Partially Supervised Learning

M.M. Pedram
pedram@khu.ac.ir
Kharazmi University
(Fall 2011)

1

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)

2

Outline

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

Fully supervised learning (traditional classification)

Document Classification

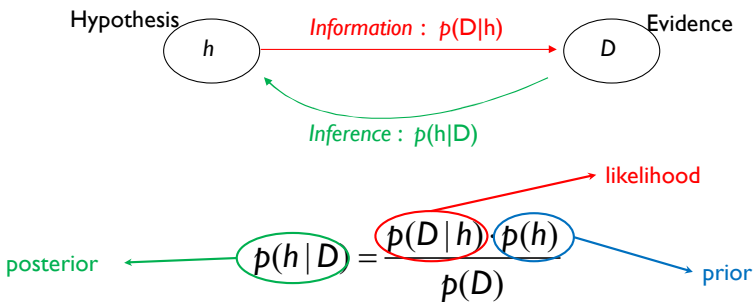
- Spam filtering, relevance rating, web page classification, ...
- f : Doc $\rightarrow$ Class

  (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
  $Pr(A_1=a_1|A_2=a_2,...,A_k=a_k, C=c_i)* Pr(A_2=a_2,...,A_k=a_k |C=c_i)$
- 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 | C=t) = 2/5 Pr(A=g | C=t) = 2/5 Pr(A=h | C=t) = 1/5
Pr(A=m | C=f) = 1/5 Pr(A=g | C=f) = 2/5 Pr(A=h | C=n) =2/5
Pr(B=b | C=t) = 1/5 Pr(B=s | C=t) = 2/5 Pr(B=q | C=t) = 2/5
Pr(B=b | C=f) = 2/5 Pr(B=s | C=f) = 1/5 Pr(B=q | C=f) = 2/5

Now we have a test example:

A = m B = q 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.

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

  - λ 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

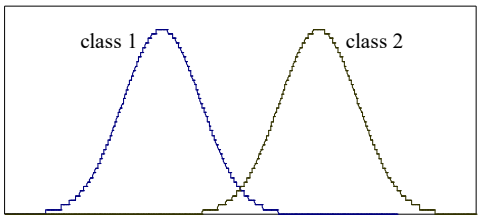
- **Generative model**: Each document is generated by *a parametric distribution* governed by *a set of hidden parameters*.
  - Training data is used to estimate these parameters.
  - The parameters are then applied to classify each test document using Bayes rule by calculating the *posterior probability* 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 θ_i) are the mean (μ_i) and the standard deviation (σ_i), i.e., $\theta_i = (\mu_i, \sigma_i)$.



Mixture model (cont …)

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

$$\Theta = \{\varphi_1, \varphi_2, \ldots, \varphi_K, \theta_1, \theta_2, \ldots, \theta_K\}$$

- 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*

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

- 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), $\varphi_j = \Pr(c_j \mid \Theta)$.
  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 $\Pr(d_i \mid c_j; \theta_j)$.

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

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.

⇒ 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 outcomes 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_t : 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 **x** = (x_1, …, $x_{no\_of\_classes}$) from a multinomial distribution with *n* trials and parameter vector

$$\theta = (\theta_1, \ldots, \theta_{no\text{-}of\text{-}classes})$$

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

$$\hat{\theta}_i = \frac{x_i + \lambda}{n + \lambda . \ no\_of\_classes}$$

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, $0 \le \lambda \le 1$

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

Parameter estimation (cont …)

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

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

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

go to

Classification

- Given a test document d_i, from the followings

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

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

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:

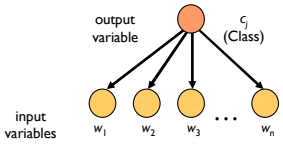
$$\textit{classifier output} = \arg\max_{c_j \in C} \Pr(c_j \mid d_i; \hat{\Theta})$$

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

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

➢ Assumption: words are conditionally independent, given class.

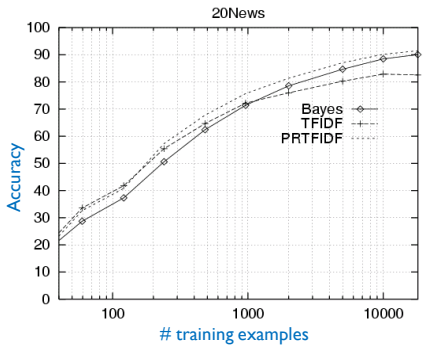
Classification

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

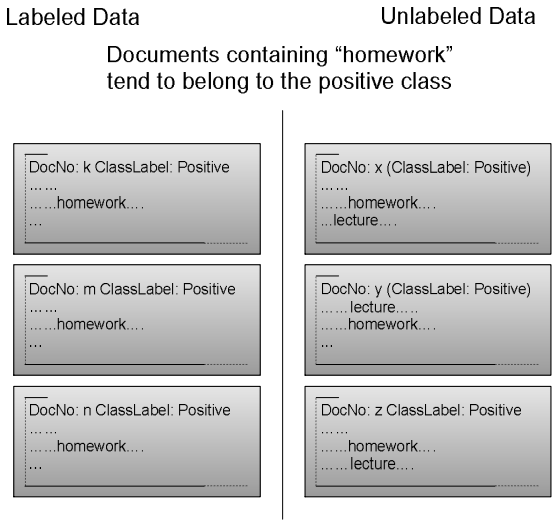
Unlabeled Data

- 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?



