# Partially Supervised Learning

M.M. Pedram
pedram@tmu.ac.ir
Tarbiat Moallem University of Tehran
(Fall 2011)

# Based on:

- http://www.cs.uic.edu/~liub/WebMiningBook.html
  - B. Liu, Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data, Springer-Verlag Berlin Heidelberg 2011.
    Chapter 3 and 5 presentations.

and

- Tom M. Mitchell, Semi-Supervised Learning over Text, Presentation, 2006.
  - Kamal Nigam, Andrew McCallum, Sebastian Thrun and Tom Mitchell, "*Learning to Classify Text from Labeled and Unlabeled Documents",* In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pp. 792-799,1998.
  - Kamal Nigam, Andrew McCallum, Sebastian Thrun and Tom Mitchell, "Text Classification from Labeled and Unlabeled Documents using EM", *Machine Learning*, 39(2/3), pp,103-134, 2000.
  - (longer version)

# Outline

- Fully supervised learning (traditional classification)
- Partially supervised learning (or classification)

# Fully supervised learning (traditional classification)

# Document Classification

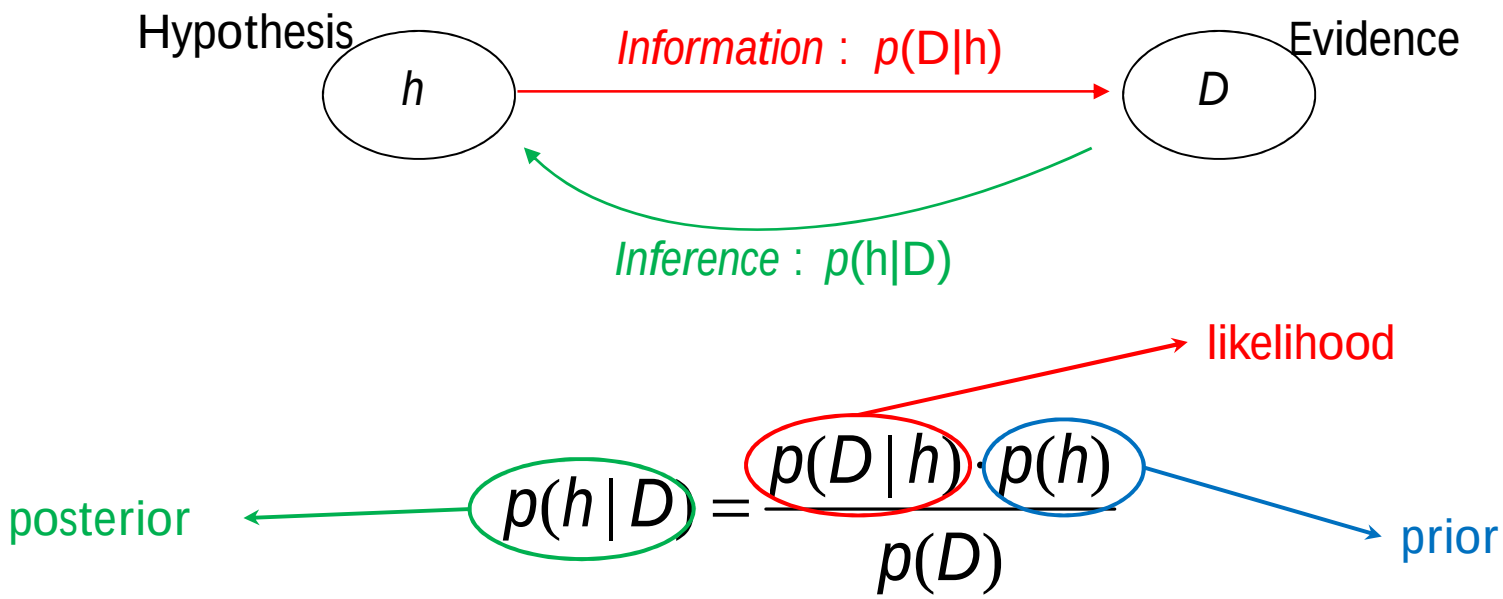
- Spam filtering, relevance rating, web page classification, ...
- $f$ : Doc $\rightarrow$ Class

  $(x_i, y_i)$

Question:

- Is it possible to classify documents by unlabeled data, i.e. $(x_i, -)$ ?

# Bayes Theorem




- $p(h)$ = prior probability of hypothesis $h$
- $p(D)$ = prior probability of training data $D$
- $p(h|D)$ = probability of $h$ given $D$
- $p(D|h)$ = probability of $D$ given $h$

# Bayesian classification

- Probabilistic view: Supervised learning can naturally be studied from a probabilistic point of view.
- Let $A_1$ through $A_k$ be attributes with discrete values. The class is C.
- Given a test example *d* with observed attribute values $a_1$ through $a_k$.
- Classification is basically to compute the following posteriori probability. The prediction is the class $c_j$ such that

$$\Pr(C = c_j \mid A_1 = a_1, \ldots, A_{|A|} = a_{|A|})$$

is maximal.

