

# AI6122 Text Data Management & Analysis

Topic: Text Classification



# Example Tasks for Text Classification

- Sentiment analysis
  - Does a movie review have positive/negative comments?
  - Predict the rating of a review, in 1 to 5?
- Personal email sorting
  - Course, research, service, private, etc.
  - Spam email or non-spam email
- News articles in different categories
  - Economics, sports, entertainment...



# Text classification

- Introduction to classification models
  - Also widely known as “text categorization”.
- Issues in applying text classification
- Text classification evaluation
- Feature extraction and feature selection
  - From text documents to feature vector
- There are many other classifiers and related topics
  - Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. ACM Comput. Surv. 34, 1 (March 2002), 1-47.



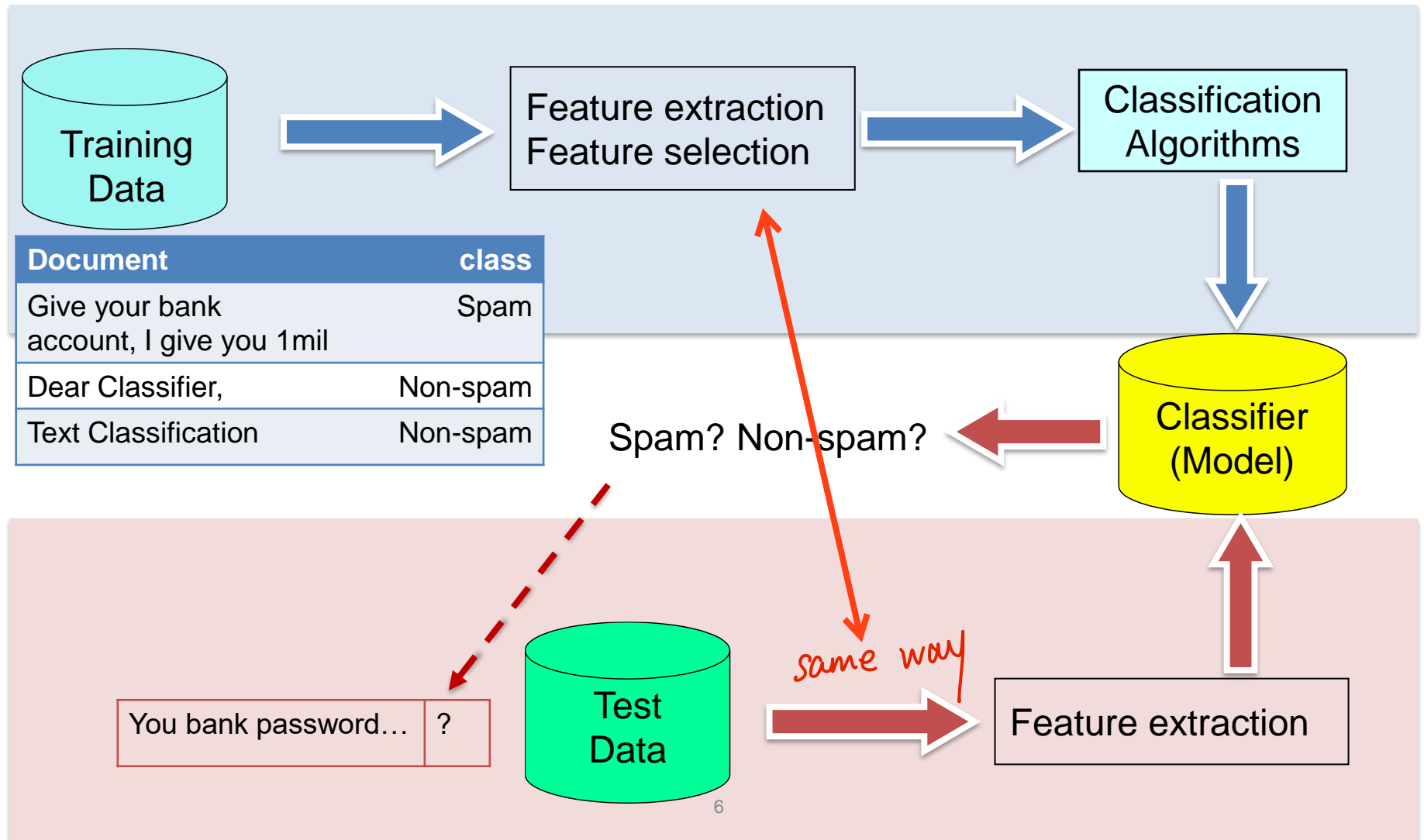
# Categorization/Classification

- Given:
  - A description of an instance  $d \in X$ , where  $X$  is the instance language or instance space.
    - Issue: how to represent a text document  $d$ .
    - Usually some type of high-dimensional space
  - A fixed set of classes:  $C = \{c_1, c_2, \dots, c_j\}$
- Determine:
  - The category of  $d : \gamma(d) \in C$ , where  $\gamma(d)$  is a classification function whose domain is  $X$  and whose range is  $C$ .
  - We want to know how to build classification functions (“classifiers”).

# Supervised Classification

- Given:
  - A description of an instance  $d \in X$ , where  $X$  is the *instance language* or *instance space*.
  - A fixed set of classes:  $C = \{c_1, c_2, \dots, c_j\}$
  - **A training set  $D$**  of labeled documents ( $D \subset X$ ):  $\langle d, c \rangle \in D \times C$
- Determine:
  - A learning method or algorithm which learns a classifier  $\gamma: X \rightarrow C$
  - For a test document  $d$ , we assign it the class  $\gamma(d) \in C$

# Classification Framework: Train vs Test



# Classification Methods

- **Manual** classification
  - Used by the original Yahoo! Directory, PubMed
  - Very accurate when job is done by experts; Consistent when the problem size and team is small
  - Difficult and expensive to scale
- **Automatic** document classification by **Hand-coded rule-based** systems
  - Assign category (e.g., Yes/No category) if document matches a predefined query. Google Alerts? Following News of predefined keywords?
  - Accuracy is often very high if a rule has been carefully refined over time by a subject expert
  - Building and **maintaining** these rules is expensive

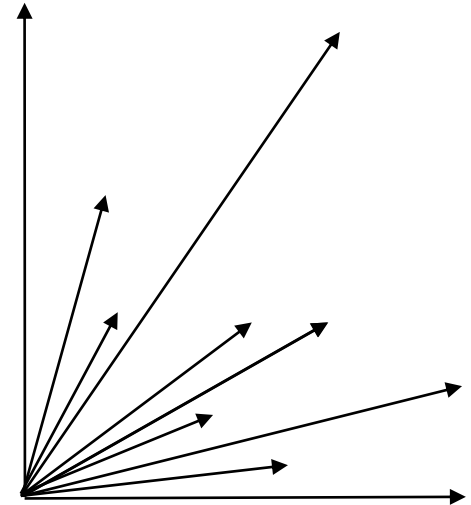
# Classification Methods

- **Supervised learning** of a document-label assignment function
  - k-Nearest Neighbors (simple, powerful)
  - Naive Bayes (simple, common method)
  - Support-vector machines (more powerful)
  - Many other methods, e.g., logistic regression, random forests, neural networks
  - No free lunch:
    - Requires hand-classified training data to learn classifiers
    - Requires feature engineering to best represent documents
    - Explainable results?
- Many commercial systems use a **mixture of methods**