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 θ,
- We want θ 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 θ 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 θ and its integral / sum need not be 1.0,
- The M.L. (maximum likelihood) θ, is the one that maximizes the likelihood function (without regard for a prior on θ)

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:
  - the probability of each word given a class +
  - 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));

$$\Pr(w_t|c_j;\hat{\Theta}) = \frac{\lambda + \sum_{i=1}^{|D|} N_{it} \Pr(c_j|d_i)}{\lambda|V| + \sum_{s=1}^{|V|}\sum_{i=1}^{|D|} N_{is} \Pr(c_j|d_i)} \qquad \Pr(c_j|\hat{\Theta}) = \frac{\sum_{i=1}^{|D|} \Pr(c_j|d_i)}{|D|}$$

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));

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

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));

$$\Pr(w_t|c_j;\hat{\Theta}) = \frac{\lambda + \sum_{i=1}^{|D|} N_{it} \Pr(c_j|d_i)}{\lambda|V| + \sum_{s=1}^{|V|}\sum_{i=1}^{|D|} N_{is} \Pr(c_j|d_i)} \qquad \Pr(c_j|\hat{\Theta}) = \frac{\sum_{i=1}^{|D|} \Pr(c_j|d_i)}{|D|}$$

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 μ.
- 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 $\mu, 0 \le \mu \le 1$.
- New M step:

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

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 μ can be chosen based on leave-one-out cross-validation accuracy on the labeled training data. The μ 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*.

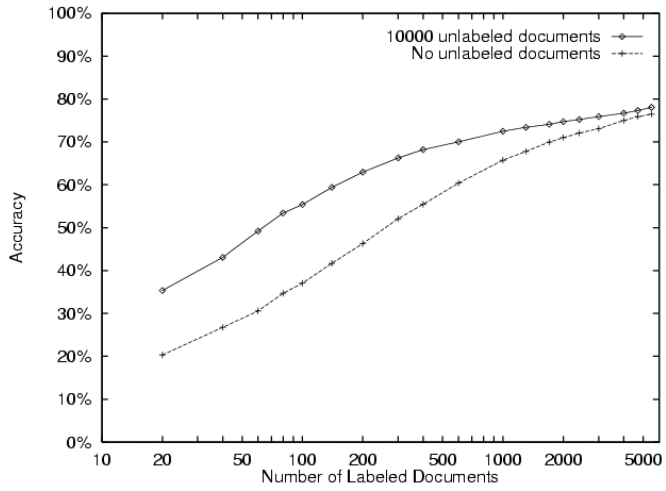
Finding Mixture Components

- 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).

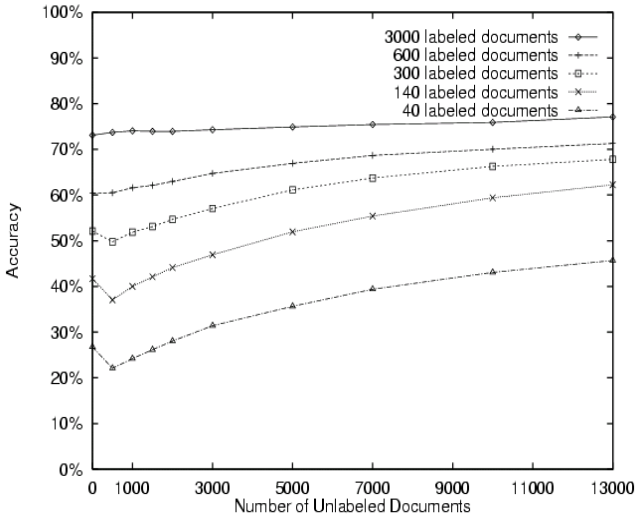
Experimental Evaluation

- 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



Co-training

- A. Blum, T. Mitchell, "Combining labeled and unlabeled data with co-training", In *Proceedings of Conference on Computational Learning Theory*, 1998.
- Again, learning with a small labeled set and a large unlabeled set.
- The attributes describing each example or instance can be partitioned into two subsets. Each of them is sufficient for learning the target function.
  - E.g., hyperlinks and page contents in Web page classification.
- Two classifiers can be learned from the same data.

Two views of an instance

- Consider the supervised learning task of *named entity* classification in natural language processing:
  - A named entity is a proper name such as "Washington State" or "Mr. Washington."
  - Each named entity has a class label depending on what it is referring to.
    - For simplicity, we assume there are only two classes: Person or Location.
  - The goal of named entity classification is to assign the correct label to each entity, for example, Location to "Washington State" and Person to "Mr. Washington."

Two views of an instance

- An instance of a named entity can be represented by two distinct sets of features.
  - The first is the set of words that make up *the named entity* itself. (in parentheses)
  - The second is the set of words in the *context* in which the named entity occurs. (Underline)

instance 1: ... headquartered in (Washington State) ...
instance 2: ... (Mr. Washington), the vice president of ...

instance	$\mathbf{x}^{(1)}$	$\mathbf{x}^{(2)}$	y
1.	Washington State	headquartered in	Location
2.	Mr. Washington	vice president	Person

Two views of an instance

- There are many other ways to express a location or person. For example:

. . . (Robert Jordan), a partner at . . .
. . . flew to (China) . . .

- These latter instances are not covered by the two labeled instances given in previous slide. Thus, a supervised learner will not be able to classify them correctly. It seems that a very large labeled training sample is necessary to cover all the variations in location or person expressions or semi-supervised learning should be applied.

Two views of an instance

- It is sufficient to have a large unlabeled training sample, which is much easier to obtain. Let us say we have the following unlabeled instances:

instance 3: . . . headquartered in (Kazakhstan) . . .
instance 4: . . . flew to (Kazakhstan) . . .
instance 5: . . . (Mr. Smith), a partner at Steptoe & Johnson . . .

It is illustrative to inspect the features of the labeled and unlabeled instances together:

Two views of an instance

instance	$\mathbf{x}^{(1)}$	$\mathbf{x}^{(2)}$	y
1.	Washington State	headquartered in	Location
2.	Mr. Washington	vice president	Person
3.	Kazakhstan	headquartered in	?
4.	Kazakhstan	flew to	?
5.	Mr. Smith	partner at	?

