COMP5318 Knowledge Discovery and Data Mining Assignment 2

Enoch Lau
SID 200415765

18 June 2007

Abstract

This report considers a paper [1] that combines the Expectation-Maximisation (EM) algorithm and a naive Bayes classifier in a bid to achieve better classification by means of using labelled as well as unlabelled data. In this report, we consider the problem that is addressed by the paper, describe the naive Bayes and EM frameworks, explain mathematically the combination of the two, describe their evaluation and related works, and consider some weaknesses to their approach.

Contents

1 The Problem

The general problem that is addressed by this paper is the automatic classification of text documents, which the paper says is important because of the massive volume of online text. A typical approach to this problem consists of obtaining a large sample of manually labelled documents, which can be used to train a machine learner that classifies unlabelled documents with statistical methods, which, for example, may include the word frequencies in a document. However, the key point here is that we need a large number of documents to train an accurate classifier. This is problematic not only because it is potentially very expensive and time-consuming, in a practical system, many users will not want to label, by hand, such a large number of documents to achieve a personalised classifier.

This article proposes to solve this problem by incorporating unlabelled data as well as labelled data into the training process. Especially with online documents, the collection of vast amounts of unlabelled data is inexpensive and can be done easily, because the problem with existing systems is the labelling of data, not the collection. Now, unlabelled data by itself is not useful, and is unable to achieve results better than just guessing, because there is no information about the class label for it to work with. Let us consider the example of data generated by two Gaussians. With a large amount of data available, we should be able to recover the parameters of the two Gaussians perfectly; however, we do not know which of the class labels correspond to which Gaussian. This is where the labelled data comes into play; if we provide enough examples, we will be able to work out which class belongs to which Gaussian. More generally, using unlabelled data provides information about the joint probability over features other than the class label. The example cited in the paper makes this clear. Suppose that we only know that the word "homework" is indicative of a particular class; if we from the unlabelled data, we can see that "homework" and "lecture" occur together frequently, "lecture" may also prove to be a useful indicator for that particular class.

The solution to the problem thus posed is the following. We first learn a classifier using only the labelled documents, as we would normally. The classifier chosen is the naive Bayes classifier, which is commonly used in text classification. We then use the Expectation-Maximisation (EM) algorithm in an iterative fashion; the EM algorithm is able to discover the parameters in situations where we have incomplete data, which in this case, consists of the missing class labels. After training the classifier with the labelled documents, we then use the classifier to calculate the expectation of the missing class labels on all of the unlabelled documents. Then, with the originally labelled data, and the unlabelled data with estimated class labels, we then create a new (naive Bayes) classifier. Using this new classifier, we reclassify all of the unlabelled documents again, and the cycle continues.

However, there are some problems with this approach from the start. It makes several important assumptions that are usually violated in real-world text data. Firstly, we assume that the data is generated by means of a mixture model; that is, documents are generated by picking a mixture component from the mixture model randomly but possibly in a weighted manner, and then we generate a document according to that component's parameters. This typically does not represent the process by which real-world text is generated. Secondly, it assumes that there is a one-to-one correspondence between mixture components and classes. We can see an example of where this fails in the Reuters newswire articles used in the evaluation; because we consider binary classifiers, the negative class consists of a large number of different classes all put together as one class. The paper does note, however, that real world data typically violates the naive Bayes classifier's assumptions, but still, the classifier can perform reasonably well.

The paper proposes two solutions to the problems caused by possibly incorrect assumptions. The first solution to to reduce the negative effect of the violated assumptions, by introducing a scaling factor that reduces the contribution of the unlabelled data to the parameter estimation in the EM algorithm. Secondly, the paper addresses the second of the problems directly, by allowing multiple mixture components for each class that we seek to classify. These two extensions are evaluated, in addition to the basic algorithm.

2 Naive Bayes Algorithm

2.1 Generative Model

A naive Bayes classifier works by estimating a model from the given data, and then calculating the probability that a document was assigned to each class. The class that has the highest probability of generating a particular document becomes the class label for that document.

