COS 424: Interacting with Data

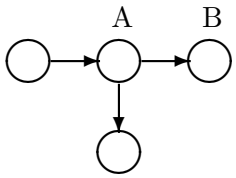
Lecturer: Dave Blei
Scribe: Andrew Ferguson

Lecture #11
March 13, 2007

1 Graphical Models Wrap-up

We began the lecture with some final words on graphical models. Choosing a graphical model is akin to choosing a probability model for your data or choosing an algorithm. Each model has advantages and disadvantages that may make it more or less suitable for modeling your data. A graphical model is a representation of a probability model, so it also carries that model's plusses and minuses.

Graphical models also come with assumptions about (in)dependency, which you should be careful of since they might not hold for your data. Use the Bayes' Ball algorithm to reveal these dependencies. Independencies in a graphical model do not have to be explicit, either. For example, in this model:



the nodes labeled "A" and "B" are not required to be dependent.

2 Text Classification

Today, we examined the problem of classifying text documents (such as e-mail messages) into some number of classes.

First off, we introduced the multinomial model to probabilistically describe the text documents. Each document is described by a vector $\boldsymbol{w}$ which is formed from a sequence of words w_n, where w_n is the n^{th} word in $\boldsymbol{w}$. Each word w_n is taken from a vocabulary of length V, that is, $w_n \; \epsilon \; \{1, \ldots, V\}$. You should think of w_n as a vector of length V with zeroes everywhere, except for a one in the position of the word it represents – w_n is a multivariate indicator random variable.

We then introduced $\boldsymbol{\beta}$, a distribution over the vocabulary. $\boldsymbol{\beta}$ has the following properties:

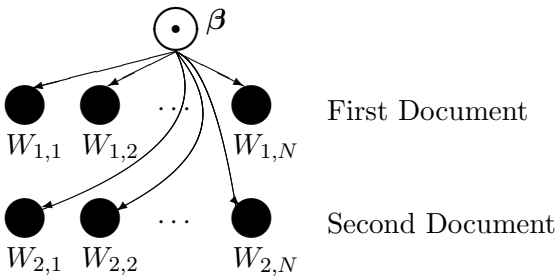
- $\boldsymbol{\beta}$ is a vector of length V
- β_i is the probability of the i^{th} word in the vocabulary occurring
- $\beta_i > 0 \; \forall \; i \; \epsilon \; \{1, \ldots, v\}$
- $\sum_{v=1}^{V} \beta_v = 1$
- The space of all possible $\boldsymbol{\beta}$'s is called the **simplex**

The words are drawn from $\boldsymbol{\beta}$, as described by the following relationship:

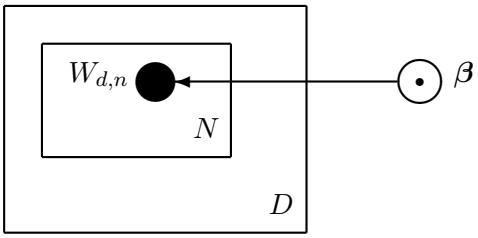
$$\mathbb{P}(w_n|\boldsymbol{\beta}) = \prod_{v=1}^{V} \beta_v^{w_n^v} = \beta_{w_n}$$

where w_n^v is the v^{th} component of the vector w_n and β_{w_n} is the component of $\boldsymbol{\beta}$ associated with the word w_n.