- One may reason about the data in the following steps:
  1. From labeled instance 1, we learn that "headquartered in" is a context that seems to indicate y =Location.
  2. If this is true, we infer that "Kazakhstan" must be a Location since it appears with the same context "headquartered in" in instance 3.
  3. Since instance 4 is also about "Kazakhstan," it follows that its context "flew to" should indicate Location.
  4. At this point, we are able to classify "China" in "*flew to* (*China*)" as a Location, even though neither "flew to" nor "China" appeared in the labeled data!
  5. Similarly, by matching "Mr. *" in instances 2 and 5, we learn that "partner at" is a context for y =Person. This allows us to classify "(Robert Jordan), a partner at" as Person, too.

Two views of an instance

- Note: We implicitly used two classifiers in turn.
  - They operate on different views of an instance: one is based on the *named entity* string itself ($x^{(1)}$), and the other is based on the *context string* ($x^{(2)}$).

Co-training Algorithm

Input: labeled data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, *unlabeled data* $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$, *a learning speed* k.
Each instance has two views $\mathbf{x}_i = [\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(2)}]$.
1. *Initially let the training sample be* $L_1 = L_2 = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$.
2. *Repeat until unlabeled data is used up:*
3. *Train a view-1 classifier* $f^{(1)}$ *from* L_1, *and a view-2 classifier* $f^{(2)}$ *from* L_2.
4. *Classify the remaining unlabeled data with* $f^{(1)}$ *and* $f^{(2)}$ *separately.*
5. *Add* $f^{(1)}$*'s top* k *most-confident predictions* $(\mathbf{x}, f^{(1)}(\mathbf{x}))$ *to* L_2.
Add $f^{(2)}$*'s top* k *most-confident predictions* $(\mathbf{x}, f^{(2)}(\mathbf{x}))$ *to* L_1.
Remove these from the unlabeled data.

- The key idea of co-training is that classifier $f^{(1)}$ adds examples to the labeled set that are used for learning $f^{(2)}$ based on the X2 view, and vice versa.

Co-training

- Recall mode:
  - At classification time, for each test example the two classifiers are applied separately and their scores are combined to decide the class.
    - For naïve Bayesian classifiers, we multiply the two probability scores, i.e.,

$$\Pr(c_j|\mathbf{x}) = \Pr(c_j|\mathbf{x}_1)\Pr(c_j|\mathbf{x}_2)$$

Co-training: Experimental Results

- begin with 12 labeled web pages (academic course)
- provide 1,000 additional unlabeled web pages
- average error: learning from labeled data 11.1%;
- average error: co-training 5.0%

	Page-base classifier	Link-based classifier	Combined classifier
Supervised training	12.9	12.4	11.1
Co-training	6.2	11.6	5.0

Co-training assumptions

1. Each view alone is sufficient to make good classifications, given enough labeled data.
2. The two views are conditionally independent given the class label.

$$P(\mathbf{x}^{(1)}|y,\mathbf{x}^{(2)}) = P(\mathbf{x}^{(1)}|y)$$
$$P(\mathbf{x}^{(2)}|y,\mathbf{x}^{(1)}) = P(\mathbf{x}^{(2)}|y)$$

- Problem with the second assumption:
  - In the case of Web page classification, this assumes that the words on a Web page are not related to the words on its incoming hyperlinks, except through the class of the Web page. This is a somewhat unrealistic assumption in practice.

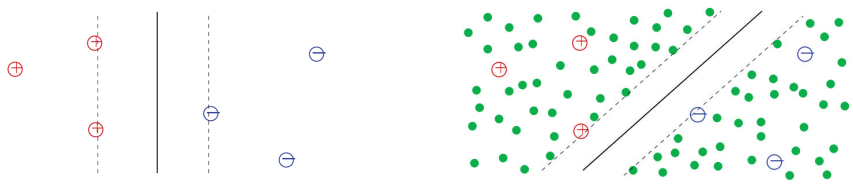
Inductive vs. Transductive

- There are two distinct goals, thus 2 types of semi-supervised learning:
  - *Inductive Semi-supervised Learning*: Training in order to predict the labels on future test data.
  - *Transductive Semi-supervised Learning*: Training in order to predict the labels on the unlabeled instances in the training sample.

- An analogy:
  - inductive semi-supervised learning is like an in-class exam, where the questions are not known in advance, and a student needs to prepare for all possible questions; in contrast, transductive learning is like a take-home exam, where the student knows the exam questions and needs not prepare beyond those.

Transductive Support Vector Machines

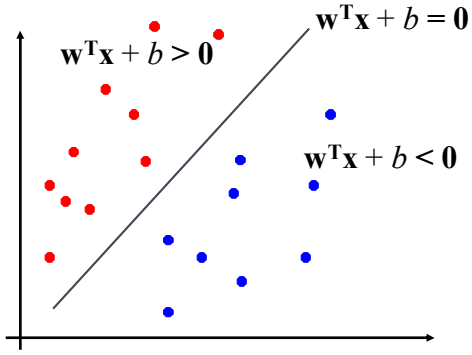
- The intuition behind Semi-Supervised Support Vector Machines (S3VMs):



- The distance from the decision boundary to a dotted line is called the *geometric margin*.