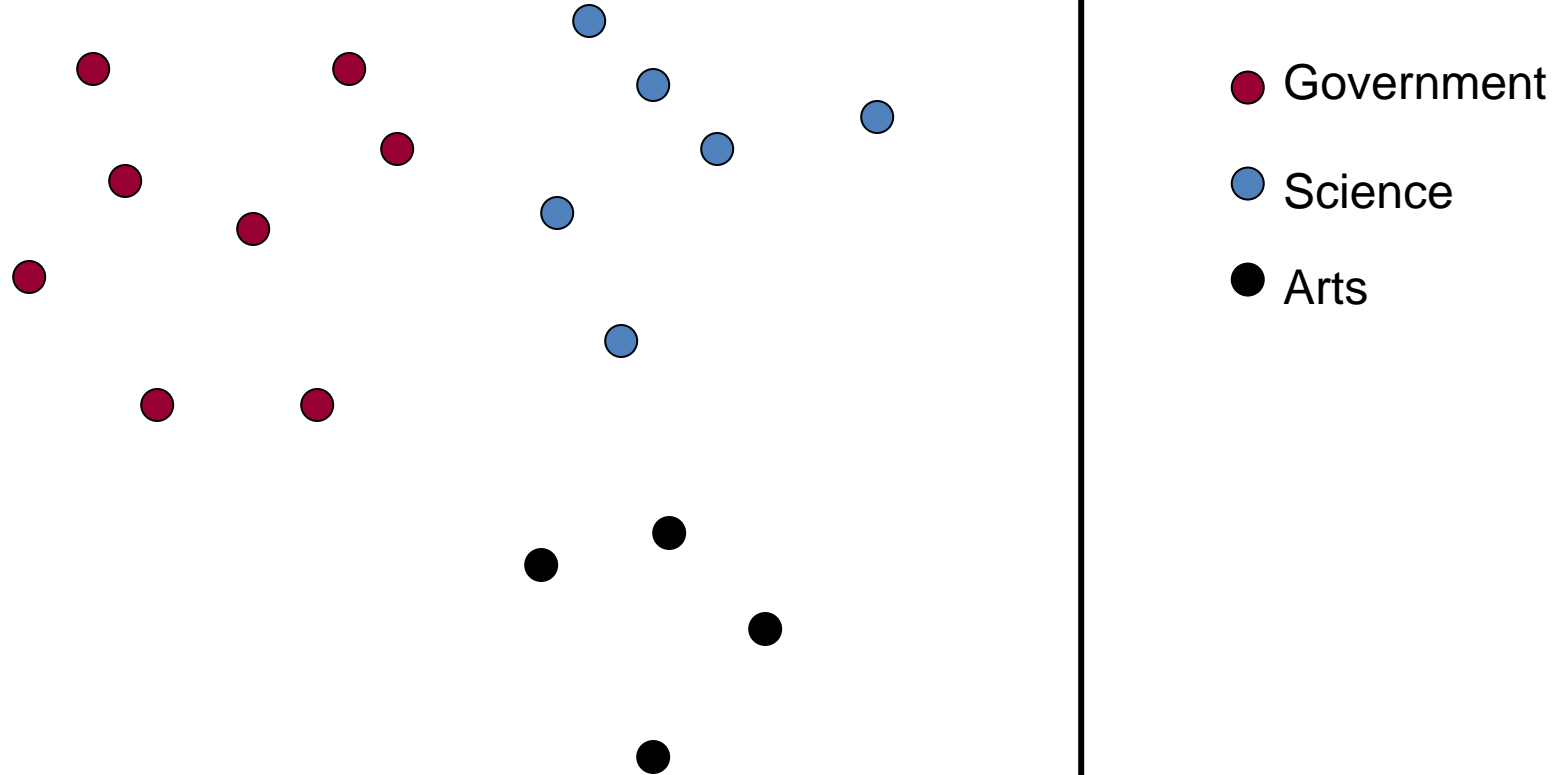


# Rocchio Classifier

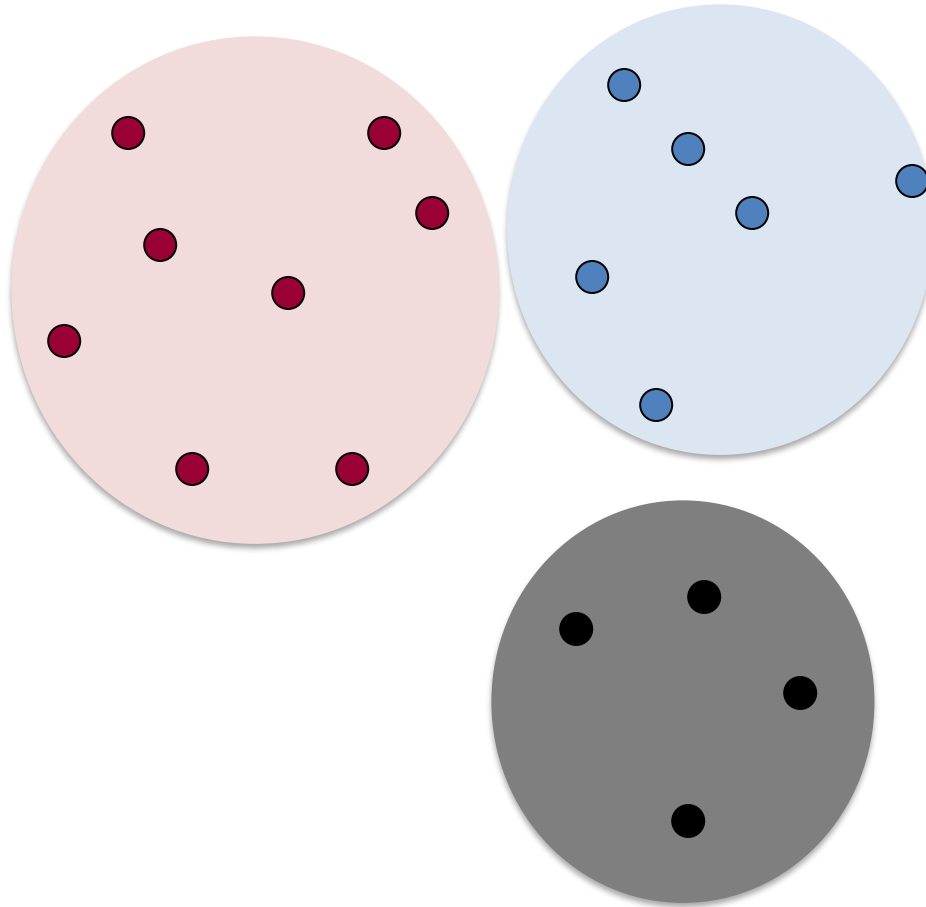
- Each document is a vector, i.e., Vector Space Representation
  - One dimension for each term/word, e.g., TFIDF vector
  - Normally normalize vectors to unit length.
- High-dimensional vector space:
  - Terms are axes
  - 10,000+ dimensions, or even 100,000+
  - Docs are vectors in this space
- Assumptions
  - Documents in the same class form a contiguous region of space
  - Documents from different classes don't overlap (much)



# Documents in a Vector Space

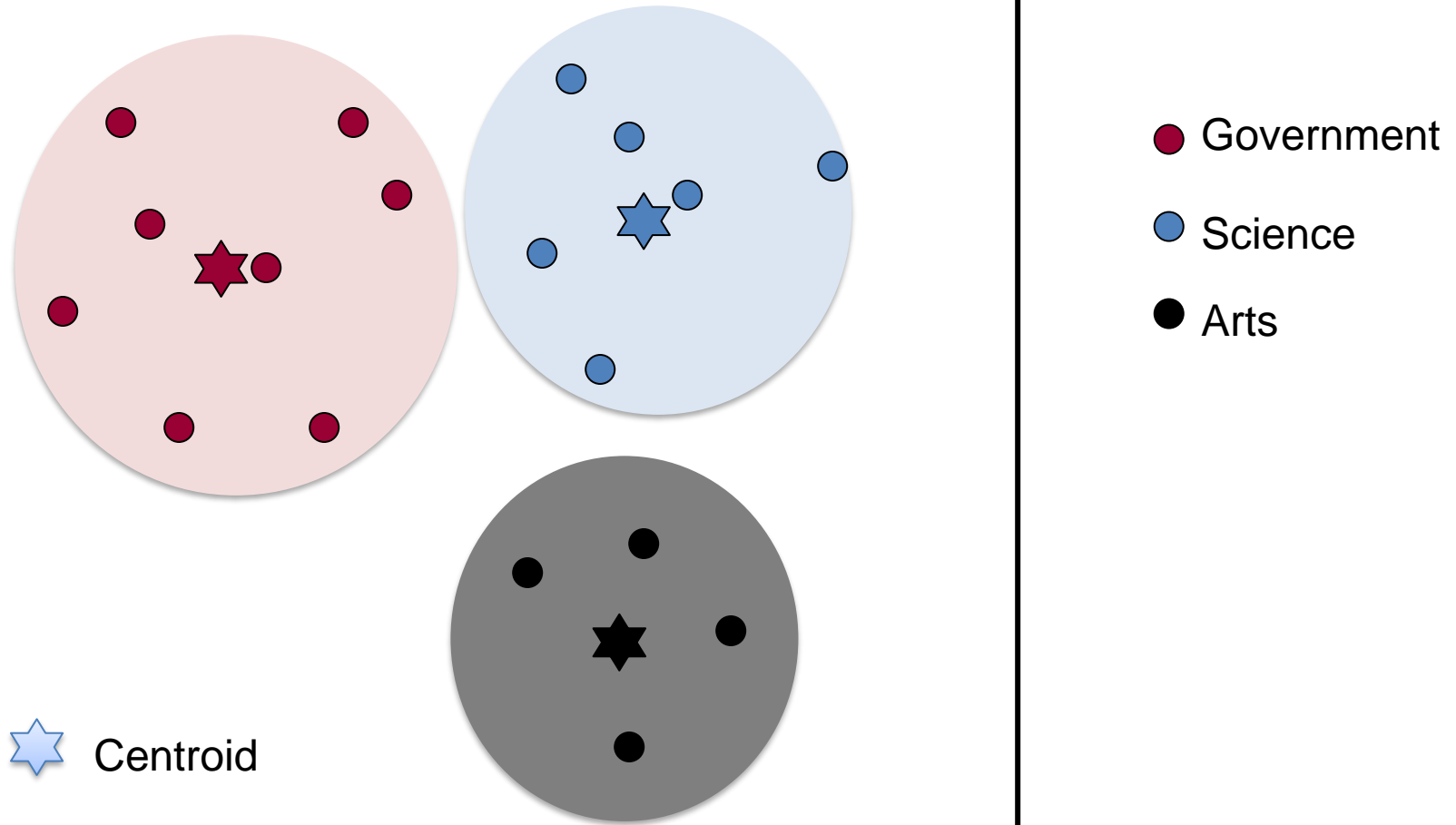


# Documents in a Vector Space

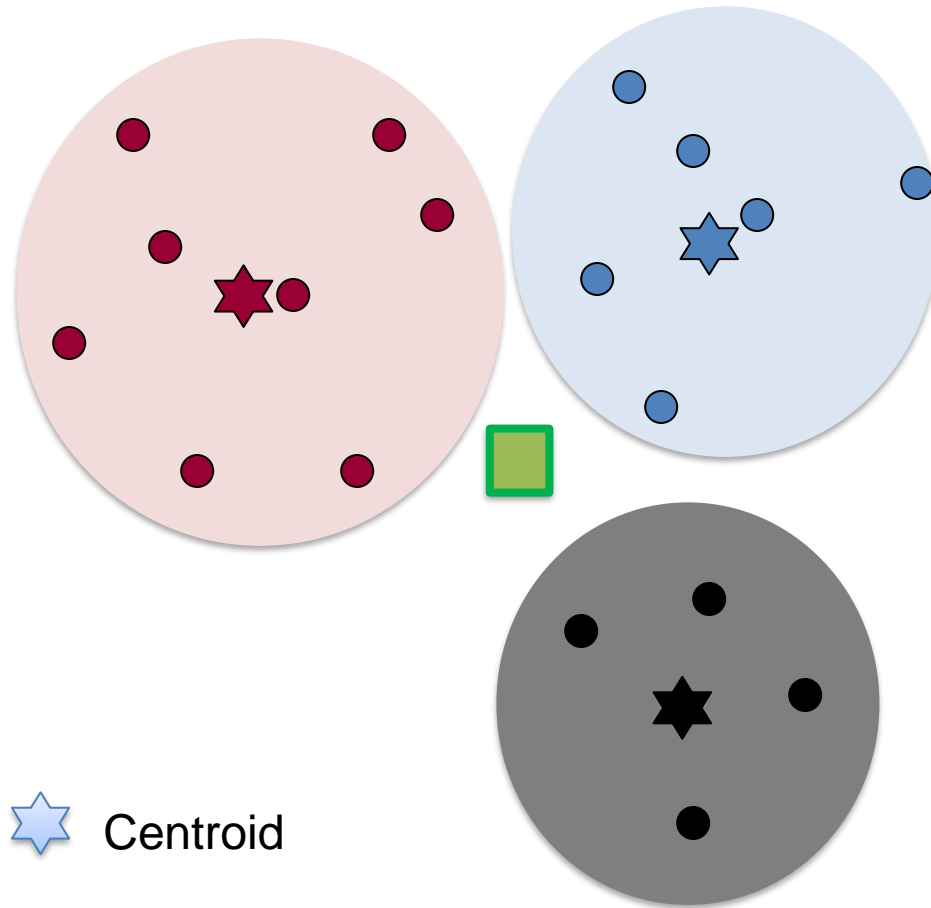


- Government
- Science
- Arts

# Documents in a Vector Space



# Documents in a Vector Space



● Government

● Science

● Arts

■ To be classified

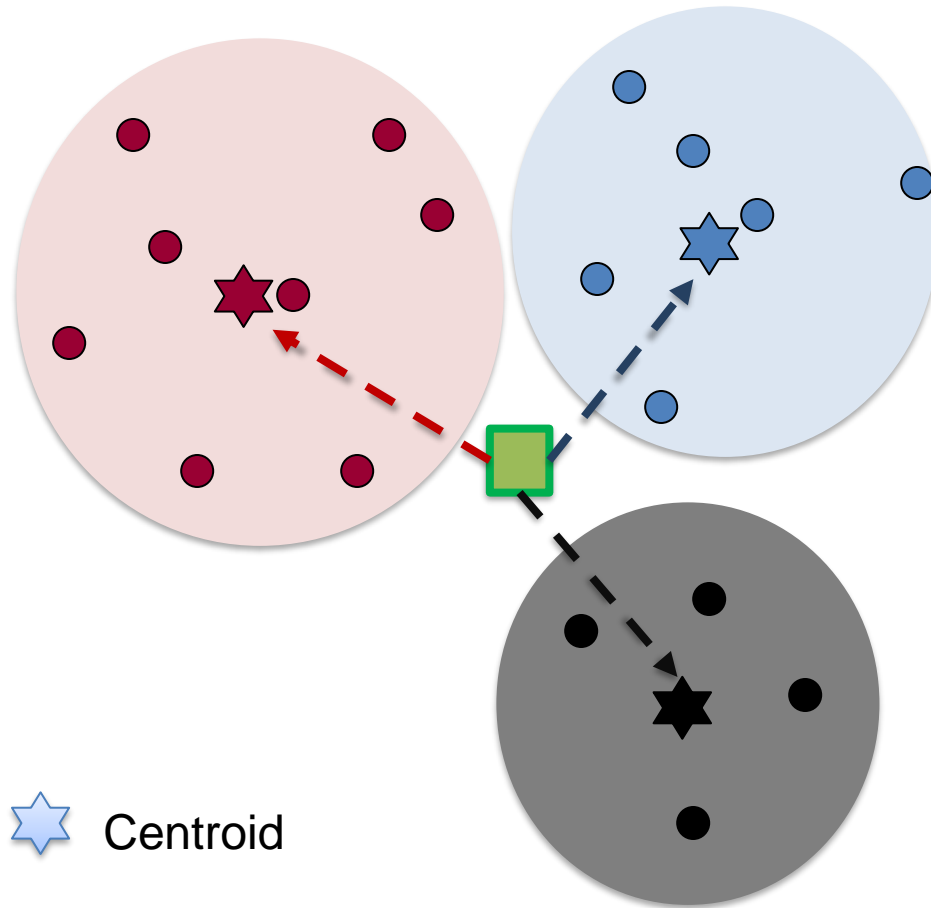
# Rocchio Classifier

- Use standard tf-idf weighted vectors to represent text documents
  - Features are words with tf-idf weighted
- For training documents in each category, compute a **prototype vector** by averaging the vectors of the training documents in the category.
  - Prototype = centroid of members of class

$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d \in D_c} \vec{v}(d)$$

- $D_c$  is the set of documents in class  $c$ .
  - Centroid in general is not a unit vector even when the inputs are unit vectors.
- Assign a test document to the category with the **closest prototype** vector based on cosine similarity.