# Apply Bayes' Rule

$$\Pr(C = c_j \mid A_1 = a_1, \ldots, A_{|A|} = a_{|A|}) = \frac{\Pr(A_1 = a_1, \ldots, A_{|A|} = a_{|A|} \mid C = c_j)\Pr(C = c_j)}{\Pr(A_1 = a_1, \ldots, A_{|A|} = a_{|A|})}$$

$$= \frac{\Pr(A_1 = a_1, \ldots, A_{|A|} = a_{|A|} \mid C = c_j)\Pr(C = c_j)}{\sum_{r=1}^{|C|} \Pr(A_1 = a_1, \ldots, A_{|A|} = a_{|A|} \mid C = c_r)\Pr(C = c_r)}$$

- $\Pr(C=c_j)$ is the class *prior* probability: easy to estimate from the training data.

# Computing probabilities

- The denominator $P(A_1=a_1,...,A_k=a_k)$ is irrelevant for decision making since it is the same for every class.

- We only need $P(A_1=a_1,...,A_k=a_k \mid C=c_i)$, which can be written as

  $Pr(A_1=a_1|A_2=a_2,...,A_k=a_k, C=c_j)* Pr(A_2=a_2,...,A_k=a_k \mid C=c_j)$

- Recursively, the second factor above can be written in the same way, and so on.

- Now an assumption is needed.

# Conditional independence assumption

- All attributes are conditionally independent given the class $C = c_j$ .
- Formally, we assume,

  $\Pr(A_1=a_1 \mid A_2=a_2, ..., A_{|A|}=a_{|A|}, C=c_j) = \Pr(A_1=a_1 \mid C=c_j)$

  and so on for $A_2$ through $A_{|A|}$. I.e.,

$$\Pr(A_1 = a_1,..., A_{|A|} = a_{|A|} \mid C = c_i) = \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_j)$$

# Final naïve Bayesian classifier

$$\Pr(C = c_j \mid A_1 = a_1, \ldots, A_{|A|} = a_{|A|}) = \frac{\Pr(C = c_j)\prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_j)}{\sum_{r=1}^{|C|} \Pr(C = c_r)\prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_r)}$$

- We are done!
- How do we estimate $P(A_i = a_i \mid C=c_j)$? Easy!.

# Classify a test instance

- If we only need a decision on the most probable class for the test instance, we only need the numerator as its denominator is the same for every class.
- Thus, given a test example, we compute the following to decide the most probable class for the test instance

$$c = \arg\max_{c_j} \Pr(c_j) \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_j)$$

# Example

- Compute all probabilities required for classification:

| A | B | C |
|---|---|---|
| m | b | t |
| m | s | t |
| g | q | t |
| h | s | t |
| g | q | t |
| g | q | f |
| g | s | f |
| h | b | f |
| h | q | f |
| m | b | f |

$Pr(C = t) = 1/2,$ $Pr(C = f) = 1/2$

$Pr(A=m \mid C=t) = 2/5$ $Pr(A=g \mid C=t) = 2/5$ $Pr(A=h \mid C=t) = 1/5$

$Pr(A=m \mid C=f) = 1/5$ $Pr(A=g \mid C=f) = 2/5$ $Pr(A=h \mid C=n) = 2/5$

$Pr(B=b \mid C=t) = 1/5$ $Pr(B=s \mid C=t) = 2/5$ $Pr(B=q \mid C=t) = 2/5$

$Pr(B=b \mid C=f) = 2/5$ $Pr(B=s \mid C=f) = 1/5$ $Pr(B=q \mid C=f) = 2/5$

Now we have a test example:

$A = m \quad B = q \quad C = ?$

# Example (cont …)

- For C = t, we have

$$\Pr(C=t)\prod_{j=1}^{2}\Pr(A_j=a_j\mid C=t)=\frac{1}{2}\times\frac{2}{5}\times\frac{2}{5}=\frac{2}{25}$$

- For class C = f, we have

$$\Pr(C=f)\prod_{j=1}^{2}\Pr(A_j=a_j\mid C=f)=\frac{1}{2}\times\frac{1}{5}\times\frac{2}{5}=\frac{1}{25}$$

- C = t is more probable $\Rightarrow$ t is the final class.

# Additional issues

- **Numeric attributes**: Naïve Bayesian learning assumes that all attributes are categorical. Numeric attributes need to be discretized.

- **Zero counts**: An particular attribute value never occurs together with a class in the training set. We need smoothing.

$$\Pr(A_i = a_i \mid C = c_j) = \frac{n_{ij} + l}{n_j + l\, n_i}$$

  - $\lambda$ is commonly set to $\lambda = 1/n$, where $n$ is the total number of examples in the training set $D$.

- **Missing values**: Ignored.

# On naïve Bayes classifier

- Advantages:
  - Easy to implement
  - Very efficient
  - Good results obtained in many applications
- Disadvantages
  - Assumption: class conditional independence,
    - therefore loss of accuracy when the assumption is seriously violated (those highly correlated data sets)

# Naïve Bayes for Text Classification

# Text classification/categorization

- Due to the rapid growth of online documents in organizations and on the Web, automated document classification has become an important problem.
- Different techniques can be applied to text classification, but they are not as effective as the next three methods.
- We first study a naïve Bayesian method specifically formulated for texts, which makes use of some text specific features.
- However, the ideas are similar to the preceding method.