Perceptron: Linear Separators

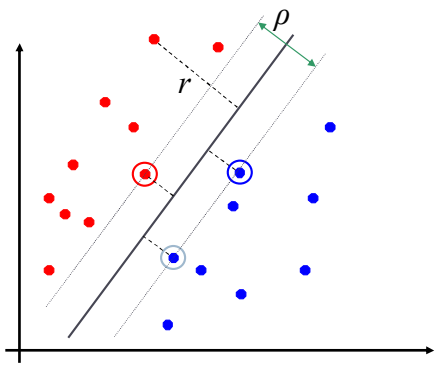
- Binary classification can be viewed as the task of separating classes in feature space:



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

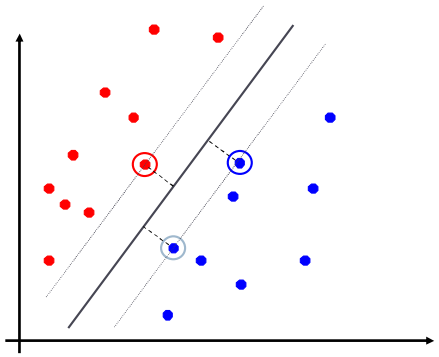
Classification Margin

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



Maximum Margin Classification

- Maximizing the margin is good.
- Implies that only support vectors matter; other training examples are ignorable.



Linear SVM Mathematically

- Let training set $\{(\mathbf{x}_i, y_i)\}_{i=1..l}, \mathbf{x}_i \in \mathbf{R}^d, y_i \in \{-1, 1\}$ be separated by a hyperplane with margin ρ. Then for each training example $(\mathbf{x}_i, y_i)$:

$$\begin{aligned} \mathbf{w}^T\mathbf{x}_i + b &\le -\rho/2 \quad \text{if } y_i = -1 \\ \mathbf{w}^T\mathbf{x}_i + b &\ge \rho/2 \quad \text{if } y_i = 1 \end{aligned} \quad \Leftrightarrow \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \ge \rho/2$$

- For every support vector $\mathbf{x}_s$ the above inequality is an equality. After rescaling $\mathbf{w}$ and b by $\rho/2$ in the equality, we obtain that distance between each $\mathbf{x}_s$ and the hyperplane is

$$r = \frac{y_s(\mathbf{w}^T\mathbf{x}_s + b)}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

- Then the margin can be expressed through (rescaled) $\mathbf{w}$ and b as:

$$\rho = 2r = \frac{2}{\|\mathbf{w}\|}$$

Linear SVMs Mathematically (cont.)

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

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

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

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

Which can be reformulated as:

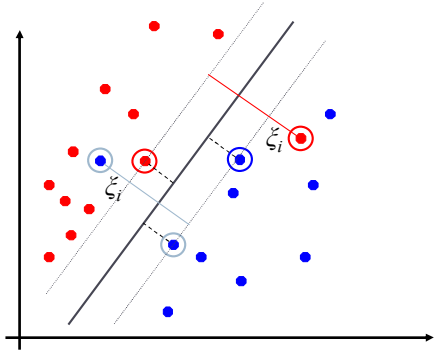
Find $\mathbf{w}$ and b such that

$\Phi(\mathbf{w}) = \|\mathbf{w}\|^2 = \mathbf{w}^T\mathbf{w}$ is minimized

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

Soft Margin Classification

- What if the training set is not linearly separable?
- *Slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called *soft.*



Soft Margin Classification

- The old formulation:

Find **w** and b such that
$\Phi(\mathbf{w}) = \mathbf{w}^T\mathbf{w}$ is minimized
and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

- Modified formulation incorporates slack variables:

Find **w** and b such that
$\Phi(\mathbf{w}) = \mathbf{w}^T\mathbf{w} + C\sum\xi_i$ *or* $\lambda\,\mathbf{w}^T\mathbf{w} + \sum\xi_i$ is minimized
and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$, $\xi_i \geq 0$

- Parameter *C* can be viewed as a way to control overfitting: it "trades off" the relative importance of maximizing the margin and fitting the training data.

Soft Margin Classification

- It is illustrative to cast ★ into a regularized risk minimization framework, as this is how we will extend it to S3VMs.

- *Model Selection Procedures*: There are a number of procedures we can use to fine-tune model complexity:
  - *cross-validation*
  - *regularization* (Breiman 1998): an augmented error function is used.

E' = error on data + λ·model complexity

Soft Margin Classification

- Consider the following optimization problem

$$\begin{aligned} \min_{\xi} \quad & \xi \\ \text{subject to} \quad & \xi \geq z \\ & \xi \geq 0 \end{aligned} \qquad \bigstar$$

It will be equivalent to evaluate the function

$$\max(z, 0)$$

- By comparing ★ to ★ , we can convert★ to an unconstrained, regularized risk minimization problem:

$$\min_{\mathbf{w},b} \sum_{i=1} \max(1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b), 0) + \lambda \|\mathbf{w}\|^2 \qquad \bigstar$$

Soft Margin Classification

- Where the first term corresponds to the (*hinge*)loss function

$$c(\mathbf{x}, y, f(\mathbf{x})) = \max(1 - y\underbrace{(\mathbf{w}^\top \mathbf{x} + b)}_{f(\mathbf{x})}, 0)$$

and the second term corresponds to the regularizer:

$$\Omega(f) = \|\mathbf{w}\|^2$$

- Now, we will consider two cases:
  - Case 1: Labeled data
  - Case 2: Unlabeled data

Soft Margin Classification

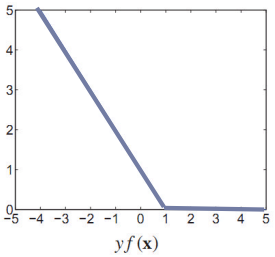
Case 1: Labeled Data

- Recall that for well-separated training instances, we have

$$y(\mathbf{w}^T\mathbf{x} + b) \geq 1$$

- Therefore, the loss function penalizes instances which are on the correct side of the decision boundary, but within the margin ($0 \leq y(\mathbf{w}^T\mathbf{x} + b) < 1$); it penalizes instances even more if they are on the wrong side of the decision boundary ($y(\mathbf{w}^T\mathbf{x} + b) < 0$).