# Documents in a Vector Space



● Government

● Science

● Arts

■ To be classified

# Rocchio classification

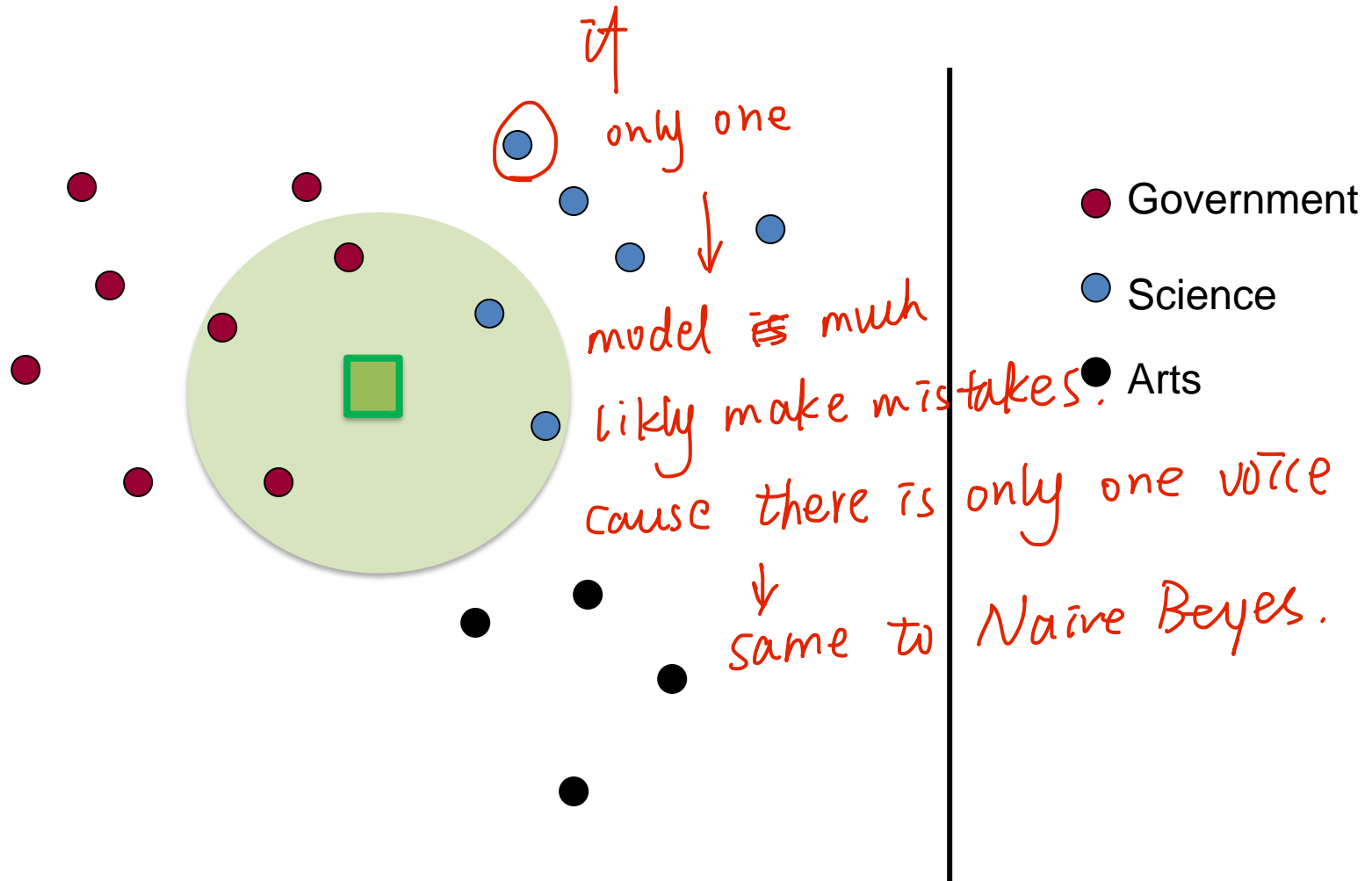
- Rocchio forms a simple representation for each class: centroid
- Classification is based on similarity to (or distance from) the centroid
- Very easy to build, but in general is worse than Naïve Bayes
- Sometimes, it is called centroid based classifier



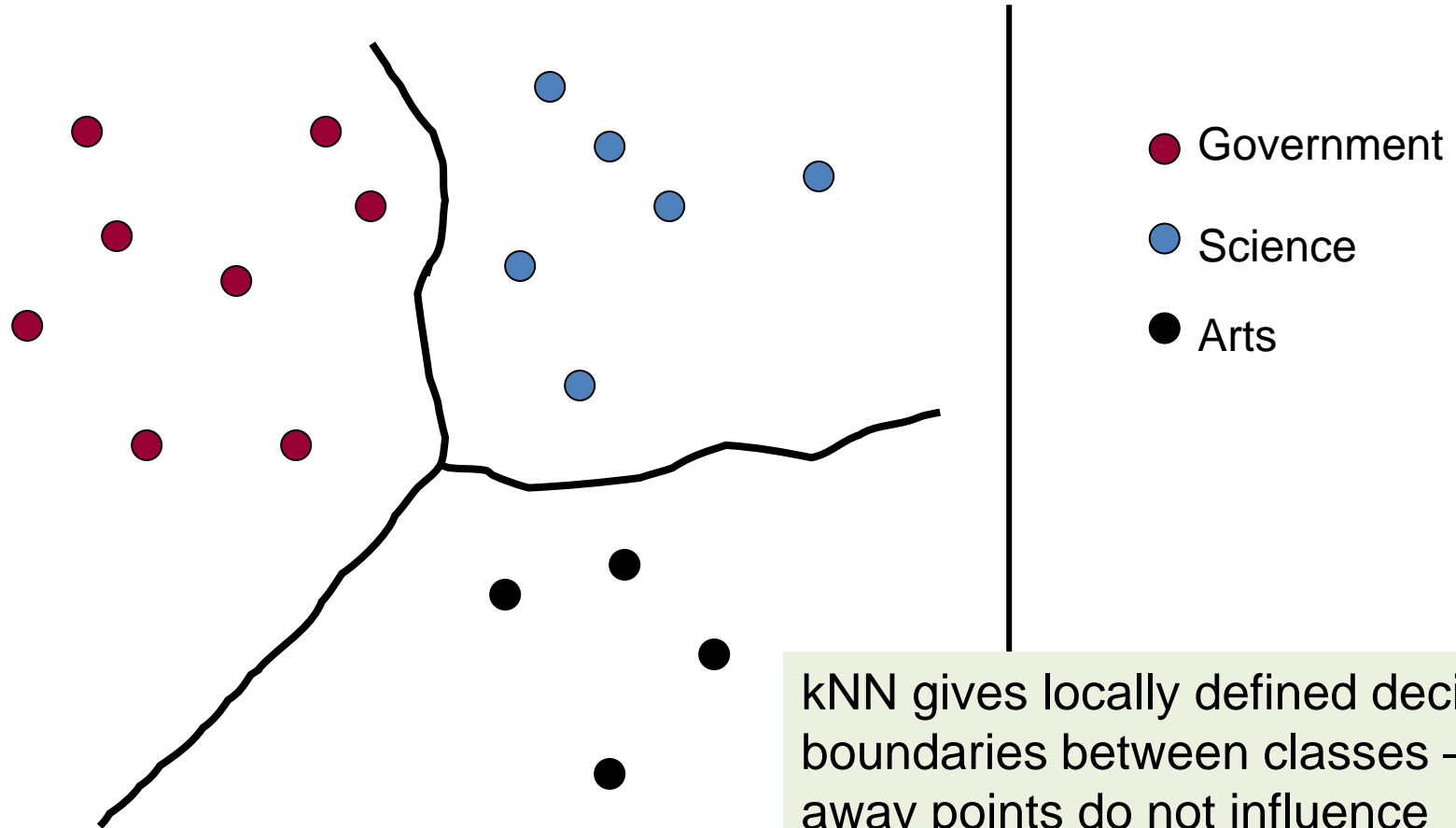
# k Nearest Neighbor Classification

- **kNN = k Nearest Neighbor**
- To classify a document  $d$ :
  - Define neighborhood  $N$  as  $k$  nearest neighbors of  $d$
  - Count number of documents in  $N$  that belong to each category
  - Choose the category that has maximum number of documents in  $N$
- Learning is just storing the representations of the training examples.
  - Also known as: Case-based learning, Memory-based learning, Lazy learning
- Nearest neighbor method depends on a similarity (or distance) metric
  - Compute similarity between a test document and all training documents.
  - Cosine similarity is widely adopted
  - In general is slow.

# kNN Illustration ( $k = 5$ in this example)



# kNN decision boundaries



kNN gives locally defined decision boundaries between classes – far away points do not influence classification decision (unlike in Naïve Bayes, Rocchio, etc.)