The standing assumption in this model is that each word in the data set is drawn from the same distribution. We can show this with the following graphical model of a collection of documents:



which can be represented more concisely using the plate notation, where there are D documents of length N:



Finally, we derive the maximum likelihood estimator for $\boldsymbol{\beta}$. First, we describe our length D data set in terms of $\boldsymbol{\beta}$:

$$\begin{aligned}\mathbb{P}(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_D|\boldsymbol{\beta}) &= \prod_{d=1}^{D}\prod_{n=1}^{N} \underbrace{\mathbb{P}(w_{d,n}|\boldsymbol{\beta})} \\ &= \prod_{d=1}^{D}\prod_{n=1}^{N} \overbrace{\prod_{v=1}^{V} \beta_v^{w_{d,n}^v}}\end{aligned}$$

where $w_{d,n}$ is the n^{th} word in the d^{th} document of the data set. This relationship answers the question "What is the probability of a data set given $\boldsymbol{\beta}$?". Then, we maximize the

logarithm of the probability to obtain the MLE in the usual manner.

$$\begin{aligned}
\mathscr{L}(\boldsymbol{\beta}, \mathscr{D}) &= \sum_{d=1}^{D}\sum_{n=1}^{N}\sum_{v=1}^{V} w_{d,n}^{v} \log \beta_v \qquad (1)\\
&= \sum_{v=1}^{V} \log \beta_v \underbrace{\sum_{d=1}^{D}\sum_{n=1}^{N} w_{d,n}^{v}}_{k_v}\\
&= \sum_{v=1}^{V} k_v \log \beta_v \qquad (2)
\end{aligned}$$

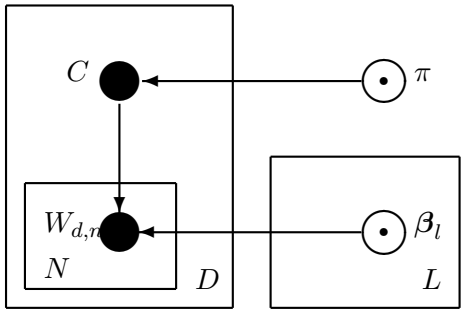
where k_v counts the number of times word v appears in the whole collection of documents. k_v is a **sufficient statistic** for our distribution – it contains all that we need to know to compute the log-likelihood. Notice in equation (1) that β_v does not depend on D or N. From equation (2) we find that the maximum likelihood estimator for $\boldsymbol{\beta}$ is:

$$\widehat{\beta_v} = \frac{k_v}{N \cdot D}$$

where $N \cdot D$ is the total number of words that we saw in the data set. For example, if the word "mortgage" appeared 15 times in our data set, which consisted of 300 words, then $\widehat{\beta}_{mortgage} = \frac{15}{300} = \frac{1}{20}$.

3 The Naive Bayes Model

The Naive Bayes Model for text classification can be described with the following graphical model:



where C represents the classes, π is some distribution over the classes (e.g., 70% spam, 30% ham), L is the number of classes, and for each class l we have a corresponding distribution $\boldsymbol{\beta}_l$ over the words. From the left-hand side of the diagram, we see that each document belongs to a class.

This graphical model encodes a generative process. It is a description of where the document came from; how it was created. In the model, some emails are randomly generated as "spam" or "ham", the two classes of emails. The graphical model doesn't tell us everything, however. We need to know that the $\boldsymbol{\beta}$ distribution used depends on the class chosen.

Using our model, we want to classify a randomly generated document. We are given the parameters π, $\boldsymbol{\beta}$, and the document itself, $\boldsymbol{w}_d$. This classification is given by the posterior

distribution over classes:

$$\begin{aligned}
\mathbb{P}(c|\boldsymbol{w}_d, \pi, \boldsymbol{\beta}) &= \frac{\mathbb{P}(c, \boldsymbol{w}_d|\pi, \boldsymbol{\beta})}{\mathbb{P}(\boldsymbol{w}_d|\pi, \boldsymbol{\beta})} && (3)\\
&= \frac{\mathbb{P}(c|\pi)\prod_{n=1}^{N}\mathbb{P}(w_{d,n}|c, \boldsymbol{\beta})}{\mathbb{P}(\boldsymbol{w}_d|\pi, \boldsymbol{\beta})} \\
&= \frac{\pi_c \prod_{n=1}^{N}\beta_{c,w_{d,n}}}{\mathbb{P}(\boldsymbol{w}_d|\pi, \boldsymbol{\beta})} && (4)
\end{aligned}$$