- Hinge loss function:



Soft Margin Classification

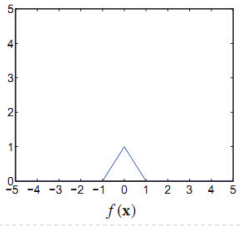
Case 2: Unlabeled data

- Here is one way to incorporate the unlabeled instance ***x*** into learning. Recall the label prediction on ***x*** is $\hat{y} = \text{sign}(f(\mathbf{x}))$. If we treat this prediction as the putative label of ***x***, then we can apply the loss function on ***x*** :

$$\begin{aligned} c(\mathbf{x}, \hat{y}, f(\mathbf{x})) &= \max(1 - \hat{y}(\mathbf{w}^\top\mathbf{x} + b), 0) \\ &= \max(1 - \text{sign}(\mathbf{w}^\top\mathbf{x} + b)(\mathbf{w}^\top\mathbf{x} + b), 0) \\ &= \max(1 - |\mathbf{w}^\top\mathbf{x} + b|, 0), \end{aligned}$$

where the fact sign(z). z = |z| was used.

- This new loss function is distinct from the hinge loss in that it does not need the real label y. It is called the *hat loss*.



Soft Margin Classification

- By the way we generate the putative label y^, the hat loss penalizes unlabeled instances.
  - Specifically, it prefers $f(\boldsymbol{x}) \geq 1$ or $f(\boldsymbol{x}) \leq -1$ (the "rim" of the hat) which are unlabeled instances outside the margin, far away from the decision boundary.
  - It penalizes unlabeled instances with $-1 < f(\boldsymbol{x}) < 1$, especially the instances within the margin with $f(\boldsymbol{x}) \approx 0$. Intuitively, they are the ones that f is uncertain about.

- We now incorporate the hat loss on the unlabeled data x_j , $j = l+1$ *to* $j = l+u$, into the SVM objective ★ to form theS3VM objective:

$$\min_{\mathbf{w},b} \sum_{i=1}^{l} \max(1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b), 0) + \lambda_1 \|\mathbf{w}\|^2 + \lambda_2 \sum_{j=l+1}^{l+u} \max(1 - |\mathbf{w}^\top \mathbf{x}_j + b|, 0)$$

Soft Margin Classification

- The S3VM objective ***prefers*** unlabeled instances to be outside the margin. Equivalently, the decision boundary would want to be in a low-density gap in the dataset, such that few unlabeled instances are close.

- Here, the regularizer involves these hat-shaped functions:

$$\Omega(f) = \lambda_1 \|\mathbf{w}\|^2 + \lambda_2 \sum_{j=l+1}^{l+u} \max(1 - |\mathbf{w}^\top \mathbf{x}_j + b|, 0)$$

Soft Margin Classification

- There is one practical consideration.
  - Empirically, it is sometimes observed that the solution to ✶ is imbalanced. That is, the majority (or even all) of the unlabeled instances are predicted in only one of the classes.
- To correct for the imbalance, one heuristic is to constrain the predicted class proportion on the unlabeled data, so that it is the same as the class proportion on the labeled data:

$$\frac{1}{u}\sum_{j=l+1}^{l+u}\hat{y}_j = \frac{1}{l}\sum_{i=1}^{l} y_i$$

$$\frac{1}{u}\sum_{j=l+1}^{l+u} f(\mathbf{x}_j) = \frac{1}{l}\sum_{i=1}^{l} y_i$$

Soft Margin Classification

Therefore

- *The complete S3VM problem with class balance constraint* is:

$$\min_{\mathbf{w},b} \sum_{i=1}^{l}\max(1-y_i(\mathbf{w}^\top\mathbf{x}_i+b),0)+\lambda_1\|\mathbf{w}\|^2+\lambda_2\sum_{j=l+1}^{l+u}\max(1-|\mathbf{w}^\top\mathbf{x}_j+b|,0)$$

$$\text{subject to}\quad \frac{1}{u}\sum_{j=l+1}^{l+u}\mathbf{w}^\top\mathbf{x}_j+b=\frac{1}{l}\sum_{i=1}^{l}y_i.$$

Soft Margin Classification

- A function g is convex, if for $\forall z_1, z_2, \forall 0 \le \lambda \le 1$,

$$g(\lambda z_1 + (1-\lambda) z_2) \le \lambda g(z_1) + (1-\lambda) g(z_2)$$

- The SVM objective ★ is a convex function of the parameters $\mathbf{w}$ and b. This can be verified by the convexity of the hinge loss, the squared norm, and the fact that the sum of convex functions is convex.

- The hat loss function is non-convex (check $z_1 = -1, z_2 = 1, \lambda = 0.5$). With the sum of a large number of hat functions, the S3VM objective is non-convex with multiple local minima.
  - A learning algorithm can get trapped in a sub-optimal local minimum, and not find the global minimum solution. The research in S3VMs has focused on how to efficiently find a near-optimum solution.

Transductive SVM

- Joachims, T., "Transductive inference for text classification using support vector machines", In Proceedings of International Conference on Machine Learning (ICML-1999), 1999.
  - Joachims used a sub-optimal iterative method that starts by learning a classifier using only the labeled data. The method then treats a subset of unlabeled instances that are most confidently labeled positive by the learned classifier as initial positive examples while the rest of the unlabeled examples are treated as initial negative examples.
  - The method then tries to improve the soft margin cost function by iteratively changing the labels of some of the instances and retraining the SVM.
  - The ratio of positive to negative instances is maintained by selecting one positively labeled instance p and one negatively labeled instance q to change in each iteration.

Transductive SVM

- It was shown that if the two instances are selected such that the slack variables $\xi_p > 0$, $\xi_q > 0$ and $\xi_p + \xi_q > 2$, the soft margin cost function will decrease at each iteration.
- Further improvements include allowing the soft margin error of unlabeled examples to be penalized differently from the soft margin error of the labeled examples and penalizing the soft margin error on the positive unlabeled examples differently from the soft margin error on the negative unlabeled examples. The penalty on the unlabeled examples is also iteratively increased from a small value to the desired value. This may improve the chances of finding a good local optimum as it may be easier to improve the cost function when the penalty is small.

