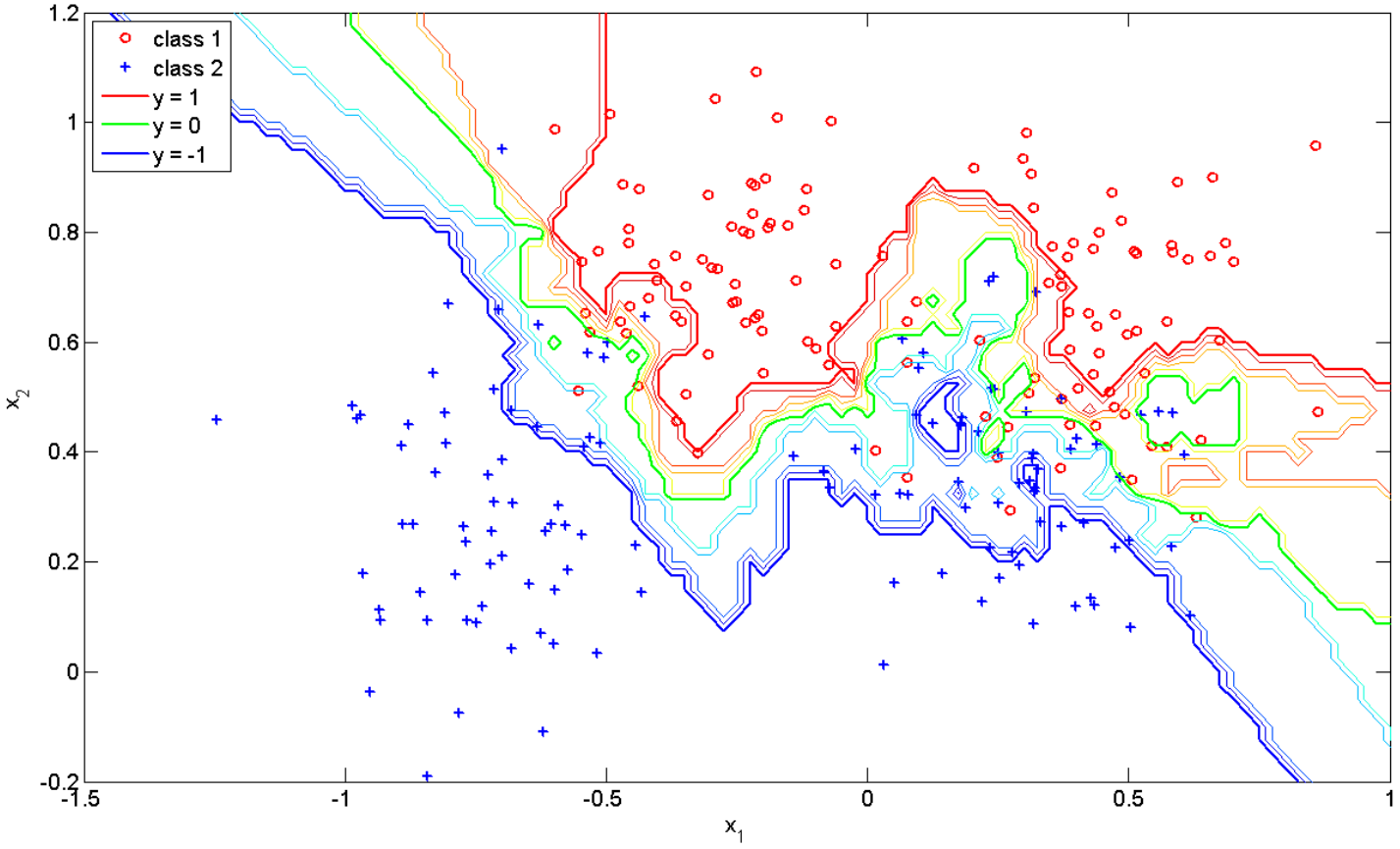
http://clopinet.com/CLOP



K=2

Decision surface of KNN

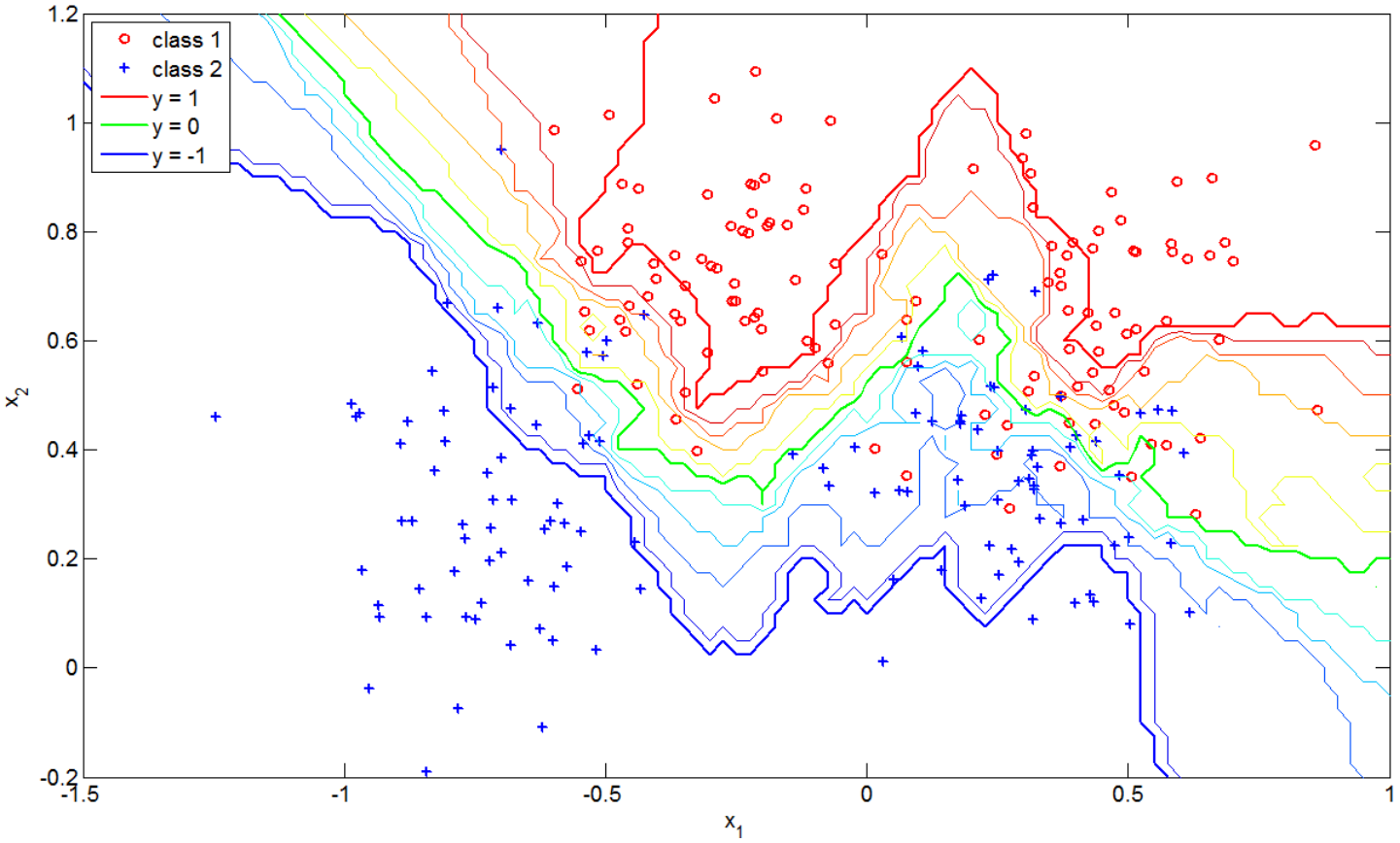
http://clopinet.com/CLOP



K=5

Decision surface of KNN

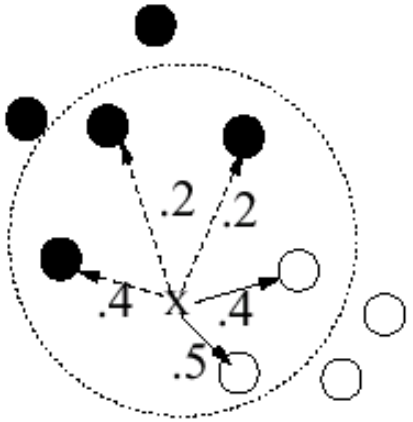
http://clopinet.com/CLOP



K=10

The weighted-sum voting scheme

k-NN using a weighted-sum voting scheme



kNN (k = 5)

Assign "white" to x because the weighted sum of "whites" is larger then the sum of "blacks".

Each neighbor is given a weight according to its nearness.