where in equation (3) we factor the numerator using the graphical model. The procedure classifies the document as from whichever class maximizes this posterior distribution. Mathematically, this is written as $\arg\max_c \mathbb{P}(c|\boldsymbol{w}_d, \pi, \boldsymbol{\beta}_c)$, where $\boldsymbol{\beta}_c$ is the probability distribution $\boldsymbol{\beta}$ for class c. For implementation, it is important to note that the denominator in equation (4) is the same across classes, and therefore one does not have to compute it to maximize the overall fraction. This procedure of maximizing the posterior distribution is known as **inference**.

But where do the distributions π and $\boldsymbol{\beta}_{1:L}$ come from? We estimate them from training data using the maximum likelihood estimators. The training data consists of a set of D documents labeled with their classes, which we will write as $\mathscr{D} = \{\boldsymbol{w}_d, c_d\}_{d=1}^{D}$, where c_d is a multinomial indicator variable, with a one in the position representing the class of document d and zeroes in the other positions. To compute the MLEs, we factor the joint distribution according to the graphical model, and follow the procedure used above.

$$\begin{aligned}
\mathbb{P}(\mathscr{D}|\pi, \boldsymbol{\beta}) &= \prod_{d=1}^{D}\mathbb{P}(c_d|\pi)\prod_{n=1}^{N}\mathbb{P}(w_{d,n}|\boldsymbol{\beta}_{c_d}) \\
&= \prod_{d=1}^{D}\prod_{l=1}^{L}\pi_l^{c_d^l}\prod_{l=1}^{L}\underbrace{\left(\prod_{n=1}^{N}\prod_{v=1}^{V}\beta_{l,v}^{w_{d,n}^v}\right)^{c_d^l}}_{\text{prob. of document given class } l} \\
\mathscr{L}(\pi, \boldsymbol{\beta}; \mathscr{D}) &= \sum_{d=1}^{D}\sum_{l=1}^{L}c_d^l \log \pi_l + \sum_{d=1}^{D}\sum_{l=1}^{L}c_d^l\left(\sum_{n=1}^{N}\sum_{v=1}^{V}w_{d,n}^v \log \beta_{l,v}\right)
\end{aligned}$$

Notice that raising to the power c_d^l serves to select only the class the document d is actually assigned to, since $x^0 = 1$ and $x^1 = x$.

The function $\mathscr{L}(\pi, \boldsymbol{\beta}; \mathscr{D})$ is what we wish to maximize with respect to π and $\boldsymbol{\beta}$ to obtain the maximum likelihood estimators. We note that only the first half of the sum above depends on π, $\sum_{d=1}^{D}\sum_{l=1}^{L}c_d^l \log \pi_l$. We have encountered this sort of expression when using indicator random variables many times before, that is, $\widehat{\pi}_{spam} = \frac{\text{\# of spam emails}}{\text{total \# of emails}}$. In general we have,

$$\widehat{\pi}_l = \frac{\text{\# of documents of class } l}{\text{total \# of documents}}$$

Similarly, since $\boldsymbol{\beta}_l$ is a multinomial model only of class l, we can conclude that

$$\widehat{\beta}_{l,v} = \frac{\text{\# of times we see word } v \text{ in documents of class } l}{\text{total \# of words in documents of class } l}$$

4 Naive Bayes Classification in Practice

An example of the Naive Bayes Model in practice was presented. Ten-thousand emails taken from the Enron subpoena documents were used to train the classifier to distinguish between `spam` and `ham`. The classifier was then explored using a testing set of 1,000 emails. The performance results of this simple classifier we remarkable and it was quite successful. Several important lessons about the Naive Bayes Model can be drawn from it.