Difficulties of Transductive SVM

- The main difficulty with applying transductive SVM is the computational complexity. When all the labels are observed, training SVM is a convex optimization problem that can be solved efficiently. The problem of assigning labels to unlabeled examples in such a way that the resulting margin of the classifier is maximized can no longer be solved efficiently.
- Like other methods of learning from labeled and unlabeled examples, transductive SVM can be sensitive to its assumptions.
  - When the large margin assumption is correct on the dataset, it may improve performance but when the assumption is incorrect, it can decrease performance compared to supervised learning. With a small number of labeled data, separating the Web pages according to some of the underlying topics of the Web pages may give a larger margin, resulting in less accurate Tansductive classifier.

Learning from Positive and Unlabeled Examples

PU learning

Learning from Positive & Unlabeled data

- Positive examples: One has a set of examples of a class *P*, and
- Unlabeled set: also has a set *U* of unlabeled (or mixed) examples with instances from *P* and also *negative examples* (not from *P*).
- Build a classifier: Build a classifier to classify the examples in *U* and/or future (test) data.
- Key feature of the problem: no labeled negative training data.
- We call this problem, PU-learning.

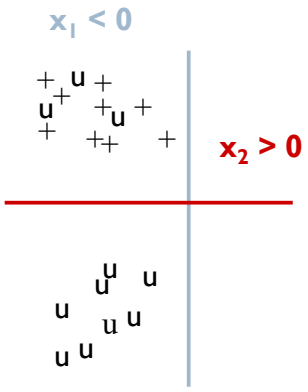
Applications of the problem

- With the growing volume of online texts available through the Web and digital libraries, one often wants to find those documents that are related to one's work or one's interest.
- For example, given a ICML proceedings,
  - find all machine learning papers from AAAI, IJCAI, KDD
  - No labeling of negative examples from each of these collections.
- Similarly, given one's bookmarks (positive documents), identify those documents that are of interest to him/her from Web sources.

Direct Marketing

- Company has database with details of its customer – positive examples, but no information on those who are not their customers, i.e., no negative examples.
- Want to find people who are similar to their customers for marketing
- Buy a database consisting of details of people, some of whom may be potential customers – unlabeled examples.

Are Unlabeled Examples Helpful?



- Function known to be either $x_1 < 0$ or $x_2 > 0$
- Which one is it?

"Not learnable" with only positive examples. However, addition of unlabeled examples makes it learnable.

Theoretical foundations

- (X, Y): X - input vector, $Y \in \{1, -1\}$ - class label.
- f : classification function
- We rewrite the **probability of error**

$$\Pr[f(X) \neq Y] = \Pr[f(X) = 1 \text{ and } Y = -1] + \Pr[f(X) = -1 \text{ and } Y = 1] \quad (1)$$

We have

$$\Pr[f(X) = 1 \text{ and } Y = -1] = \Pr[f(X) = 1] - \Pr[f(X) = 1 \text{ and } Y = 1]$$
$$= \Pr[f(X) = 1] - (\Pr[Y = 1] - \Pr[f(X) = -1 \text{ and } Y = 1])$$

Plug this into (1), we obtain

$$\Pr[f(X) \neq Y] = \Pr[f(X) = 1] - \Pr[Y = 1] + 2\Pr[f(X) = -1 | Y = 1].\Pr[Y = 1] \quad (2)$$

Theoretical foundations (cont)

- $\Pr[f(X) \neq Y] = \Pr[f(X) = 1] - \Pr[Y = 1]$ (2)
  $+ 2\Pr[f(X) = -1 | Y = 1] \Pr[Y = 1]$
- Note that $\Pr[Y = 1]$ is constant.
- If we can hold $\Pr[f(X) = -1 | Y = 1]$ small, then learning is approximately the same as minimizing $\Pr[f(X) = 1]$.

	Retrieved	Not-Retrieved	
Relevant	TP	FN	*Recall or SENS*
Not-Relevant	FP	TN	*SPEC*
	Precision	*NPV*	

- Holding $\Pr[f(X) = -1 | Y = 1]$ small while minimizing $\Pr[f(X) = 1]$ is approximately the same as
  - minimizing $\Pr_u[f(X) = 1]$
  - while holding $\Pr_P[f(X) = 1] \geq r$ (where r is recall Pr[f(X)=1|Y=1]), which is the same as $(\Pr_P[f(X) = -1] \leq 1 - r)$

  if the set of positive examples P and the set of unlabeled examples U are large enough.

Theoretical foundations (cont)

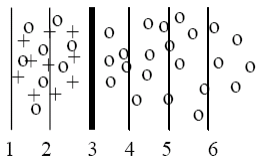
- Theorem 1 and Theorem 2 state these formally in the noiseless case and in the noisy case, in the following paper:
  - Liu, B., W. Lee, P.Yu, and X. Li, "Partially supervised classification of text documents", Proceedings of the Nineteenth International Conference on Machine Learning (ICML-2002), 8-12, July 2002, Sydney, Australia.

Put it simply

- A constrained optimization problem.
- A reasonably good generalization (learning) result can be achieved
  - If the algorithm tries to minimize the number of unlabeled examples labeled as positive
  - subject to the constraint that the fraction of errors on the positive examples is no more than 1-*r*.

An illustration

- Assume a linear classifier.



- Assume that the positive set has no error, and we want the recall to be r=%100 on the positive set.
- Lines 1 and 2 are clearly not solutions because the constraint "the fraction of errors on the positive examples must be no more than 1- r (= 0)" is violated, although the number of unlabeled examples labeled (classified) as positive is minimized by line 1.
- Lines 4, 5, and 6 are poor solutions too because the number of unlabeled examples labeled as positive is not minimized by any of them.
- Line 3 is the optimal solution.