# Probabilistic framework

- **Generative model**: Each document is generated by *a parametric distribution* governed by *a set of hidden parameters*.
  - Training data is used to estimate these parameters.
  - The parameters are then applied to classify each test document using Bayes rule by calculating the *posterior probability* that the distribution associated with a class (represented by the unobserved class variable) would have generated the given document.

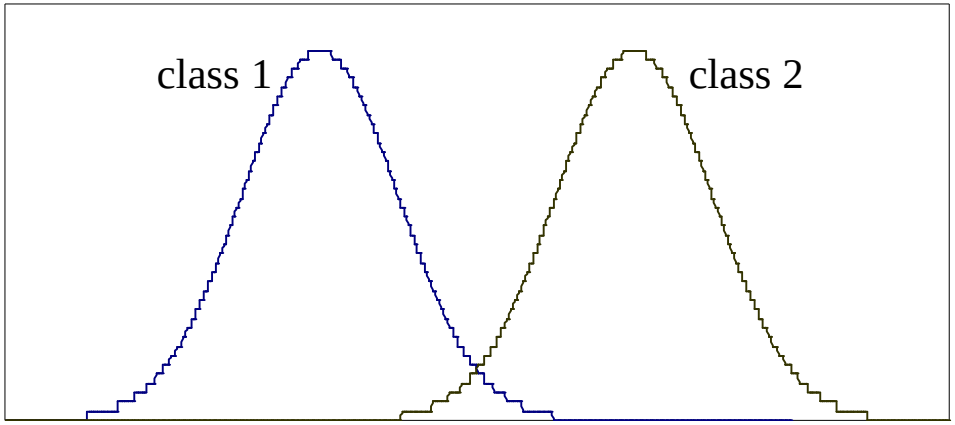
- The generative model makes two assumptions
  1. The data (or the text documents) are generated by a mixture model,
  2. There is one-to-one correspondence between mixture components and document classes.

# Mixture model

- A **mixture model** models the data with a number of statistical distributions.
  - Intuitively, each distribution corresponds to a data cluster and the parameters of the distribution provide a description of the corresponding cluster.
- Each distribution in a mixture model is also called a **mixture component**.
- The distribution/component can be of any kind.

# An example

- The figure shows a plot of the **probability density function** of a 1-dimensional data set (with 2 classes) generated by
  - a mixture of two Gaussian distributions,
  - one Gaussian distribution per class, whose parameters (denoted by $q_i$) are the mean ($m_i$) and the standard deviation ($s_i$), i.e., $q_i = (m_i, s_i)$.



# Mixture model (cont …)

- K:The number of mixture components (or distributions) in a mixture model,
- $j$ : Index of the mixture component,
- $q_j$ : The parameter of the $j$-th distribution.
- $j_j$ : The mixture weight (or *mixture probability*) of the mixture component $j$.
- Q :The set of parameters of all components:

$$Q = \{j_1, j_2, \ldots, j_K, q_1, q_2, \ldots, q_K\}$$

- How does the model generate documents?

# Document generation

- Due to one-to-one correspondence, each class corresponds to a mixture component. The mixture weights are *class prior probabilities*

$$j_j = \Pr(c_j \mid \Theta)$$

- The mixture model generates each document $d_i$ by:
  i. selecting a mixture component (or class) according to class prior probabilities (i.e., mixture weights), $j_j = \Pr(c_j \mid \Theta)$.
  ii. having this selected mixture component ($c_j$) generate a document $d_i$ according to its parameters, with distribution $\Pr(d_i \mid c_j; \Theta)$ or more precisely $\Pr(d_i \mid c_j; q_j)$.

$$\Pr(d_i \mid \Theta) = \sum_{j=1}^{|C|} \Pr(c_j \mid \Theta) \Pr(d_i \mid c_j; \Theta)$$

# Model text documents

- The naïve Bayesian classification treats each document as a "*bag of words*".


| | |
|---|---|
| aardvark | 0 |
| about | 2 |
| all | 2 |
| Africa | 1 |
| apple | 0 |
| anxious | 0 |
| ... | |
| gas | 1 |
| ... | |
| oil | 1 |
| … | |
| Zaire | 0 |

# Model text documents

- The generative model makes the following further assumptions:
  I. Words of a document are generated independently of context given the class label. (The familiar naïve Bayes assumption used before.)
  II. The probability of a word is independent of its position in the document.
  III. The document length is chosen independent of its class.

Þ each document can be regarded as generated by a multinomial distribution.

# Reminder!

- A *multinomial trial* is a process that can result in any of k outcomes, where $k \geq 2$.
- The probabilities of the k outcomes are denoted by $p_1, p_2, \ldots, p_k$.
  - The rolling of a die is a multinomial trial, with six possible outtcomes 1, 2, 3, 4, 5, 6.
    - For a fair die, $p_1 = p_2 = \ldots = p_k = 1/6$.
- Assume:
  - *n* independent trials are conducted, each with the k possible outcomes and the k probabilities, $p_1, p_2, \ldots, p_k$.
  - Number the outcomes 1, 2, 3, …, k.
  - $X_1, X_2, \ldots, X_k$ : the number of trials that result in that outcome.
    - in rolling toss example:
      - $X_4$: the number of trials that result in appearing 4.