# Nearest Neighbor with Inverted Index

- Naively finding nearest neighbors requires a linear search through  $|D|$  documents in collection
  - 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.
- Discussion on kNN Classifier
  - No feature selection necessary
  - Scales well with large number of classes
  - Scores can be hard to convert to probabilities
  - (Almost) no training necessary, but is expensive at test time
  - In most cases it is more accurate than Naïve Bayes or Rocchio

# Next: Naïve Bayes

- Bayes' Rule (for events  $a$  and  $b$ ):

$$- P(a, b) = P(a \cap b) = P(a|b)P(b) = P(b|a)P(a)$$

$$- P(b) = P(b|a)P(a) + P(b|\bar{a})P(\bar{a})$$

$$- P(a|b) = \frac{P(b|a)P(a)}{P(b)} = \frac{P(b|a)P(a)}{\sum_{x=a, \bar{a}} P(b|x)P(x)}$$

- Odds:

$$- O(a) = \frac{P(a)}{P(\bar{a})} = \frac{P(a)}{1-P(a)}$$

# Bayes' Rule for text classification

- For a document  $d$  and a class  $c$

$$P(c, d) = P(c|d)P(d) = P(d|c)P(c)$$

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

# Naïve Bayes Classifiers

- Classify a new instance  $d$  based on a tuple of attribute values  $d = x_1, x_2, \dots, x_n$  into one of the classes  $c_j \in \mathcal{C}$

$$c_{MAP} = \operatorname{argmax}_{c_j \in \mathcal{C}} P(c_j | x_1, x_2, \dots, x_n) \leftarrow \text{The probability of a document } d \text{ being in class } c.$$

$$= \operatorname{argmax}_{c_j \in \mathcal{C}} \frac{P(x_1, x_2, \dots, x_n | c_j) P(c_j)}{P(x_1, x_2, \dots, x_n)} \leftarrow \text{Bayes' Rule}$$

We can ignore the denominator

$$= \operatorname{argmax}_{c_j \in \mathcal{C}} P(x_1, x_2, \dots, x_n | c_j) P(c_j)$$

MAP is “*maximum a posteriori*” = most likely<sup>23</sup> class



# Naïve Bayes Classifier: Prior probability

- $P(c_j)$  can be estimated from the frequency of classes in the training examples.
  - simply use the frequencies in the data
  - maximum likelihood estimates:

$$\operatorname{argmax}_{c_j \in C} P(x_1, x_2, \dots, x_n | c_j) P(c_j)$$

$$\hat{P}(c_j) = \frac{N(C = c_j)}{N}$$

number of documents belong to class  $c_j$  in training data

number of documents in training data



# Naïve Bayes Classifier: Prior probability Example

- $P(c_j)$ : estimated from the frequency of classes in the training examples.

	docID	words in document	in c = China?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Japan Chinese Chinese Chinese Tokyo	?

$$\hat{P}(c_j) = \frac{N(C = c_j)}{N}$$

$$P(c) = \frac{3}{4} = 0.75$$

$$P(\bar{c}) = \frac{1}{4} = 0.25$$

# Naïve Bayes Classifier: Naïve Bayes Assumption

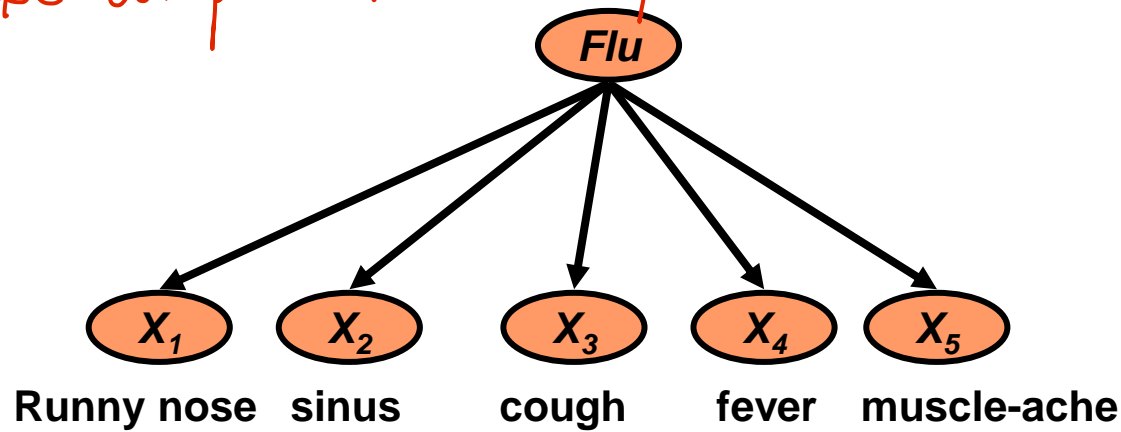
$$\operatorname{argmax}_{c_j \in C} P(x_1, x_2, \dots, x_n | c_j) P(c_j)$$

- $P(x_1, x_2, \dots, x_n | c_j)$ 
  - Could only be estimated from a very, very large number of training examples.
- Naïve Bayes Conditional Independence Assumption:
  - Assume that the probability of observing the conjunction of attributes is equal to the product of the individual probabilities  $P(x_i | c_j)$ .
  - With this assumption:  $P(x_1, x_2 | c_j) = P(x_1 | c_j) \times P(x_2 | c_j)$

# The Naïve Bayes Classifier

- Conditional Independence Assumption *it just works well*
  - Features (or terms/words) are independent of each other given the class

*& make computation easy.*



$$P(x_1, \dots, x_5|c) = P(x_1|c) \times P(x_2|c) \times P(x_3|c) \times P(x_4|c) \times P(x_5|c)$$

# Learning the Model

- First attempt: **maximum likelihood estimates**, simply use the frequencies in the data

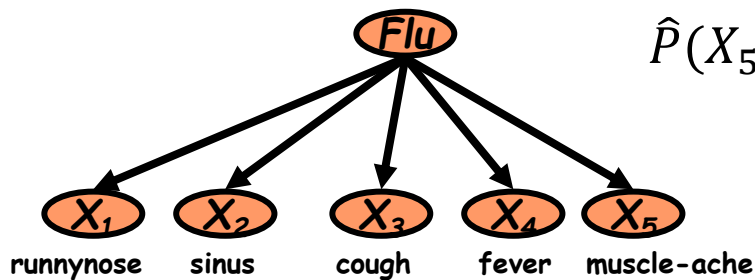
$$\hat{P}(x_i|c_j) = \frac{N(X_i = x_i, C = c_j)}{N(C = c_j)}$$

number of occurrences of **word**  $x_i$  in documents belong to **class**  $c_j$  in training data

number of all term occurrences in documents belong to **class**  $c_j$  in training data

# Problem with Maximum Likelihood

- What if we have seen no training documents with the word muscle-ache in the topic “Flu”?
  - But the word appears in documents in other topics
- Zero probability



$$\hat{P}(X_5 = t | C = nf) = \frac{N(X_5 = t, C = nf)}{N(C = nf)} = 0$$

$$\operatorname{argmax}_{c_j \in C} P(x_1, x_2, \dots, x_n | c_j) P(c_j)$$

# Smoothing

- Before smoothing

$$\hat{P}(x_i|c_j) = \frac{N(X_i = x_i, C = c_j)}{N(C = c_j)}$$

- **Smoothing**: assume each word appears at least once in the given category

$$\hat{P}(x_i|c_j) = \frac{N(X_i = x_i, C = c_j) + 1}{N(C = c_j) + k}$$

$k$  is the number of values in  $X_i$  (or the size of vocabulary)



# Naive Bayes Classifiers: Summary

$$d = \langle x_1, x_2, \dots, x_n \rangle$$

$$c_{MAP} = \operatorname{argmax}_{c_j \in C} P(c_j | x_1, x_2, \dots, x_n)$$

$$= \operatorname{argmax}_{c_j \in C} P(x_1, x_2, \dots, x_n | c_j) P(c_j)$$

$$= \operatorname{argmax}_{c_j \in C} P(x_1 | c_j) P(x_2 | c_j) \dots P(x_n | c_j) P(c_j)$$

$$\hat{P}(x_i | c_j) = \frac{N(X_i = x_i, C = c_j) + 1}{N(C = c_j) + k}$$

$$\hat{P}(c_j) = \frac{N(C = c_j)}{N}$$

# Naive Bayes: Learning

- From training corpus, extract Vocabulary
- Calculate required  $P(c_j)$  and  $P(x_k|c_j)$  terms  $\frac{N(x_k)}{\text{Total}(C)}$ 
  - For each  $c_j$  in  $C$  do
    - $\text{docs}_j \leftarrow$  subset of documents which belong to  $c_j$

$$P(c_j) \leftarrow \frac{|\text{docs}_j|}{|\text{total \# documents}|}$$

- $\text{Text}_j \leftarrow$  create a virtual single document containing all  $\text{docs}_j$ 
  - for each word  $x_k$  in Vocabulary
  - $n_k \leftarrow$  number of occurrences of  $x_k$  in  $\text{Text}_j$

$$P(x_k|c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha|\text{Vocabulary}|}$$



# Naive Bayes: Classifying

- Appearance of the same word at different positions
  - all word appearances in current document  $d$  are counted
  - Same as considering all words at all positions
- Return  $c_{NB}$  as the class of  $d$ , where

$$c_{NB} = \operatorname{argmax}_{c_j \in \mathcal{C}} P(c_j) \prod_{i \in \text{positions}} P(x_i | c_j)$$

	docID	words in document	in c = China?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Japan Chinese Chinese Chinese Tokyo	?

➤ Training:

- Vocabulary  $V = \{\text{Chinese, Beijing, Shanghai, Macao, Tokyo, Japan}\}$  and  $|V| = 6$ .

- $P(c) = \frac{3}{4}$

$$P(\bar{c}) = \frac{1}{4}$$

- $P(\text{Chinese}|c) = \frac{5+1}{8+6} = \frac{3}{7}$

$$P(\text{Chinese}|\bar{c}) = \frac{1+1}{3+6} = \frac{2}{9}$$

- $P(\text{Tokyo}|c) = \frac{0+1}{8+6} = \frac{1}{14}$

$$P(\text{Tokyo}|\bar{c}) = \frac{1+1}{3+6} = \frac{2}{9}$$

- $P(\text{Japan}|c) = \frac{0+1}{8+6} = \frac{1}{14}$

$$P(\text{Japan}|\bar{c}) = \frac{1+1}{3+6} = \frac{2}{9}$$

➤ Testing:

- $P(c|d) \propto P(c) \times P(\text{Japan}|c) \times P(\text{Chinese}|c)^3 \times P(\text{Tokyo}|c) = \frac{3}{4} \times \frac{1}{14} \times \left(\frac{3}{7}\right)^3 \times \frac{1}{14} \approx 0.0003$

- $P(\bar{c}|d) \propto P(\bar{c}) \times P(\text{Japan}|\bar{c}) \times P(\text{Chinese}|\bar{c})^3 \times P(\text{Tokyo}|\bar{c}) = \frac{1}{4} \times \frac{2}{9} \times \left(\frac{2}{9}\right)^3 \times \frac{2}{9} \approx 0.0001$

# Bayesian Methods

- Learning and classification methods based on probability theory.
- Builds a **generative model** that approximates how data is produced
  - $P(X|Y) \sim P(Y|X)P(X)$
  - A discriminative model would directly estimate  $P(X|Y)$
- 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.