Approaches to Build PU Classifiers

- Based on the constrained optimization idea, two kinds of approaches have been proposed to build PU classifiers:
  - *the two-step approach,*
  - *the direct approach.*

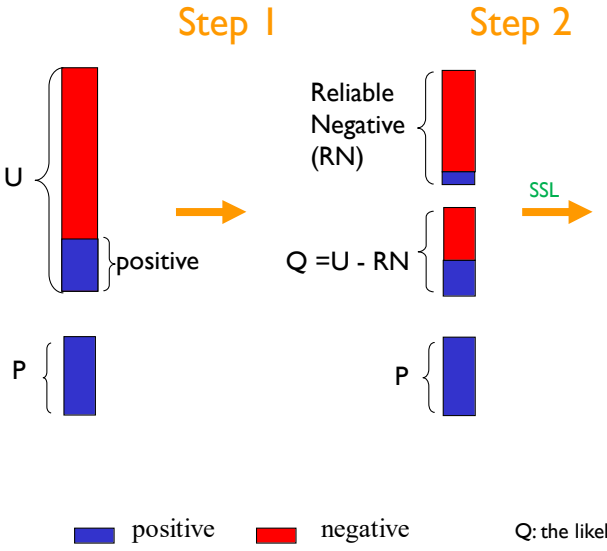
- In the actual learning algorithms, the user may not need to specify a desired recall level *r* on the positive set because some of these algorithms have their evaluation methods that can automatically determine whether a good solution has been found.

Existing 2-step Approach

- Step 1: Identifying a set of reliable negative (RN)documents from the unlabeled set.
  - S-EM [Liu et al, 2002] uses a Spy technique,
  - PEBL [Yu et al, 2002] uses a 1-DNF technique
  - Roc-SVM [Li & Liu, 2003] uses the Rocchio algorithm.
  - ...

- Step 2: Building a sequence of classifiers by iteratively applying a classification algorithm and then selecting a good classifier, i.e. Building a classifier using P, RN and U – RN. (This step may apply an existing learning algorithm once or iteratively depending on the quality and the size of the RN set.)
  - S-EM uses the Expectation Maximization (EM) algorithm, with an error based classifier selection mechanism
  - PEBL uses SVM, and gives the classifier at convergence. I.e., no classifier selection.
  - Roc-SVM uses SVM with a heuristic method for selecting the final classifier.

Existing 2-step Approach



109

Existing 2-step Approach

- In step 2:
  - the algorithm iteratively improves the results by adding more documents to RN until a convergence criterion is met. The process is trying to minimize the number of unlabeled examples labeled positive.

110

Step 1: The Spy technique

- Sample a certain % of positive examples and put them into unlabeled set to act as "spies".
- Run a classification algorithm assuming all unlabeled examples are negative,
  - we will know the behavior of those actual positive examples in the unlabeled set through the "spies".
- We can then extract reliable negative examples from the unlabeled set more accurately.

Step 1: The Spy technique

Algorithm Spy(P, U)

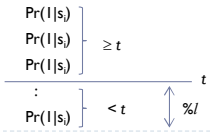
1. $RN \leftarrow \emptyset$;
2. $S \leftarrow Sample(P, s\%)$;
3. $Us \leftarrow U \cup S$;
4. $Ps \leftarrow P - S$;
5. Assign each document in Ps the class label 1;
6. Assign each document in Us the class label –1;
7. NB(Us, Ps); // This produces a NB classifier.
8. Classify each document in Us using the NB classifier;
9. Determine a probability threshold t using S; *See next slide*
10. **for** each document $d \in Us$ **do**
11. **if** its probability $\Pr(1|d) < t$ **then**
12. $RN \leftarrow RN \cup \{d\}$;
13. **endif**
14. **endfor**

Step 1: The Spy technique

- How to determine t using spies (line 9)
  - Let
    - $S = \{s_1, s_2, \ldots, s_k\}$: The set of spies,
    - $Pr(1|s_i)$: the probabilistic labels assigned to each s_i.
  - In a noiseless case :
    - The minimum probability in S as the threshold value t, i.e., $t = \min\{Pr(1|s_1), Pr(1|s_2), \ldots, Pr(1|s_k)\}$, which means that we want to retrieve all spy documents.

Step 1: The Spy technique

- How to determine t using spies (line 9)
  - Let
    - $S = \{s_1, s_2, \ldots, s_k\}$: The set of spies,
    - $Pr(1|s_i)$: the probabilistic labels assigned to each s_i.
  - Cases with outliers and noise :
    - Using the minimum probability is unreliable. The reason is that the posterior probability $Pr(1|s_i)$ of an outlier document s_i in S could be 0 or smaller than most (or even all) actual negative documents. However, we do not know the noise level of the data. To be safe, the S-EM system uses a large noise level l = 15% as the default. The final classification result is not very sensitive to l as long it is not too small.
    - To determine t, we first sort the documents in S according to their $Pr(1|s_i)$ values. We then use the selected noise level l to decide t: we select t such that l percent of documents in S have probability less than t. Hence, t is not a fixed value. (The actual parameter is in fact l.)



Step 1: The Spy technique

Note

- The reliable negative set RN can also be found through multiple iterations. That is, we run the spy algorithm multiple times. Each time a new random set of spies S is selected from P and a different set of reliable negative documents is obtained, denoted by RN_i. The final set of reliable negative documents is the intersection of all RN_i.

Step 1: Other methods

- 1-DNF method:
  - Find the set of words W that occur in the positive documents more frequently than in the unlabeled set.
  - Extract those documents from unlabeled set that do not contain any word in W. These documents form the reliable negative documents.
- Rocchio method from information retrieval.
- Naïve Bayesian method.