# Reminder!

*Note:* $X_1, X_2, \ldots, X_k$ are discrete random variables.

- The collection of $X_1, X_2, \ldots, X_k$ is said to have the *multinomial distribution* with parameters $n, p_1, p_2, \ldots, p_k$.

# Multinomial distribution

- With the assumptions (given in slide 25), each document can be regarded as generated by a multinomial distribution.
  - In other words, each document is drawn from a multinomial distribution of words with as many independent trials as the length of the document.

# Nomenclature (In text doc. classification)

- n : the length of a document.
- $w_t$ : the $t$-th word in the vocabulary.
  - The words are from a given vocabulary $V = \{w_1, w_2, \ldots, w_{|V|}\}$.
- $w_{d_i,m}$ : the word in position m of document $d_i$.
- |V|: The number of words in the vocabulary.
  - Note: $|V|=k$ is the number of outcomes (words).
- $C = \{c_1, c_2, \ldots, c_{|C|}\}$ : The set of (document) classes
- $p_i$ : The probability of occurrence of the word $w_t$ in a document class $c_j$.

$$p_t = \Pr(w_t \mid c_j; \Theta)$$

- $X_t$ : a random variable as the number of times that word $w_t$ appears in a document.
- $N_{ti}$ : the number of times that word $w_t$ occurs in document $d_i$ .
- $|d_i|$ : document length.
- $\Pr(|d_i|)$ : the probability of document length.
- $D_j$ : the subset of (the labeled training) documents for class $c_j$ .
- $D = \{D_1, D_2, \ldots, D_{|C|}\}$ : the labeled training data.

# Use probability function of multinomial distribution

- We can directly apply the probability function of the multinomial distribution to find the probability of a document given its class ($\Pr(|d_i|)$, which is assumed to be independent of the class):

$$\Pr(d_i \mid c_j;\Theta) = \Pr(\langle w_{d_i,1},\ldots,w_{d_i,|d_i|}\rangle \mid c_j;\Theta)$$

$$= \Pr(|d_i|)\prod_{k=1}^{|d_i|}\Pr(w_{d_i,k} \mid c_j;\Theta;\, w_{d_i,q}, q<k)$$

$$= \Pr(|d_i|)\prod_{k=1}^{|d_i|}\Pr(w_{d_i,k} \mid c_j;\Theta)$$

$$\sum_{t=1}^{|V|} N_{ti} = |d_i|$$

$$\sum_{t=1}^{|V|}\Pr(w_t \mid c_j;\Theta) = 1$$

# Parameter estimation

- The parameters are estimated based on empirical counts.

$$\Pr(w_t \mid c_j; \hat{\Theta}) = \frac{\text{the number of times that } w_t \text{ occurs in the training data } D_j \text{ (of class } c_j)}{\text{the total number of word occurrences in the training data for that class}}$$

$$= \frac{\sum_{i=1}^{|D|} N_{ti} \Pr(c_j \mid d_i)}{\sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N_{si} \Pr(c_j \mid d_i)}$$

Note: $\Pr(c_j \mid d_i) = \begin{cases} 1 & \text{for each document in } D_j \\ 0 & \text{documents in of other classes} \end{cases}$

back

# Parameter estimation (cont …)

*Reminder*

- *Additive smoothing*, also called *Laplace smoothing* or *Lidstone smoothing*, is a technique used to smooth categorical data.
- Given an observation $\mathbf{x} = (x_1, \ldots, x_{no\_of\_classes})$ from a multinomial distribution with $n$ trials and parameter vector

$$q = (q_1, \ldots, q_{no\text{-}of\text{-}classes})$$

a "smoothed" version of the data gives the estimator:

$$\hat{q}_i = \frac{x_i + l}{n + l \,.\, no\_of\_classes}$$

where $\lambda > 0$ is the smoothing parameter. $\lambda = 0$ corresponds to no smoothing. as the resulting estimate will be between the empirical estimate $x_i/n$, and the uniform probability $1/_{no\text{-}of\text{-}classes}$.

# Parameter estimation (cont …)

- In order to handle 0 counts for infrequent occurring words that do not appear in the training set, but may appear in the test set, we need to smooth the probability. *Lidstone* smoothing, $0 \leq l \leq 1$

$$\Pr(w_t \mid c_j; \hat{\Theta}) = \frac{l + \sum_{i=1}^{|D|} N_{ti} \Pr(c_j \mid d_i)}{l\,|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N_{si} \Pr(c_j \mid d_i)}.$$