# Underflow Prevention: using logs

- Multiplying lots of probabilities can result in floating-point underflow.

$$c_{NB} = \operatorname{argmax}_{c_j \in C} P(c_j) \prod_{i \in \text{positions}} P(x_i | c_j)$$

- Perform all computations by summing logs of probabilities rather than multiplying probabilities.

$$c_{NB} = \operatorname{argmax}_{c_j \in C} [\log P(c_j) + \sum_{i \in \text{positions}} \log P(x_i | c_j)]$$

- Class with highest final un-normalized log probability score is still the most probable. The model is now just max of sum of weights

# Naive Bayes Classifier

$$c_{NB} = \operatorname{argmax}_{c_j \in \mathcal{C}} [\log P(c_j) + \sum_{i \in \text{positions}} \log P(x_i | c_j)]$$

- Simple interpretation:
  - Each conditional parameter  $\log P(x_i | c_j)$  is a weight that indicates how good an indicator  $x_i$  is for class  $c_j$ .
  - The prior  $\log P(c_j)$  is a weight that indicates the relative frequency of  $c_j$ .
  - The sum is then a measure of how much evidence there is for the document being in the class.
- We select the class with the most evidence for it

# Two Naive Bayes Models

- **Model 1: Multivariate Bernoulli**

- One feature  $X_w$  for each word in dictionary
- $X_w = 1$  in document  $d$  if word  $w$  appears in  $d$
- Naive Bayes assumption: Given the document's topic, appearance of one word in the document tells us nothing about chances that another word appears (or the appearance of the words are independent)

$$\hat{P}(x_i|c_j) = \frac{N(X_i = x_i, C = c_j) + 1}{N(C = c_j) + k}$$

$$\hat{P}(x_w|c_j) = \frac{N(X_w = x_w, C = c_j) + 1}{N(C = c_j) + k}$$

number of documents belonging to class  $c_j$  in training data, which contain word  $x_w$

number of all documents belonging to class  $c_j$  in training data

number of all classes

# Two Naive Bayes Models

- **Model 2: Multinomial = Class conditional unigram**

- One feature  $X_i$  for each word position in document. Value of  $X_i$  is the word in position  $i$  in document  $d$
- Naïve Bayes assumption: Given the document's topic, word in one position in the document tells us nothing about words in other positions
- Second assumption: Word appearance does not depend on position

$$\hat{P}(x_i|c_j) = \frac{N(X_i = x_i, C = c_j) + 1}{N(C = c_j) + k}$$

number of occurrences of word  $x_i$  in documents belong to class  $c_j$  in training data

number of all term occurrences in documents belong to class  $c_j$  in training data

number of all terms in vocabulary

# Parameter estimation: $\hat{P}(X_w = t|C_j)$

- Multivariate Bernoulli model:
  - Fraction of **documents** of topic  $c_j$  in which word  $w$  appears
- Multinomial model:
  - Fraction of **times of word**  $w$  appears among all words in documents of topic  $c_j$   
*much better accuracy*
- Difference in how **non-occurring** terms are used
  - used in Bernoulli model, but not in multinomial model
  - (see example in the next slides)



	docID	words in document	in c = China?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Japan Chinese Chinese Chinese Tokyo	?

➤ Bernoulli Model Training:

- Vocabulary  $V = \{\text{Chinese, Beijing, Shanghai, Macao, Tokyo, Japan}\}$  and  $|V| = 6$ .

- $P(c) = \frac{3}{4}$

$$P(\bar{c}) = \frac{1}{4}$$

- $P(\text{Chinese}|c) = \frac{3+1}{3+2} = \frac{4}{5}$

$$P(\text{Chinese}|\bar{c}) = \frac{1+1}{1+2} = \frac{2}{3}$$

- $P(\text{Beijing}|c) = P(\text{Shanghai}|c) = P(\text{Macao}|c) = \frac{1+1}{3+2} = \frac{2}{5}$

- $P(\text{Beijing}|\bar{c}) = P(\text{Shanghai}|\bar{c}) = P(\text{Macao}|\bar{c}) = \frac{0+1}{1+2} = \frac{1}{3}$

- $P(\text{Tokyo}|c) = P(\text{Japan}|c) = \frac{0+1}{3+2} = \frac{1}{5}$

$$P(\text{Tokyo}|\bar{c}) = P(\text{Japan}|\bar{c}) = \frac{1+1}{1+2} = \frac{2}{3}$$

➤ Testing:

- $$P(c|d) \propto P(c) \times P(\text{Japan}|c) \times P(\text{Chinese}|c) \times P(\text{Tokyo}|c) \times (1 - P(\text{Beijing}|c)) \times (1 - P(\text{Shanghai}|c)) \times (1 - P(\text{Macao}|c))$$
  

$$= \frac{3}{4} \times \frac{1}{5} \times \frac{4}{5} \times \frac{1}{5} \times (1 - \frac{2}{5}) \times (1 - \frac{2}{5}) \times (1 - \frac{2}{5}) \approx 0.005$$

- $$P(\bar{c}|d) \propto \frac{1}{4} \times \frac{2}{3} \times \frac{2}{3} \times \frac{2}{3} \times (1 - \frac{1}{3}) \times (1 - \frac{1}{3}) \times (1 - \frac{1}{3}) \approx 0.022$$

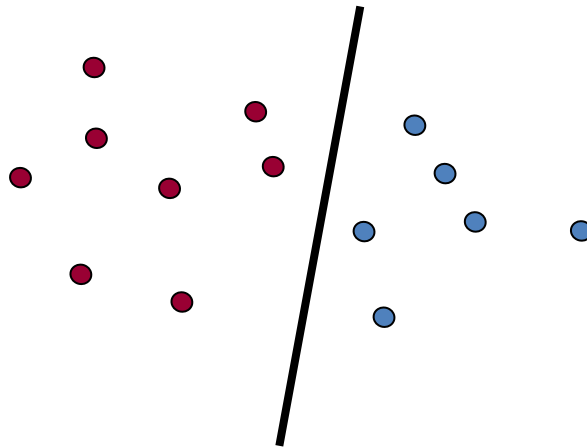
# Classification

- Multinomial vs Multivariate Bernoulli?
  - Multinomial model is almost always more effective in text applications!
- References:
  - Andrew McCallum and Kamal Nigam. A Comparison of Event Models for Naive Bayes Text Classification. In AAI/ICML-98 Workshop on Learning for Text Categorization, pp. 41-48.

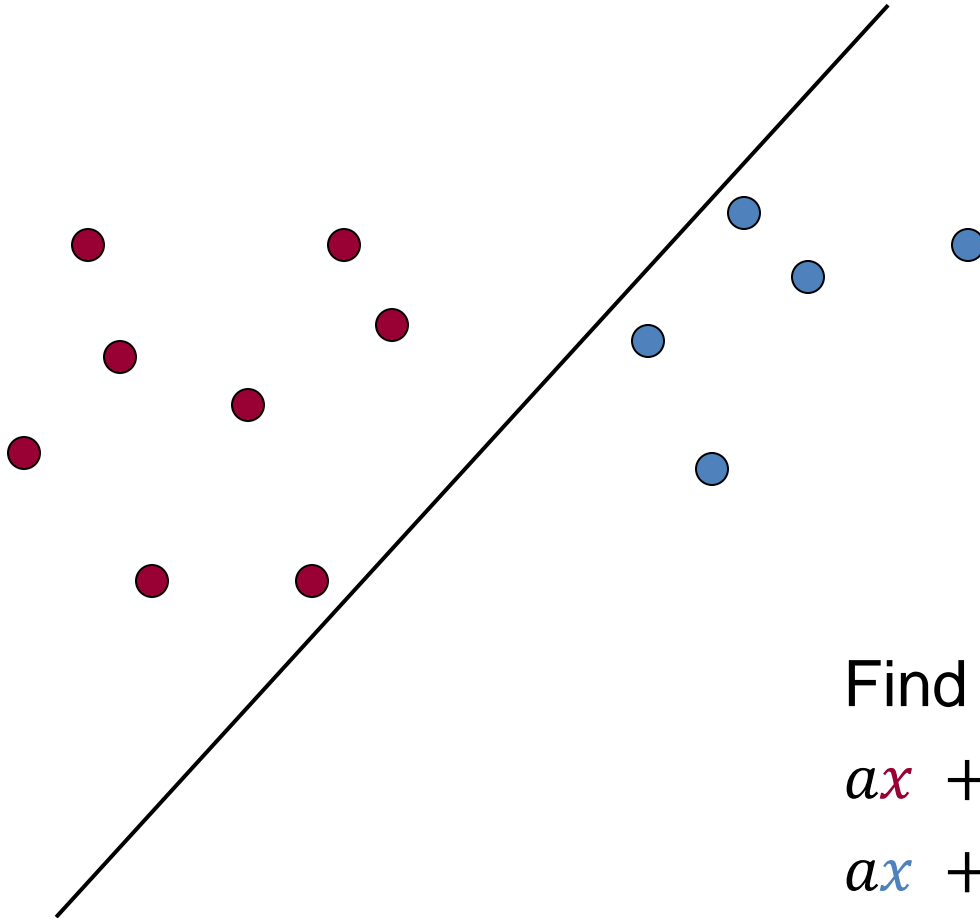


# Support Vector Machines: binary classification

- Binary classification:
  - Deciding between two classes, e.g., spam vs non-spam
- Linear separability:
  - In two dimensions, separate two classes by a line. Separator can be expressed as  $ax + by = c$
  - in higher dimensions, need hyperplanes. A hyperplane can be considered as a generalization of a line to higher dimensions

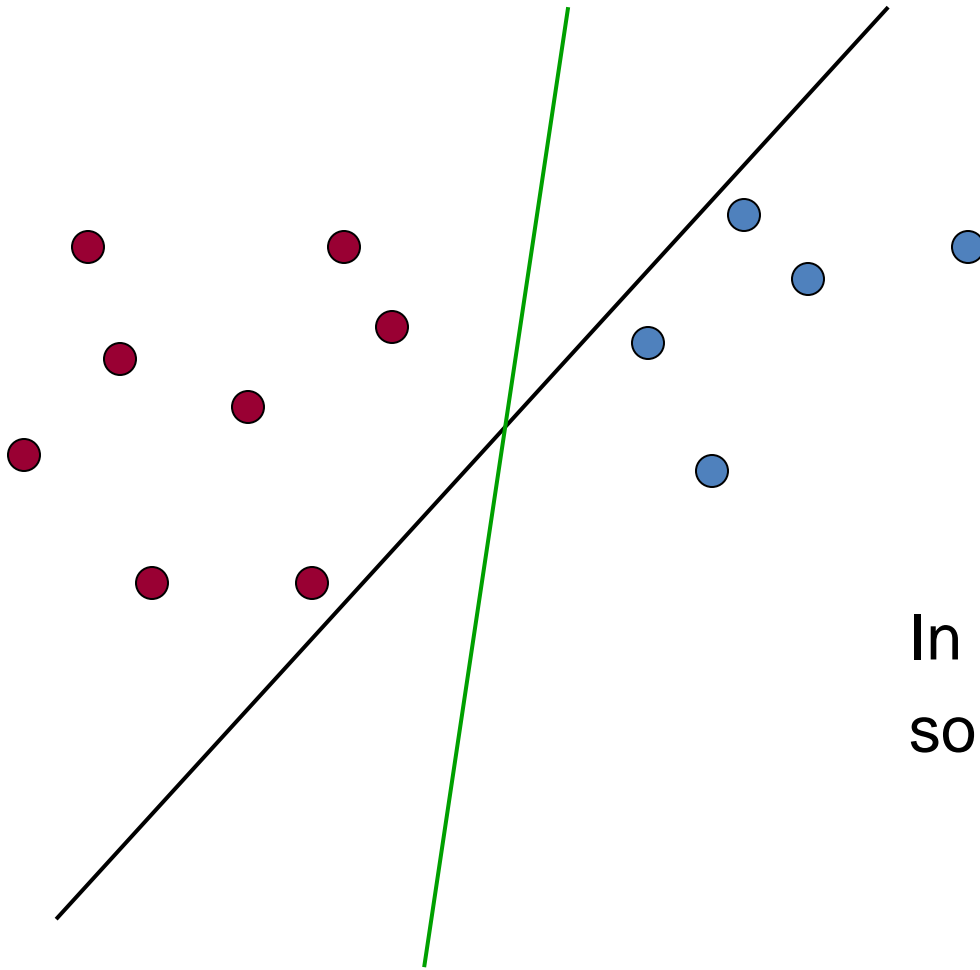


# Linear programming / Perceptron



Find  $a, b, c$ , such that  
 $ax + by > c$  for red points  
 $ax + by < c$  for blue points.