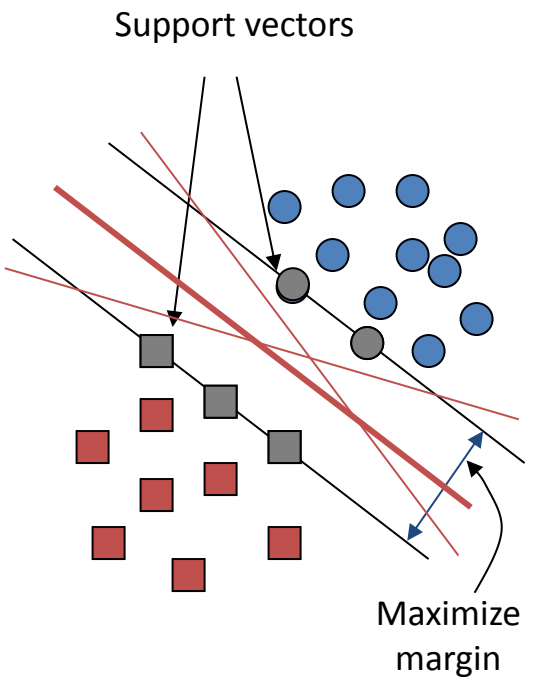
Other alternatives for computing the weights?

KNN - comments

- One of the best-performing text classifiers.
- It is robust in the sense of not requiring the categories to be linearly separated.
- The major drawback is the computational effort during classification.
- Other limitation is that its performance is primarily determined by the choice of *k* as well as the distance metric applied.

Support vector machines (SVM)

- A binary SVM classifier can be seen as a hyperplane in the feature space separating the points that represent the positive from negative instances.
  - SVMs selects the hyperplane that maximizes the *margin* around it.
  - Hyperplanes are fully determined by a small subset of the training instances, called the *support vectors.*



Support vector machines (SVM)

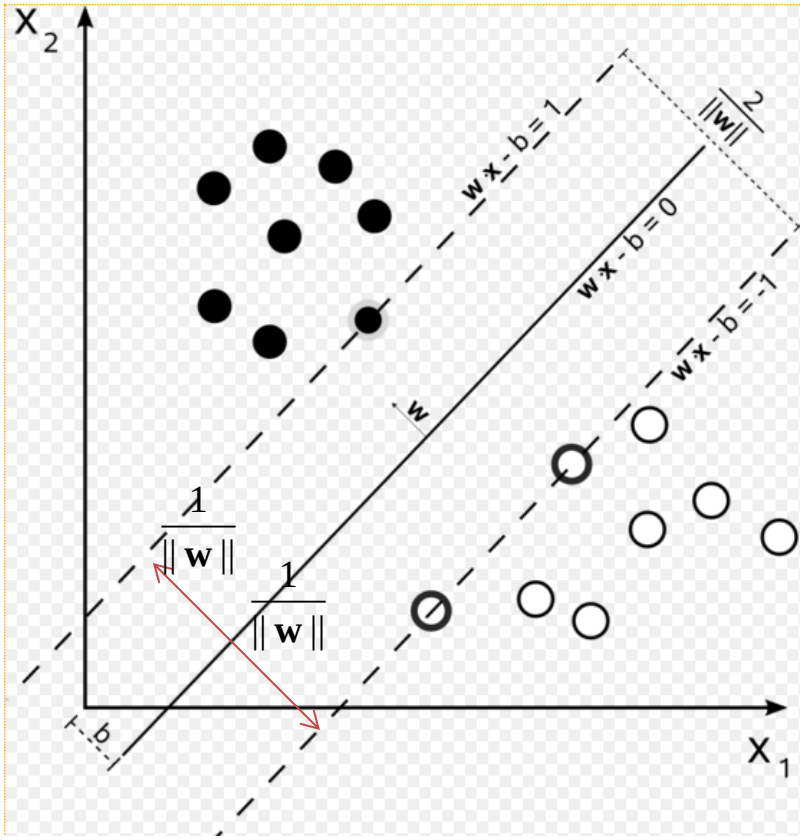
- When data are linearly separable we have:

$$\min \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

Subject to:

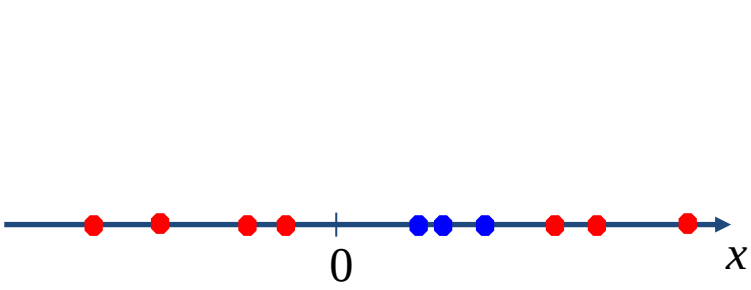
$$y_i(\mathbf{w}^T\phi(\mathbf{x}_i)+b) \geq 1$$

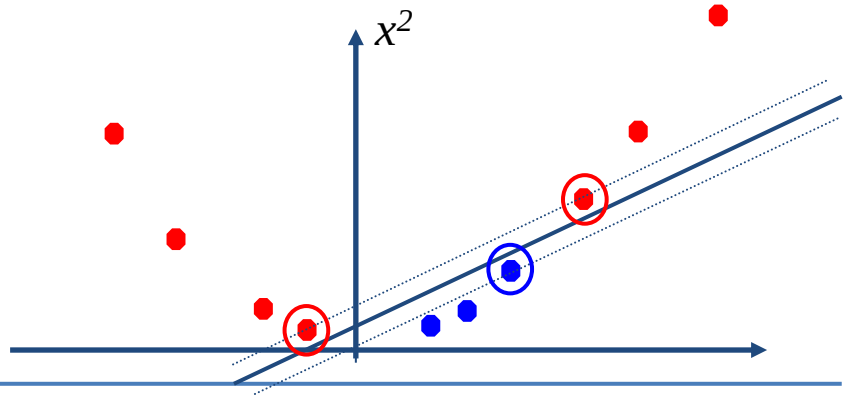
$$i \in \{1,\ldots,m\}$$



Non-linear SVMs (*on the inputs*)

- What about classes whose training instances are not linearly separable?
  - The original input space can always be *mapped* to some higher-dimensional feature space where the training set is separable.
    - A *kernel function* is some function that corresponds to an inner product in some expanded feature space.





SVM – discussion

- The support vector machine (SVM) algorithm is very fast and effective for text classification problems.
  - Flexibility in choosing a similarity function
    - By means of a kernel function
  - Sparseness of solution when dealing with large data sets
    - Only support vectors are used to specify the separating hyperplane
  - Ability to handle large feature spaces
    - Complexity does not depend on the dimensionality of the feature space

Naïve Bayes

- It is the simplest probabilistic classifier used to classify documents
  - Based on the application of the Bayes theorem

- Builds a *generative model* that approximates how data is produced
  - Uses *prior* probability of each category given no information about an item
  - Categorization produces a *posterior* probability distribution over the possible categories given a description of an item.

A. M. Kibriya, E. Frank, B. Pfahringer, G. Holmes. **Multinomial Naive Bayes for Text Categorization Revisited.** Australian Conference on Artificial Intelligence 2004: 488-499

Naïve Bayes

- Bayes theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- Why?
  - We know that:
  - Then

$$P(A,B) = P(A|B)P(B);\ P(A,B) = P(B|A)P(A)$$

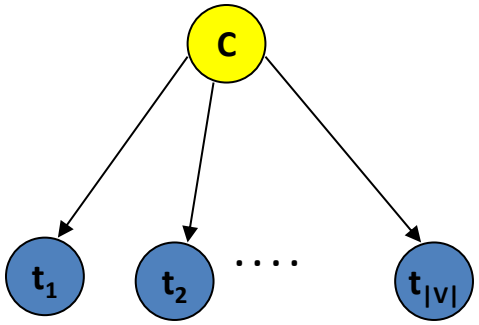
  - Then

$$P(A|B)P(B) = P(B|A)P(A)$$

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Naïve Bayes

- For a document *d* and a class c_j



$$P(C_j \mid \mathbf{d}) = \frac{P(\mathbf{d} \mid C_j)P(C_j)}{P(\mathbf{d})}$$

$$= \frac{P(t_1,...,t_{|V|} \mid C_j)P(C_j)}{P(t_1,...,t_{|V|})}$$

- Assuming terms are independent of each other given the class (naïve assumption)

$$= \frac{\prod_{i=1}^{|V|} P(t_i \mid C_j)P(C_j)}{P(\mathbf{d})}$$

- Assuming each document is *equally probable*

$$\propto \prod_{i=1}^{|V|} P(t_i \mid C_j)P(C_j)$$

Bayes' Rule for text classification

- For a document *d* and a class c_j

$$P(C_j|\mathbf{d}) \propto P(C_j)\prod_{i=1}^{|V|} P(t_i|C_j)$$

Bayes' Rule for text classification

- For a document *d* and a class c_j

$$P(C_j|\mathbf{d}) \propto P(C_j)\prod_{i=1}^{|V|} P(t_i|C_j)$$

- Estimation of probabilities

Smoothing to avoid zero-values

$$P(C_j) = \frac{N_j^c}{|D|} \qquad P(t_i|c_j) = \frac{1+N_{ij}}{|V|+\sum_{k=1}^{|V|} N_{kj}}$$