Step 2: Running EM or SVM iteratively

There are two approaches for this step:

1. Running a classification algorithm iteratively
   - Run EM using P, RN and Q until it converges, or
   - Run SVM iteratively using P, RN and Q until this no document from Q can be classified as negative. RN and Q are updated in each iteration, or
   - …
2. Classifier selection.
   - In general, each iteration of the algorithm gives a classifier that may potentially be a better classifier than the classifier produced at convergence. This is true for both EM and SVM.
     - The main problem with EM is that classes and topics may not have one-to-one correspondence. This is the same problem as in LU learning.
     - SVM may also produce poor classifiers at the convergence because SVM is sensitive to noise.

Step 2: Running EM or SVM iteratively

- If the RN set is not chosen well or in an iteration some positive documents are classified as negative, then the subsequent iterations may produce very poor results. In such cases, it is often better to stop at an earlier iteration. One simple method is to apply the theory directly. That is, each classifier is applied to classify a positive validation set, Pv. If many documents from Pv (e.g., > 5%) are classified as negative, the algorithm should stop (that is a recall of 95%). If the set P is small, the method can also be applied to P directly.

Do they follow the theory?

- Yes, heuristic methods because
  - Step 1 tries to find some initial reliable negative examples from the unlabeled set.
  - Step 2 tried to identify more and more negative examples iteratively.
- The two steps together form an iterative strategy of increasing the number of unlabeled examples that are classified as negative while maintaining the positive examples correctly classified.

Can SVM be applied DIRECTLY?

- Can we use SVM to directly deal with the problem of learning with positive and unlabeled examples, without using two steps?
- Yes, with a little re-formulation.

Support Vector Machines

- Support vector machines (SVM) are linear functions of the form $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$, where $\boldsymbol{w}$ is the weight vector and $\boldsymbol{x}$ is the input vector.
- Let the set of training examples be $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$, where $\boldsymbol{x}_i$ is an input vector and y_i is its class label, $y_i \in \{1, -1\}$.
- To find the linear function:

Minimize: $\frac{1}{2} w^T w$

Subject to: $y_i(w^T x_i + b) \geq 1, \quad i = 1, 2, \ldots, n$

Soft margin SVM

- This approach modifies the SVM formulation slightly so that it is suitable for PU learning.
- To deal with cases where there may be no separating hyperplane due to noisy labels of both positive and negative training examples, the soft margin SVM is proposed:

Minimize: $\frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i$

Subject to: $y_i(w^T x_i + b) \geq 1 - \xi_i, \quad i = 1, 2, \ldots, n$

where $C \geq 0$ is a parameter that controls the amount of training errors allowed.

Direct Method:Biased SVM (noiseless case)

- Assume that the first *k*-1 examples are positive examples (labeled 1), while the rest are unlabeled examples, which we label negative (-1).

Minimize: $\frac{1}{2}w^T w + C\sum_{i=k}^{n}\xi_i$

Subject to: $w^T x_i + b \geq 1, \quad i = 1, 2, \ldots, k-1$

$-1(w^T x_i + b) \geq 1 - \xi_i, \quad i = k, k+1, \ldots, n$

$\xi_i \geq 0, \quad i = k, k+1 \ldots, n$

Direct Method: Biased SVM (noisy case)

- If we also allow positive set to have some noisy negative examples, then we have:

Minimize: $\frac{1}{2}w^T w + C_+\sum_{i=1}^{k-1}\xi_i + C_-\sum_{i=k}^{n}\xi_i$

Subject to: $y_i(w^T x_i + b) \geq 1 - \xi_i, \quad i = 1, 2 \ldots, n$

$\xi_i \geq 0, i = 1, 2, \ldots, n.$

- This turns out to be the same as the asymmetric cost SVM for dealing with unbalanced data. Of course, we have a different motivation.

Estimating Performance

- We need to estimate the performance in order to select the parameters.

	Retrieved	Not-Retrieved	
Relevant	TP	FN	*Recall* or *SENS*
Not-Relevant	FP	TN	*SPEC*
	Precision	*NPV*	

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

Note:

- Greater precision decreases recall and greater recall leads to decreased precision.

Estimating Performance

- There is a trade-off between precision and recall.
- Since learning from positive and negative examples often arise in retrieval situations, we use F score as the classification performance measure

$$F = \frac{2}{\frac{1}{recall} + \frac{1}{precision}} = \frac{2p.r}{p + r}$$

(p: precision, r: recall).

- To get a high F score, both precision and recall have to be high.
- However, without labeled negative examples, we do not know how to estimate the F score.

A performance criterion

- Performance criteria $pr/Pr[Y=1]$: It can be estimated directly from the validation set as $r^2/Pr[f(X) = 1]$
  - Recall $r = Pr[f(X)=1|Y=1]$
  - Precision $p = Pr[Y=1|f(X)=1]$

To see this

$$\underbrace{Pr[f(X)=1|Y=1]}_{r}\ Pr[Y=1] = \underbrace{Pr[Y=1|f(X)=1]}_{p}\ Pr[f(X)=1]$$

$$\Leftrightarrow \quad \frac{r}{Pr[f(X)=1]} = \frac{p}{Pr[Y=1]}$$

//both side times r

- $pr/Pr[Y=1]$ behaves similar to the F-score (= $2pr / (p+r)$), i.e.
  - It is large when both p and r are large and is small when either p or r is small.

A performance criterion (cont …)

- $r^2/Pr[f(X) = 1]$
- r can be estimated from positive examples in the validation set.
- $Pr[f(X) = 1]$ can be obtained using the whole validation set.
- This criterion actually reflects the theory very well, i.e.
  - The quantity is large when r is large and $Pr(f(X) = 1)$ is small, which means that the number of unlabeled examples labeled as positive should be small.