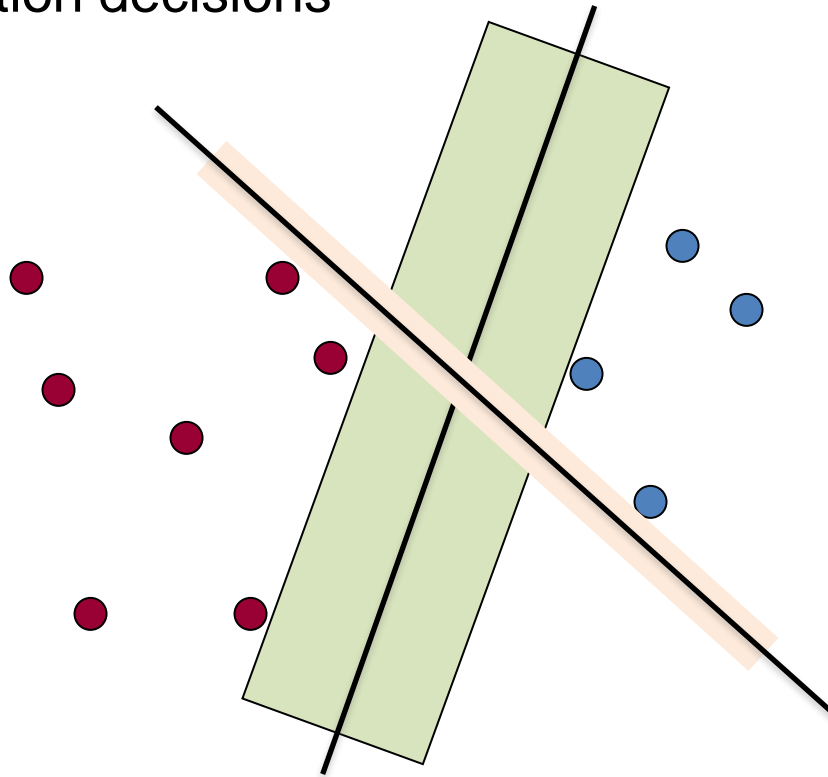
# Which Hyperplane?



In general, lots of possible solutions for  $a, b, c$ .

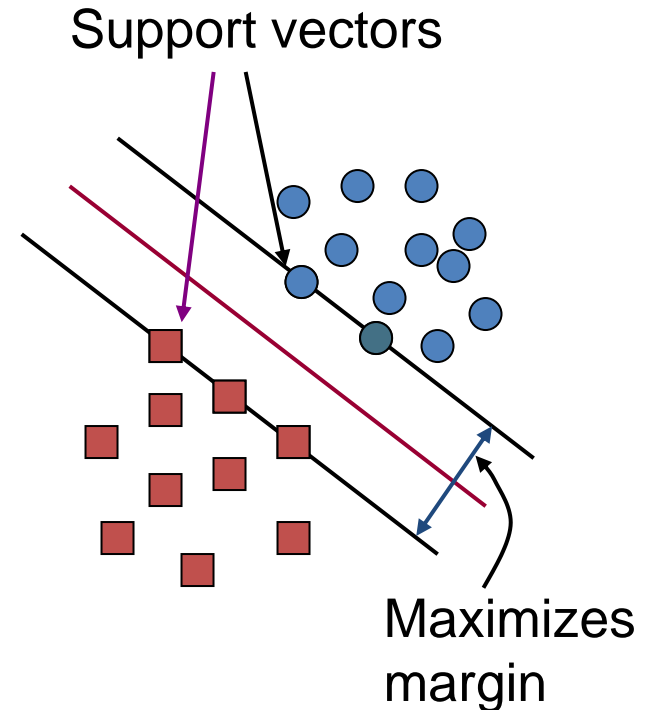
# Which Hyperplane?

- It seems good to have a fat separator between classes, since points near the separator represent very uncertain classification decisions
- If there are no points near the decision surface, then there are no uncertain classification decisions



# Support Vector Machine

- 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.
- Seen by many as the most successful current text classification method



# What we have covered so far

- Rocchio classifier
  - Classify by closeness to centroid
- kNN classifier
  - Classify by voting from most similar documents
- Naïve Bayes classifier
  - Classify by probability of generating a test document from probability models by training documents
- Support Vector Machine
  - Classify by hyperplane defined by support vectors (a few training documents)



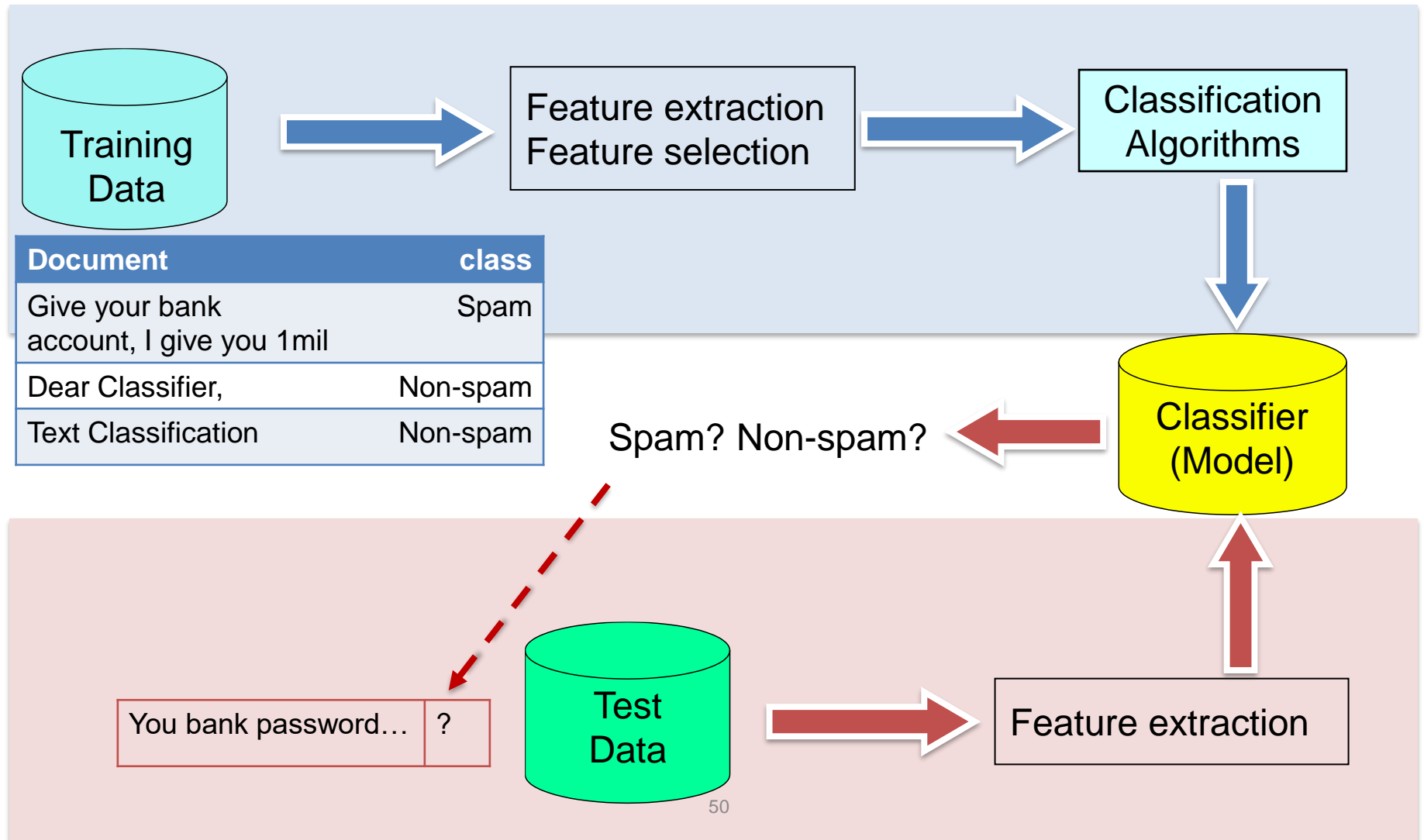


# We now have classifiers in hand

- Next is to **solve text classification problems**
- What are the potential issues?
  - Binary vs multi-class classification
  - Single-label or multi-label classification
  - Flat vs hierarchical classification
  - Balanced vs Imbalanced classification
- How to represent documents?
  - Feature extraction and feature selection
- Evaluation of classification results



# Classification Framework: Train vs Test



# Binary vs Multi-class Classification

- How many categories:
  - Binary classification: Yes or No
  - Multi-class classification: three or more categories
- Classifiers
  - All classifiers naturally handle binary classification problem
  - Not all classifiers are designed to handle multi-class problem, e.g., SVM
- Strategies
  - **One vs All** (or One vs Rest, One against All): One classifier is trained for each category and all documents in this category are positive examples; documents in ALL remaining categories are negative examples.
  - **One vs One**: train  $K(K - 1)/2$  classifiers for  $K$  categories. At prediction time, a voting scheme is applied: all  $K(K - 1) / 2$  classifiers are applied to an unseen document; the class that got the highest number of "+1" predictions gets predicted.

$a, b, c \Rightarrow (a, b), (b, c), (c, a)$



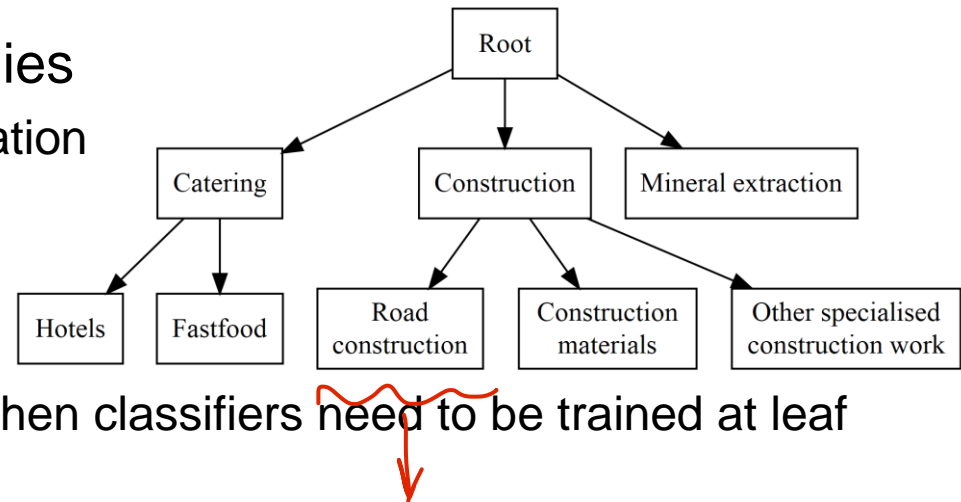
# Single-label vs Multi-label classification

- Binary classification
  - A document is assigned to one category
- When there are  $K$  categories ( $K > 2$ )
  - Single-label (non-overlapping categories): A document can be assigned **ONE of the  $K$  categories**
  - Multi-label: A document can be assigned to **0 to  $K$  categories** among the  $K$  categories
- Binary classification is more general from theoretical point of view
  - A multi-label classification problem can be transformed to multiple independent binary classification problem. (but, a classifier designed for multi-label classification cannot be applied to binary classification)



# Flat vs hierarchical classification

- Category Structure
  - Flat classification: categories are kind of independent from each other
  - Hierarchical classification: categories are organized into a **topical hierarchy**
- Hierarchical classification strategies
  - Ignore the structure -> flat classification
  - Make good use of the hierarchy
  - Classifier(s) is trained
    - At root and internal node
    - If binary classifiers are applied, then classifiers need to be trained at leaf nodes as well
  - **Blocking** issue: mistakes at higher levels will not be corrected at lower levels.