# Parameter estimation (cont …)

- Class prior probabilities, which are mixture weights $j_j$, can be easily estimated using training data:

$$\Pr(c_j \mid \hat{\Theta}) = \frac{\text{the training data } D_j \text{ (of class c}_j\text{)}}{\text{the total number of the training data}}$$

$$= \frac{\sum_{i=1}^{|D|} \Pr(c_j \mid d_i)}{|D|}$$

go to

# Classification

- Given a test document $d_i$, from the followings

$$\Pr(d_i \mid \Theta) = \sum_{j=1}^{|C|} \Pr(c_j \mid \Theta) \Pr(d_i \mid c_j;\ \Theta)$$

$$\Pr(d_i \mid c_i; \Theta) = \Pr(|d_i|) \prod_{k=1}^{|d_i|} \Pr(w_{d_i,k} \mid c_j; \Theta)$$

the probability of occurrence class $c_j$ is derived:

$$\Pr(c_j \mid d_i; \hat{\Theta}) = \frac{\Pr(c_j \mid \hat{\Theta})\ \ \Pr(d_i \mid c_j; \hat{\Theta})}{\Pr(d_i \mid \hat{\Theta})}$$

$$= \frac{\Pr(c_j \mid \hat{\Theta}) \prod_{k=1}^{|d_i|} \Pr(w_{d_i,k} \mid c_j; \hat{\Theta})}{\sum_{r=1}^{|C|} \Pr(c_r \mid \hat{\Theta}) \prod_{k=1}^{|d_i|} \Pr(w_{d_i,k} \mid c_r; \hat{\Theta})}$$

# Classification

- If the final classifier is to classify each document into a single class, the class with the highest posterior probability is selected:

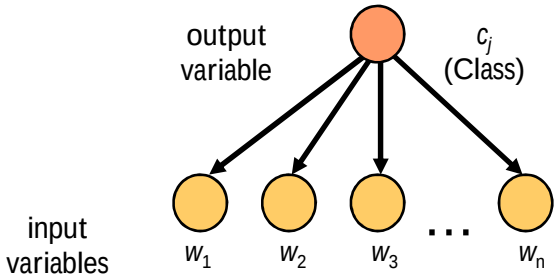
$$\textit{classifier output} = \arg\max_{c_j \in C} \Pr(c_j \mid d_i; \hat{\Theta})$$

# Classification

- ***Train***

  For each class $c_j$ of documents

  1. Estimate $p(c_j)$
  2. For each word $w_i$ estimate $p(w_i \mid c_j)$




- ***Classify (doc)***

  Assign *doc* to most probable class

$$\arg\max_j \; p(c_j, a_1, a_2, \ldots) = p(c_j) \prod_{w_i \in doc} p(w_i \mid c_j)$$

*or*

$$\arg\max_j \; p(a_1, a_2, \ldots \mid c_j) = \prod_{w_i \in doc} p(w_i \mid c_j)$$

Ø Assumption: words are conditionally independent, given class.

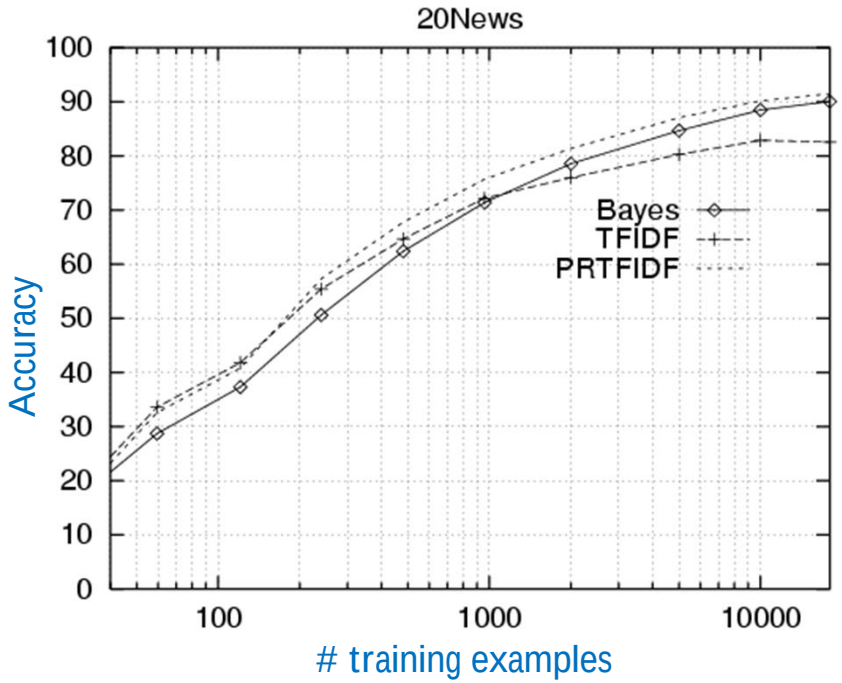
# Classification

**Twenty NewsGroups**

Given 1000 training documents from each group
Learn to classify new documents according to which newsgroup it came from

| | |
|---|---|
| comp.graphics | misc.forsale |
| comp.os.ms-windows.misc | rec.autos |
| comp.sys.ibm.pc.hardware | rec.motorcycles |
| comp.sys.mac.hardware | rec.sport.baseball |
| comp.windows.x | rec.sport.hockey |
| | |
| alt.atheism | sci.space |
| soc.religion.christian | sci.crypt |
| talk.religion.misc | sci.electronics |
| talk.politics.mideast | sci.med |
| talk.politics.misc | |
| talk.politics.guns | |

Naive Bayes: 89% classification accuracy



# Discussions

- Most assumptions made by naïve Bayesian learning are violated to some degree in practice.
  - words in a document are clearly not independent of each other.
  - the mixture model assumption of one-to-one correspondence between classes and mixture components may not be true because a class may contain documents from multiple topics.
- Despite such violations, Naïve Bayesian learning is extremely efficient:
  - researchers have shown that naïve Bayesian learning produces very accurate models.
  - It scans the training data only once to estimate all the probabilities required for classification.
  - It can be used as an incremental algorithm as well.The model can be updated easily as new data comes in because the probabilities can be conveniently revised.