Prior probability of class c_j

Probability of occurrence of word t_i in class c_j

Naïve Bayes classifier

- Assignment of the class:

$$class = \arg\max_{C_j \in \mathrm{C}} P\left(C_j \left| \mathbf{d} \right.\right) = \arg\max_{C_j \in \mathrm{C}} P\left(C_j\right) \prod_{i=1}^{|V|} P\left(t_i \left| C_j \right.\right)$$

- Assignment using underflow prevention:
  - Multiplying lots of probabilities can result in floating-point underflow
  - Since log(*xy*) = log(*x*) + log(*y*), it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities

$$class = \operatorname{argmax}_{C_j \in \mathrm{C}} \left[\log P\left(C_j\right) + \sum_{i=1}^{|V|} \log P\left(t_i \mid C_j\right) \right]$$

Comments on NB classifier

- Very simple classifier which works very well on numerical and textual data
- Very easy to implement and computationally cheap when compared to other classification algorithms.
- One of its major limitations is that it performs very poorly when features are highly correlated.
- Concerning text classification, it fails to consider the frequency of word occurrences in the feature vector.

Evaluation of text classification

- What to evaluate?
- How to carry out this evaluation?
  - Which elements (information) are required?
- How to know which is the best classifer for a given task?
  - Which things are important to perform a fair comparison?

Evaluation – general ideas

- Performance of classifiers is evaluated experimentally

- Requires a document set labeled with categories.
  - Divided into two parts: *training* and *test* sets
  - Usually, the test set is the smaller of the two

- A method to smooth out the variations in the corpus is the *n-fold cross-validation.*
  - The whole document collection is divided into *n* equal parts, and then the training-and-testing process is run *n* times, each time using a different part of the collection as the test set. Then the results for *n* folds are averaged.

Performance metrics

- Considering a binary problem

$$\text{accuracy} = \frac{a+d}{a+b+c+d}$$

	Label YES	Label NO
Classifier YES	***a***	***b***
Classifier NO	***c***	***d***

$$\text{recall (R)} = \frac{a}{a+c} \qquad \text{precision (P)} = \frac{a}{a+b} \longrightarrow F = \frac{2PR}{P+R}$$

- Recall for a category is defined as the percentage of correctly classified documents among all documents belonging to that category, and precision is the percentage of correctly classified documents among all documents that were assigned to the category by the classifier.

What happen if there are more than two classes?

Micro and macro averages

- *Macroaveraging*: Compute performance for each category, then average.
  - Gives equal weights to all categories

- *Microaveraging*: Compute totals of a, b, c and d for all categories, and then compute performance measures.
  - Gives equal weights to all documents

Is it important the selection of the averaging strategy?

What happen if we are very bad classifying the minority class?

References

- G. Forman. **An Extensive Empirical Study of Feature Selection Metrics for Text Classification.** JMLR, 3:1289—1305, 2003
- H. Liu, H. Motoda. **Computational Methods of Feature Selection.** Chapman & Hall, CRC, 2008.
- Y. Yang, J. O. Pedersen. **A Comparative Study on Feature Selection in Text Categorization.** Proc. of the 14th International Conference on Machine Learning, pp. 412—420, 1997.
- D. Mladenic, M. Grobelnik. **Feature Selection for Unbalanced Class Distribution and Naïve Bayes.** Proc. of the 16th Conference on Machine Learning, pp. 258—267, 1999.
- I. Guyon, et al. **Feature Extraction Foundations and Applications,** Springer, 2006.
- Guyon, A. Elisseeff. **An Introduction to Variable and Feature Selection.** Journal of Machine Learning Research, Vol. 3:1157—1182, 2003.
- M. Lan, C. Tan, H. Low, S. Sung. **A comprehensive comparative study on term weighting schemes for text categorization with support vector machines.** Proc. of WWW, pp. 1032—1033, 2005.