# Balanced vs Imbalanced classification

- Consider a binary classification problem
  - The same applies to multi-class classification problem
- The numbers of positive instances and negative instances in 100 training examples show the following distribution
  - Positive vs Negative = 50 vs 50
  - Positive vs Negative = 1 vs 99
- Classifiers are trained to minimize errors, i.e., the number of misclassifications.
  - For imbalanced classification, a classifier tends to predict every instances to be the majority class



# Strategies for imbalanced classification

- (Assuming negative class is the majority)
- **Instance weighting**
  - To assign different error classification costs to positive and negative training examples

*pos 1 / 99 neg*

*error in pos, give cost 10 / in negative*
- **Under-sampling:**
  - To select a subset of negative examples, e.g., random sampling
  - Biased sampling: to select negative examples that are expected to be close to the decision surface (or more similar to positive examples)
  - Cluster based Under-Sampling
    - All training examples are clustered by  $k$ -means
    - $k$  is the number of positive examples,  $k$ -means is initialized by  $k$  positive examples (each cluster seed is a positive example at beginning)
    - After clustering, clusters each contains all negative examples are discarded



# Strategies for imbalanced classification

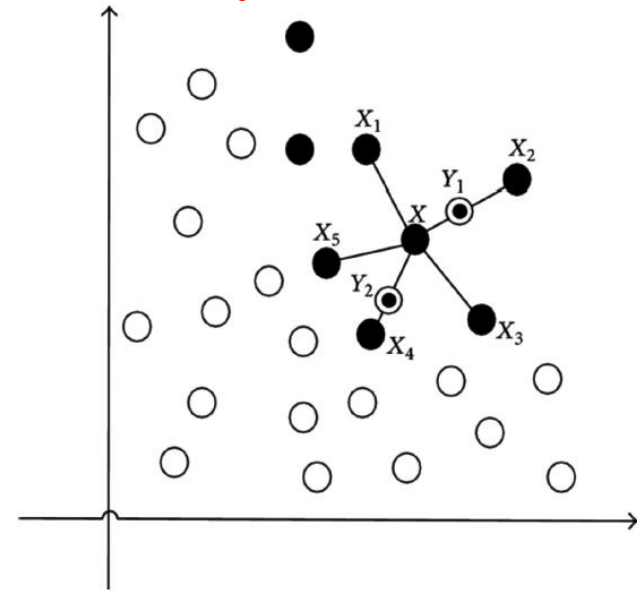
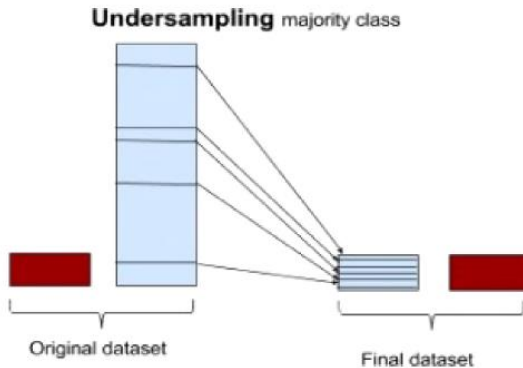
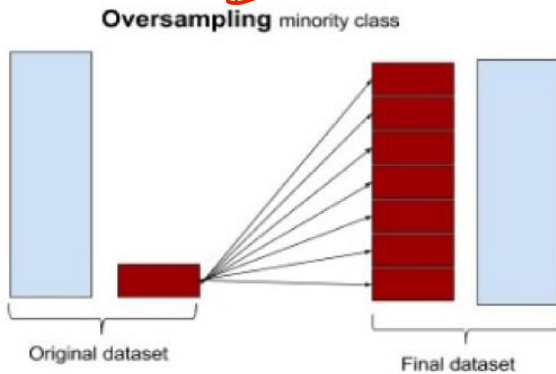
- **Over-sampling**
  - Replication does not help much
  - To (synthetically) generate more positive examples
- **Synthetic Minority Over-sampling Technique (SMOTE algorithm)**  
creates positive training instances synthetically
  - Given a positive training document, its  $k$  nearest neighbors among other positive training documents are first identified.
  - Let  $x_i$  be the feature vector of document  $d_i$ , and  $x_j$  be the feature vector of one of  $d_i$ 's  $k$  nearest neighbors. The feature vector of a synthetic document is created by  $(x_i + g(x_j - x_i))$ , where  $g$  is a random value between 0 and 1.





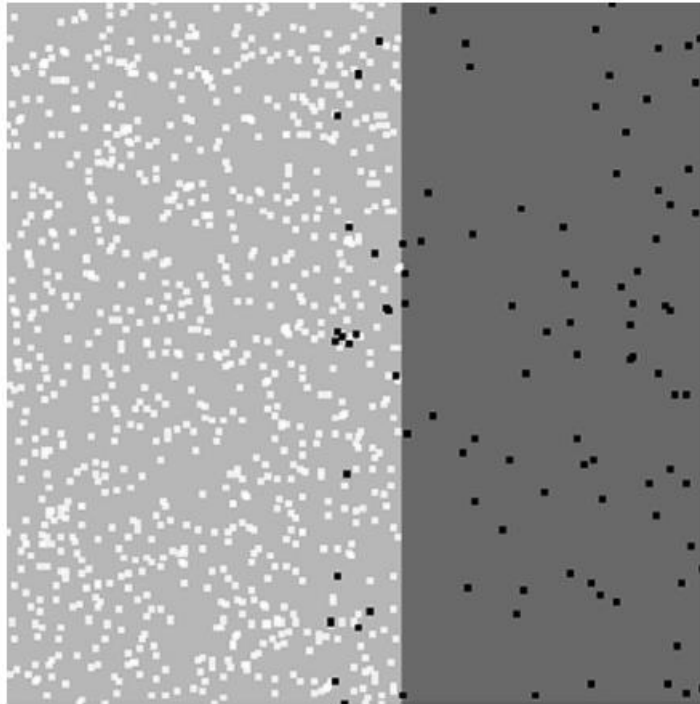
# Illustrations

SMOTE only for oversampling

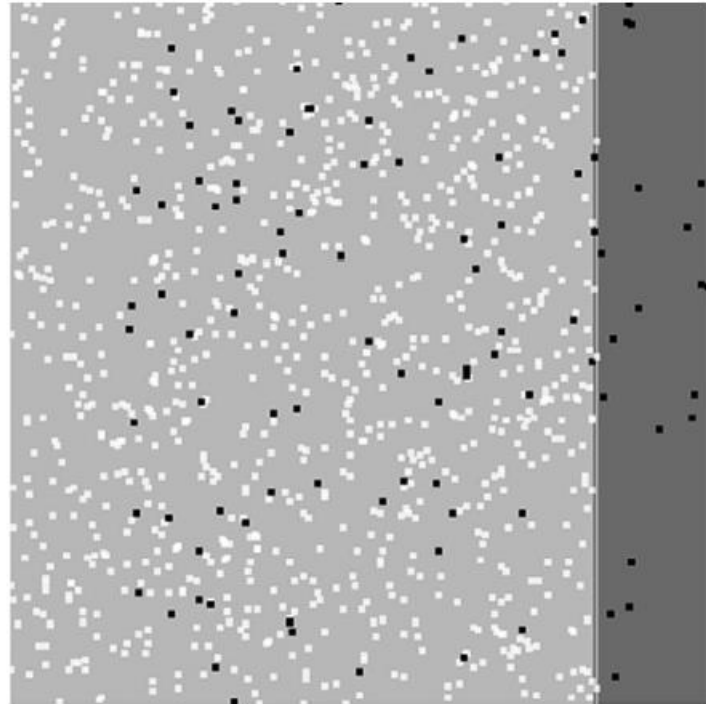


SMOTE Algorithm

# Overlapping or class separability



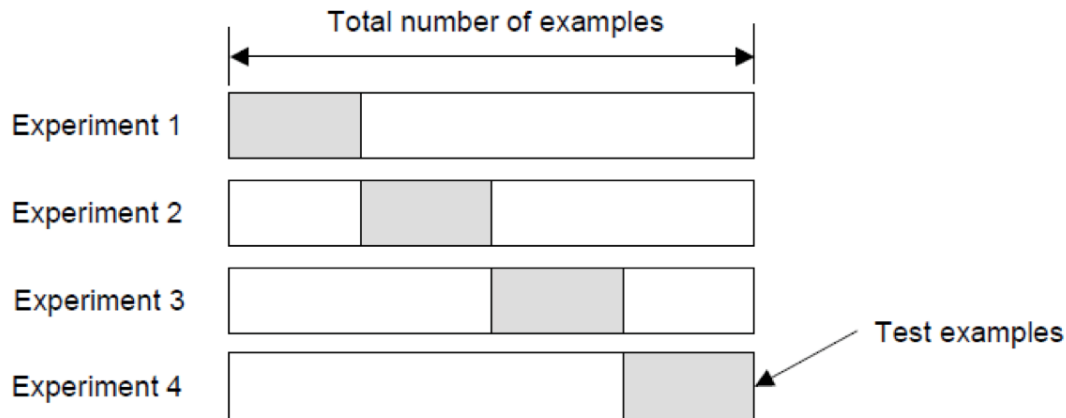
(a) 20% overlap



(b) 80% overlap

# Evaluating Categorization

- Evaluation must be done on test data that are independent of the training data
  - Usually a disjoint set of instances
- When a defined train/test split not available
  - $k$ -fold cross-validation, e.g. 4-fold cross-validation
  - Averaging results over multiple training and test splits of the overall data



# Classification Evaluation

- Precision, Recall, and F1 (for each category)

$$Precision = \frac{TP_i}{TP_i + FP_i}$$

$$Recall = \frac{TP_i}{TP_i + FN_i}$$

$$F_1 = \frac{2 \cdot Pr \cdot Re}{Pr + Re}$$

$$Accuracy = \frac{TP_i + TN_i}{TP_i + FN_i + FP_i + TN_i}$$

**Table II.** The Contingency Table for Category  $c_i$

Category $c_i$		Expert judgments	
		YES	NO
Classifier Judgments	YES	$TP_i$	$FP_i$
	NO	$FN_i$	$TN_i$



# Classification Evaluation

- Macro-average Precision/Recall (average over all categories)

*bad*

$$Pr^M = \frac{1}{|C|} \sum_{c_i \in C} Pr_i$$

$$Re^M = \frac{1}{|C|} \sum_{c_i \in C} Re_i$$

- Micro-average Precision/Recall (average over all instances/documents)

*in balanced case*

*good*

$$Pr^\mu = \frac{TP}{TP + FP}$$

$$Re^\mu = \frac{TP}{TP + FN}$$

**Table III.** The Global Contingency Table


Category set $C = \{c_1, \dots, c_{ C }\}$		Expert judgments	
		<b>YES</b>	<b>NO</b>
Classifier	<b>YES</b>	$TP = \sum_{i=1}^{ C } TP_i$	$FP = \sum_{i=1}^{ C } FP_i$
	<b>NO</b>	$FN = \sum_{i=1}^{ C } FN_i$	$TN = \sum_{i=1}^{ C } TN_i$



# Example of pooled table

class 1		
	truth: yes	truth: no
call: yes	10	40
call: no	10	940

class 2		
	truth: yes	truth: no
call: yes	90	110
call: no	10	790



pooled table		
	truth: yes	truth: no
call: yes	100	150
call: no	20	1730



# Example dataset: Reuters-21578

- One of the most popular data sets
- 21,578 documents
  - Appeared on Reuters newswire in 1987
- 9,603 training, 3,299 test articles
  - Indexed with categories by personnel from Reuters Ltd.
- 118 categories
  - An article may belong to more than one category
  - Only about 10 out of 118 categories are large

take the large 10, ignore others

8:27

```
<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>
```

11

# Comparison evaluation results: Sample

(a)		NB	Rocchio	kNN	SVM	
	micro-avg-L (90 classes)	80	85	86	89	
	macro-avg (90 classes)	47	59	60	60	
(b)		NB	Rocchio	kNN	trees	SVM
	earn	96	93	97	98	98
	acq	88	65	92	90	94
	money-fx	57	47	78	66	75
	grain	79	68	82	85	95
	crude	80	70	86	85	89
	trade	64	65	77	73	76
	interest	65	63	74	67	78
	ship	85	49	79	74	86
	wheat	70	69	77	93	92
	corn	65	48	78	92	90
	micro-avg (top 10)	82	65	82	88	92
	micro-avg-D (118 classes)	75	62	n/a	n/a	87

Evaluation measure:  $F_1$

Naive Bayes does pretty well, but some methods beat it consistently (e.g., SVM).