# Discussions

- The main problem is the mixture model assumption. When this assumption is seriously violated, the classification performance can be poor.

# Partially supervised Learning

# Partially supervised Learning

- Partially supervised learning (or classification)
  - Learning with a small set of *labeled* examples and a large set of *unlabeled* examples (LU learning or semi-supervised learning)
    - In this learning setting, there is a small set of labeled examples of every class, and a large set of unlabeled examples.
    - The objective is to make use of the unlabeled examples to improve learning.
  - Learning with positive and unlabeled examples (no labeled negative examples) (PU learning).
    - This problem assumes two-class classification.
    - The training data only has a set of labeled positive examples and a set of unlabeled examples, but no labeled negative examples.
    - The objective is to build an accurate classifier without labeling any negative examples.

- These two learning problems do not need full supervision, and thus are able to reduce the labeling effort.

# Learning from a small labeled set and a large unlabeled set

## LU learning

# Unlabeled Data

- One of the bottlenecks of classification is the labeling of a large set of examples (data records or text documents).
  - Often done manually
  - Time consuming
- Can we label only a small number of examples and make use of a large number of unlabeled examples to learn?
- Possible in many cases.

# Why unlabeled data are useful?

- Unlabeled data are usually plentiful, labeled data are expensive.

- Unlabeled data provide information about the joint probability distribution over words and collocations (in texts).
  - using only the labeled data we find that documents containing the word "homework" tend to belong to a particular class. If we use this fact to classify the unlabeled documents, we may find that "lecture" co-occurs with "homework" frequently in the unlabeled set. Then, "lecture" may also be an indicative word for the class. Such correlations provide a helpful source of information to increase classification accuracy, especially when the labeled data are scarce.

- We will use text classification to study this problem.

# Why unlabeled data are useful?

Labeled Data

Unlabeled Data

Documents containing “homework” tend to belong to the positive class

DocNo: k ClassLabel: Positive
……
……homework….
…

DocNo: m ClassLabel: Positive
……
……homework….
…

DocNo: n ClassLabel: Positive
……
……homework….
…

DocNo: x (ClassLabel: Positive)
……
……homework….
…lecture….

DocNo: y (ClassLabel: Positive)
……lecture…..
……homework….
…

DocNo: z ClassLabel: Positive
……
……homework….
……lecture….

# How to use unlabeled data

- One way is to use the EM algorithm
  - EM: Expectation Maximization
- The EM algorithm is a popular iterative algorithm for *maximum likelihood estimation* in problems with *missing data.*
- The EM algorithm consists of two steps,
  - *Expectation* step: filling in the missing data based on the current estimation of the parameters.
  - *Maximization* step: calculate a new maximum *a posteriori* (or likelihood) estimate for the parameters.

# Incorporating unlabeled Data with EM

- Kamal Nigam, Andrew McCallum, Sebastian Thrun and Tom Mitchell, "Text Classification from Labeled and Unlabeled Documents using EM", *Machine Learning*, 39(2/3), pp,103-134, 2000.
- Basic EM
- Augmented EM with weighted unlabeled data
- Augmented EM with multiple mixture components per class

# EM – Expectation / Maximization

- EM is a *class of algorithms* *that is used to estimate a probability distribution in the presence of missing values*.
- Using it, requires an assumption on the underlying probability distribution.
- The algorithm can be very sensitive to this assumption and to the starting point (that is, the initial guess of parameters).
- It is a hill-climbing algorithm and converges to a local maximum of the likelihood function.
- That the EM algorithm is not really a specific “algorithm”, but is a framework or strategy.

# The Likelihood Function

- Set of training data D,
- A parametric family of models w/ parameters $\theta$,
- We want $\theta$ that maximizes $Pr(\theta \mid D)$,
- Bayes:

$$Pr(\theta \mid D) = Pr(D \mid \theta) \cdot Pr(\theta) / Pr(D)$$

  - $P(D \mid \theta)$ when viewed as a function of $\theta$ is the *likelihood function*:

$$L(\theta;D) = Pr(D \mid \theta)$$

Sometimes: $L(\theta,D) = L(D \mid \theta)$

- The likelihood function is NOT a probability distribution over $\theta$ and its integral / sum need not be 1.0,
- The M.L. (maximum likelihood) $\theta$, is the one that maximizes the likelihood function (without regard for a prior on $\theta$)

# Algorithm Outline

1. Train a classifier with only the labeled documents.
2. Use it to probabilistically classify the unlabeled documents.
3. Use ALL the documents to train a new classifier.
4. Iterate steps 2 and 3 to converge.