The first lesson is that non-trivial words, that is, words that are present almost exclusively in one class or another, will have the greatest effect on classification. In this example, if we look at a quantity proportional to the log probability of a message being `spam`:

$$\log \mathbb{P}(\texttt{spam}|\pi) + \log \mathbb{P}(w_1|\beta_{\texttt{spam}}) + \cdots + \log \mathbb{P}(w_N|\beta_{\texttt{spam}})$$

and the same quantity of the same message being `ham`:

$$\log \mathbb{P}(\texttt{ham}|\pi) + \log \mathbb{P}(w_1|\beta_{\texttt{ham}}) + \cdots + \log \mathbb{P}(w_N|\beta_{\texttt{ham}})$$

Then any words where the difference $\log \mathbb{P}(w_i|\beta_{\texttt{spam}}) - \log \mathbb{P}(w_i|\beta_{\texttt{ham}})$ is relatively large will provide a strong indication that the document is `spam` or `ham`.

Secondly, we encountered several problems or short-comings of the Naive Bayes Model. In this model, words are treated independently of each other, while in reality, the choice of words is almost certainly dependent on the previous words. An even more serious problem is that words only seen in `spam` messages or `ham` messages are not handled very well. Such words can cause a zero to appear in one of the terms of a product such as $\mathbb{P}(\texttt{spam}|\boldsymbol{w}_d)$, which drives the whole product to zero. The solution to this problem is called **smoothing**.

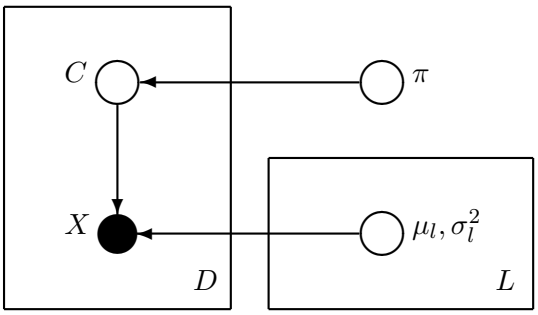
In smoothing, we add a small number, α, to each estimate of $\widehat{\beta}_{l,v}$. That is,

$$\widehat{\beta}_{l,v} = \frac{\#\text{ of times we see word } v \text{ in documents of class } l + \alpha}{\text{total } \#\text{ of words in documents of class } l + V \cdot \alpha}$$

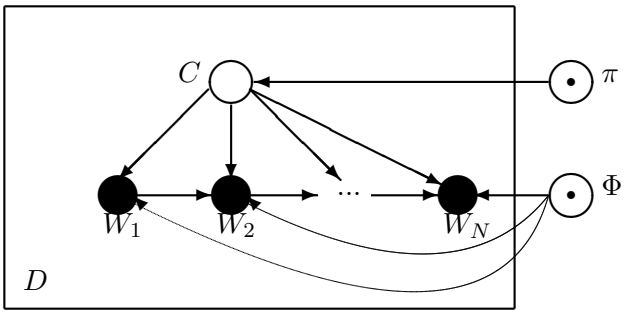
This prevents any of the maximum likelihood estimators from being zero, even if a word is unique to a particular class. The choice of α is up to the implementor, and changing the value generally affects the testing error of the classifier. Setting $\alpha = 1$ is known as Laplace Smoothing; $\alpha = 0.5$ is using Jeffrey's Prior. On the Enron test set, smoothing radically improved the performance of the Naive Bayes classifier to well over 90%. Finally, we note that on the test set, we generally ignore words that were not seen in the training set.

5 Extensions to the Naive Bayes Model

If we go back to the posterior distribution over classes given in equation (4), we see that it consists of two principal components: π_c, the probability of class c and $\prod_{n=1}^{N} \beta_{c,w_{d,n}}$, which is the probability of the words under the model. Here, we could substitute an alternate model, which we will generically describe as $\mathbb{P}(X|c)$. For example, we can use a Gaussian model:



where μ_l and σ_l^2 are the standard Gaussian MLE's. This is also known as Linear Discriminate Analysis. Another option is to work with a sequence of words, such as in this model:



where Φ is the distribution over words conditioned on the previous word.