# Evaluation at different threshold settings

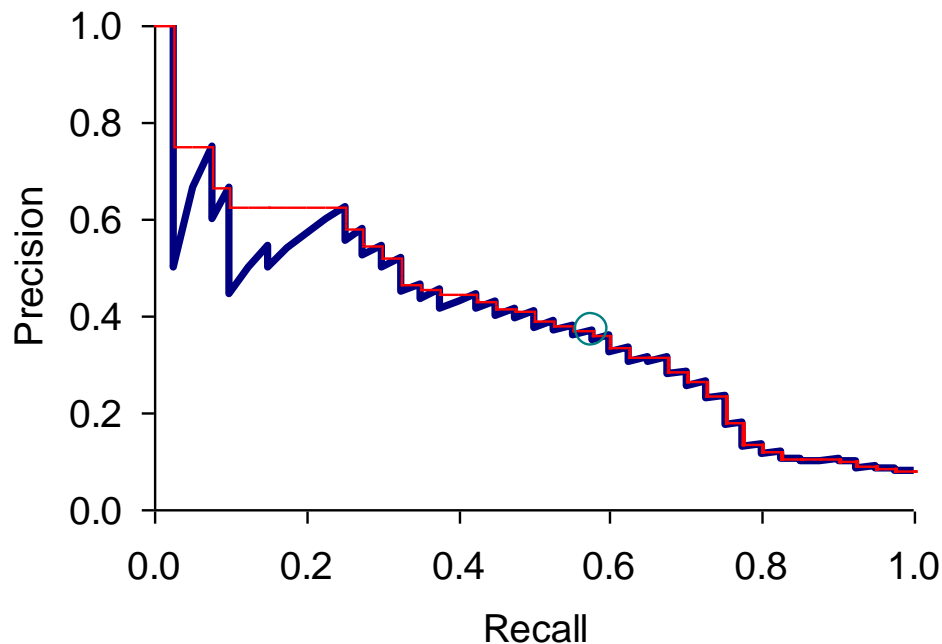
- Consider a classifier as a ranking function to compute a “Category Status Value” (CSV) for a given document  $d_j$  to category  $c_i$ 
  - For a given category, we rank the documents by CSV, the evidence for  $d_j \in c_i$
  - Then the decision of classification involves a setting of threshold on CSV
  - This is “document-ranking” classification
- Examples
  - Rocchio: ranking by distance to centroid
  - Naïve bayes: ranking in terms of probabilities
  - SVM: ranking by distance to decision hyperplane
- “Category-ranking” classification is, for a given document  $d_j$ , its CSV scores are ranked for the different categories in  $\mathcal{C} = \{c_1, c_2, \dots, c_i\}$



# Evaluation at different threshold settings

- Area under PR (Precision-Recall) Curve
  - The PR curve is created by plotting the Precision against Recall at various threshold settings.

*better to measure model ✓*



# Evaluation at different threshold settings

- AUR: Area under ROC (Receiver Operating Characteristic) Curve
  - The ROC curve is created by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings.
  - [https://en.wikipedia.org/wiki/Receiver\\_operating\\_characteristic](https://en.wikipedia.org/wiki/Receiver_operating_characteristic)
  - The true-positive rate is also known as sensitivity, **recall** or probability of detection
    - $TP/(TP+FN)$
  - The false-positive rate is also known as probability of false alarm
    - $FP/(FP+TN)$

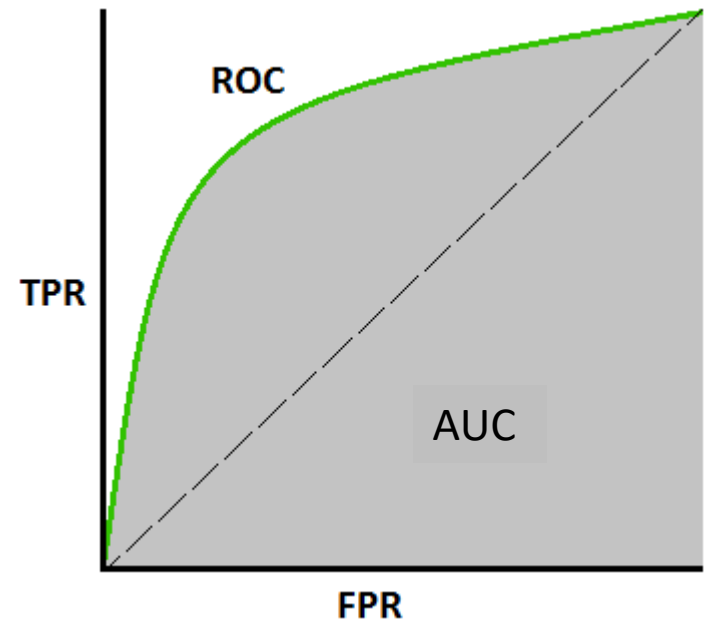


Image: <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>



# Feature Extraction and Feature Selection

- Feature extraction
  - What are the features best represent a document for classification
- Most widely used features
  - Words with TF-IDF weighting - Bag of words
  - Words with binary weighting? Not so effective
- Other features
  - N-gram, a sequence of consecutive words
    - Each n-gram is treated as one feature, can be weighted by TF-IDF as well
  - Phrases? Not so effective



# Feature Selection: Why?

- Text collections have a large number of features
  - 10,000 – 1,000,000 unique words ... and more
  - Some classifiers can't deal with 100,000 of features
- Reduces training time (classifier dependent)
  - Training time for some methods is quadratic or worse in the number of features
- Feature extraction
  - What are the features best indicative or discriminative for classification
  - Easy and effective feature selection technique: **document frequency**
    - Remove the words with small document frequency, e.g.,  $DF \leq 5$
    - There are lot of such words

cut the tail

# Feature selection: Information-Theoretic term selection

**Table I.** Main Functions Used for Term Space Reduction Purposes. Information Gain Is Also Known as *Expected Mutual Information*, and Is Used Under This Name by Lewis [1992a, page 44] and Larkey [1998]. In the  $RS(t_k, c_i)$  Formula,  $d$  Is a Constant Damping Factor.

Function	Denoted by	Mathematical form
DIA association factor	$z(t_k, c_i)$	$P(c_i   t_k)$
Information gain	$IG(t_k, c_i)$	$\sum_{c \in \{c_i, \bar{c}_i\}} \sum_{t \in \{t_k, \bar{t}_k\}} P(t, c) \cdot \log \frac{P(t, c)}{P(t) \cdot P(c)}$
Mutual information	$MI(t_k, c_i)$	$\log \frac{P(t_k, c_i)}{P(t_k) \cdot P(c_i)}$
Chi-square	$\chi^2(t_k, c_i)$	$\frac{ Tr  \cdot [P(t_k, c_i) \cdot P(\bar{t}_k, \bar{c}_i) - P(t_k, \bar{c}_i) \cdot P(\bar{t}_k, c_i)]^2}{P(t_k) \cdot P(\bar{t}_k) \cdot P(c_i) \cdot P(\bar{c}_i)}$
NGL coefficient	$NGL(t_k, c_i)$	$\frac{\sqrt{ Tr  \cdot [P(t_k, c_i) \cdot P(\bar{t}_k, \bar{c}_i) - P(t_k, \bar{c}_i) \cdot P(\bar{t}_k, c_i)]}}{\sqrt{P(t_k) \cdot P(\bar{t}_k) \cdot P(c_i) \cdot P(\bar{c}_i)}}$
Relevancy score	$RS(t_k, c_i)$	$\log \frac{P(t_k   c_i) + d}{P(\bar{t}_k   \bar{c}_i) + d}$
Odds ratio	$OR(t_k, c_i)$	$\frac{P(t_k   c_i) \cdot (1 - P(t_k   \bar{c}_i))}{(1 - P(t_k   c_i)) \cdot P(t_k   \bar{c}_i)}$
GSS coefficient	$GSS(t_k, c_i)$	$P(t_k, c_i) \cdot P(\bar{t}_k, \bar{c}_i) - P(t_k, \bar{c}_i) \cdot P(\bar{t}_k, c_i)$



# Feature selection: Information-Theoretic term selection

- Probabilities are interpreted on an event space of documents (e.g.,  $P(t_k, c_i)$  denotes the probability that, for a random document  $x$ , term  $t_k$  occur in  $x$  and  $x$  belongs to category  $c_i$ 
  - Estimated by counting occurrences in the training set.
- All functions are specified “locally” to a specific category  $c_i$ 
  - The best terms for  $c_i$  are the ones distributed most differently in the sets of positive and negative examples of  $c_i$
- To assess the value of a term  $t_k$  in a “global,” category independent sense:
  - We can take sum or weighted sum (based on category size) of  $t_k$ ’s category-specific values or take the maximum among all categories.
  - Select the subset of terms with high scores as feature space.



# Resources

- Introduction to Information Retrieval
  - Chapters 13, 14
- Overview of text classification and performance evaluation
  - Fabrizio Sebastiani. Machine Learning in Automated Text Categorization. ACM Computing Surveys, 34(1):1-47, 2002.
  - Tom Mitchell, Machine Learning. McGraw-Hill, 1997. (Clear simple explanation of Naïve Bayes)
- Imbalanced classification
  - López, Victoria & Fernández, Alberto & García, Salvador & Palade, Vasile & Herrera, Francisco. (2013). An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. Information Sciences. 250. 113–141.
  - Harsurinder Kaur, Husanbir Singh Pannu, and Avleen Kaur Malhi. 2019. A Systematic Review on Imbalanced Data Challenges in Machine Learning: Applications and Solutions. ACM Comput. Surv. 52, 4, Article 79 (August 2019), 36 pages.
- Sample benchmark datasets
  - Reuters-21578 – the most famous text classification evaluation set (<http://www.daviddlewis.com/resources/testcollections/reuters21578/>)
  - 20 New Group data (<http://people.csail.mit.edu/jrennie/20Newsgroups/>)