# The idea

- The documents in the unlabeled set (denoted by U) can be regarded as having missing class labels.
- The parameters that EM estimates in this case are the probability of each word  given a class and the class prior probabilities.
- EM here can also be seen as a clustering method with some initial seeds (labeled data) in each cluster. The class labels of the seeds indicate the class labels of the resulting clusters.
- Two assumptions are made in the derivation of the algorithm, which are in fact the two mixture model assumptions:
  1. the data is generated by a mixture model,
  2. there is a one-to-one correspondence between mixture components and classes.

# The EM algorithm with naïve Bayes classifier

**Algorithm** EM($L$, $U$)

1. Learn an initial naïve Bayesian classifier $f$ from only the labeled set $L$ (Equations (3-31) and (3-32));

$$\Pr(w_t \mid c_j; \hat{\Theta}) = \frac{l + \sum_{i=1}^{|D|} N_{ti} \Pr(c_j \mid d_i)}{l\,|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N_{si} \Pr(c_j \mid d_i)} \qquad \Pr(c_j \mid \hat{\Theta}) = \frac{\sum_{i=1}^{|D|} \Pr(c_j \mid d_i)}{|D|}$$

2. **repeat**

   // E-Step

3. **for** each example $d_i$ in $U$ **do**

4. Using the current classifier $f$ to compute $\Pr(c_j|d_i)$ (Equation (3-33));

$$\Pr(c_j \mid d_i; \hat{\Theta}) = \frac{\Pr(c_j \mid \hat{\Theta})\ \Pr(d_i \mid c_j; \hat{\Theta})}{\Pr(d_i \mid \hat{\Theta})} = \frac{\Pr(c_j \mid \hat{\Theta}) \prod_{k=1}^{|d_i|} \Pr(w_{d_i,k} \mid c_j; \hat{\Theta})}{\sum_{r=1}^{|C|} \Pr(c_r \mid \hat{\Theta}) \prod_{k=1}^{|d_i|} \Pr(w_{d_i,k} \mid c_r; \hat{\Theta})}$$

5. **end**

   // M-Step

6. Learn a new naïve Bayesian classifier $f$ from $L \cup U$ by computing $\Pr(c_j)$ and $\Pr(w_t|c_j)$ ((3-31) and (3-32));

$$\Pr(w_t \mid c_j; \hat{\Theta}) = \frac{l + \sum_{i=1}^{|D|} N_{ti} \Pr(c_j \mid d_i)}{l\,|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N_{si} \Pr(c_j \mid d_i)} \qquad \Pr(c_j \mid \hat{\Theta}) = \frac{\sum_{i=1}^{|D|} \Pr(c_j \mid d_i)}{|D|}$$

7. **until** the classifier parameters stabilize;

**Return** the classifier $f$ from the last iteration.

# Example

- Find the naïve Bayesian classifier *f* based on labeled and unlabeled examples.
- Does the result of classification change if the 3rd record is deleted?

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | y |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | ? |
| 0 | 1 | 0 | 1 | ? |

# The problem

- It has been shown that the EM algorithm works well if
  - The two mixture model assumptions for a particular data set are true.
- Although naïve Bayesian classification makes additional three assumptions, it performs surprisingly well despite the obvious violation of the assumptions.
- The two mixture model assumptions, however, can cause major problems when they do not hold. In many real-life situations, they may be violated.
- The first assumption above is usually not a problem, while the second assumption is critical.
- It is often the case that a class (or topic) contains a number of sub-classes (or sub-topics).
  - For example, the class Sports may contain documents about different sub-classes of sports, Baseball, Basketball, Tennis, and Softball.

# The problem

- Note:
  - If the second condition holds, EM works very well and is particularly useful when the labeled set is very small, e.g., fewer than five labeled documents per class. In such cases, every iteration of EM is able to improve the classifier dramatically.
  - If the second condition does not hold, the unlabeled set hurts learning instead of helping it.
- Some methods to deal with the problem:
  - Weighting the influence of unlabeled examples
  - Finding Mixture Components

# Weighting the Unlabeled Data

- This method weights the influence of unlabeled examples by factor *m*.
- In LU learning, the labeled set is small, but the unlabeled set is very large. So the EM's parameter estimation is almost completely determined by the unlabeled set after the first iteration.
  - This means that EM essentially performs unsupervised clustering. When the two mixture model assumptions are true, the natural clusters of the data are in correspondence with the class labels. The resulting clusters can be used as the classifier.
  - when the assumptions are not true, the clustering may not converge to mixture components corresponding to the given classes.

*Solution*:

- Reduce the effect of the problem by weighting down the unlabeled data during parameter estimation (EM iterations).

# Weighting the Unlabeled Data

- The computation of $\Pr(w_t|c_j)$ is changed to the following, where the counts of the unlabeled documents are decreased by a factor of $\mu, 0 \leq \mu \leq 1$.
- New M step:

$$\Pr(w_t \mid c_j) = \frac{\lambda + \sum_{i=1}^{|D|} \Lambda(i) N_{ti} \Pr(c_j \mid d_i)}{\lambda |V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} \Lambda(i) N_{ti} \Pr(c_j \mid d_i)}, \quad (1)$$

where

$$\Lambda(i) = \begin{cases} \mu & \text{if } d_i \in U \\ 1 & \text{if } d_i \in L. \end{cases} \quad (2)$$

- The value of $\mu$ can be chosen based on leave-one-out cross-validation accuracy on the labeled training data. The $\mu$ value that gives the best result is used.
- The prior probability also needs to be weighted.

# Finding Mixture Components

- Instead of weighting unlabeled data low, we can attack the problem head on, i.e., by finding the mixture components (sub-classes) of the class.
  - For example, the original class Sports may consist of documents from Baseball, Tennis, and Basketball, which are three mixture components (sub-classes or sub-topics) of Sports. Instead of using the original class, we try to find these components and treat each of them as a class and replace the class Sports.

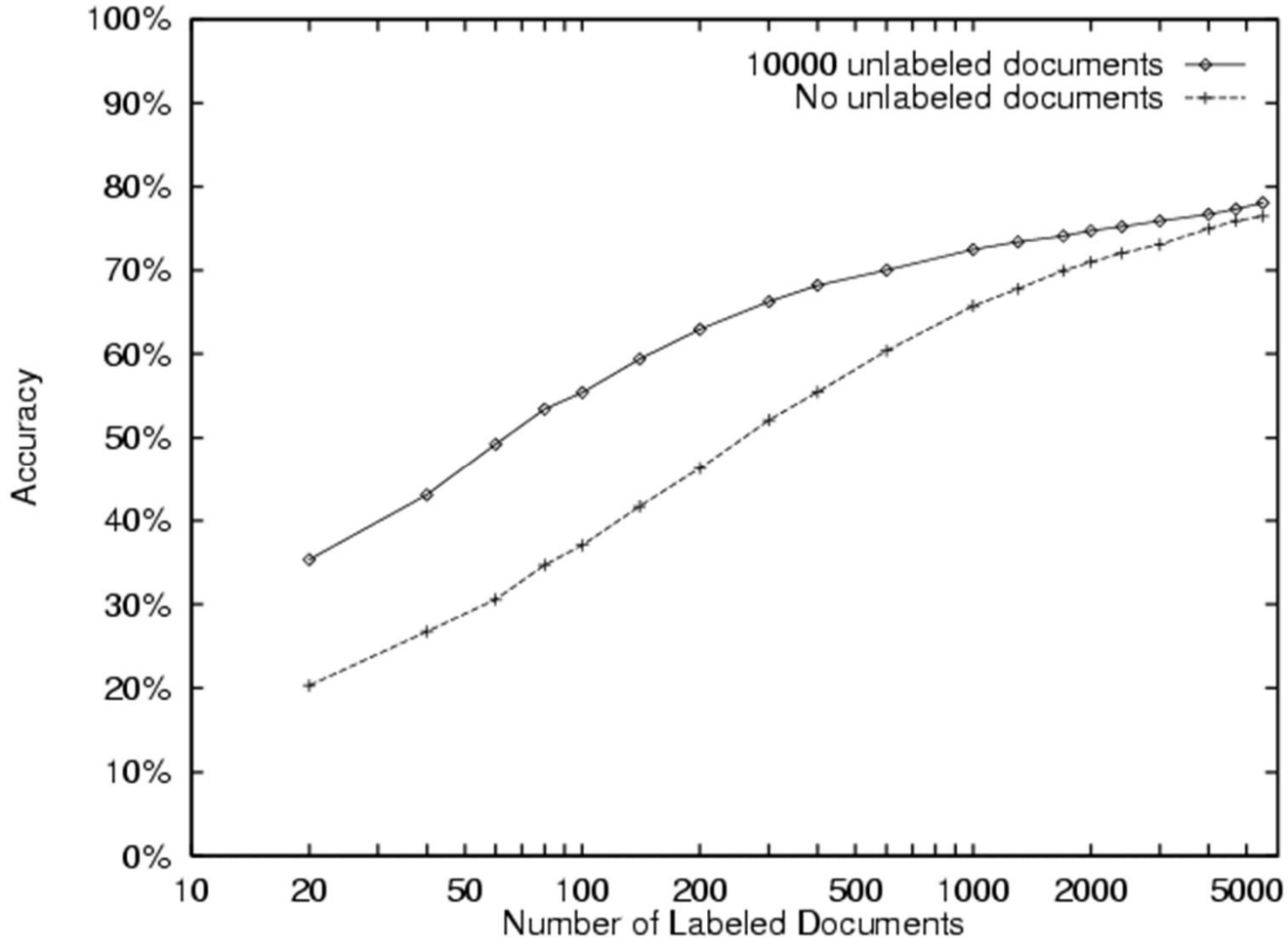
# Finding Mixture Components

- Approaches to identify different components:
  - Manually identifying different components:
    - one only needs to read the documents in the labeled set (or some sampled unlabeled documents), which is very small.
  - Automatic approaches for identifying mixture components:
    - For example, a hierarchical clustering technique was proposed in to find the mixture components (sub-classes).
    - A simple approach based on leave-one-out cross-validation on the labeled training set

# Experimental Evaluation

- Newsgroup postings
  - 20 newsgroups, 1000/group
- Web page classification
  - student, faculty, course, project
  - 4199 web pages
- Reuters newswire articles
  - 12,902 articles
  - 10 main topic categories

# 20 Newsgroups



# 20 Newsgroups