We begin by defining our notation and the naive Bayes model, which we will keep consistent with the paper. A document d_i is an ordered list of word events $\langle w_{d_{i,1}}, w_{d_{i,2}}, \ldots, w_{d_{i,|d|}} \rangle$. Words come from a fixed vocabulary, $V = \langle w_1, w_2, \ldots, w_{|V|}$. We make two initial assumptions, namely, that the data is produced by a mixture model, and that there is a one-to-one correspondence between mixture components and classes. Hence, we can denote by c_j, the jth mixture component as well as the jth class, and let the mixture of components be called $C = \{c_1, \ldots, c_{|C|}\}$. Let the class label for document d_i be y_i. Each component has some parameters, and all the parameters from all the components taken together makes up θ.

Thus, a document is generated according to a probability distribution, which is parameterised by θ. We first select one of the components to generate this document, according to the class prior probabilities, $P(c_j|\theta)$, that is, the probability of selecting component c_j given the parameters θ. Once we have the component, we generate the document according to the component's parameters; this has a distribution of $P(d_i|c_j;\theta)$. Thus, the probability that a document d_i was generated by c_j, according to this model, is $P(c_j|\theta)P(d_i|c_j;\theta)$. Using the law of total probability, we sum over all classes/components to get:

$$P(d_i|\theta) = \sum_{j=1}^{|C|} P(c_j|\theta)P(d_i|c_j;\theta) \tag{1}$$

We then make some further assumptions: namely, the document length $|d_i|$ is generated independently of the component, each word is generated independently of the length, the words of a document are generated independently of context and the probability of a word is independent of its position within the document. The first two assumptions imply that:

$$P(d_i|c_j;\theta) = P(|d_i|) \prod_{k=1}^{|d_i|} P(w_{d_{i,k}}|c_j;\theta;w_{d_{i,q}}, q<k) \tag{2}$$

where $P(w_{d_{i,k}}|c_j;\theta;w_{d_{i,q}}, q<k)$ is the probability of a particular word being generated at position k of the document, given the class, the parameters of the model and the words that have been generated before position k.

We can simplify that probability using the last two assumptions, which will state that that probability is simply equal to the probability of that word appearing somewhere in a document for a given class:

$$P(w_{d_{i,k}}|c_j;\theta;w_{d_{i,q}}, q<k) = P(w_{d_{i,k}}|c_j;\theta) \tag{3}$$

Combining equations (2) and (3), we obtain the probability of a document given a class:

$$P(d_i|c_j;\theta) = P(|d_i|) \prod_{k=1}^{|d_i|} P(w_{d_{i,k}}|c_j;\theta) \tag{4}$$

Let us examine the parameters in θ. From equation (4), we can see that parameters of the model, which the naive Bayes algorithm needs to discover, are the collection of word probabilities and the collection of class prior probabilities. Let $\theta_{w_t|c_j} = P(w_t|c_j;\theta)$ be the probability of a word $w_t \in V$ given the class, and $\sum_{w_t} P(w_t|c_j;\theta) = 1$ (the probability over all words for a particular class must sum to 1, by the law of total probability). In addition, let the class prior probabilities be $\theta_{c_j} = P(c_j;theta)$. Therefore, $\theta = \{\theta_{w_t|c_j} : w_t \in V, c_j \in$

$C;\theta_{c_j} : c_j \in C\}$. We do not parameterise the document length, because we assume that it is identically distributed for all classes, but the paper notes that in a more general model, this too should be parameterised.

2.2 Training and Using a Classifier

We take a set of labelled training data, $D = \{d_1,\ldots,d_{|D|}\}$, and estimate θ for use by the naive Bayes classifier. Let the estimate of θ be $\hat{\theta}$. We use maximum a priori a posteriori parameter estimation to find the parameters for naive Bayes. In other words, we want to find $\arg\max_\theta P(\theta|D)$, the value of θ that is the most probable given the training data and a prior.

Firstly, using Bayes' rule, we obtain:

$$\begin{aligned} P(\theta|D) &= \frac{P(D|\theta)P(\theta)}{P(D)} \\ &\propto P(D|\theta)P(\theta) \end{aligned} \tag{5}$$

since $P(D)$ is constant from the viewpoint of maximising θ. $P(D|\theta)$ is calculated by multiplying together all the probabilities of the documents from equation (4), since we can assume that the documents are independently generated. We use:

$$P(\theta) \propto \prod_{c_j \in C} \left(\theta_{c_j} \prod_{w_t \in V} \theta_{w_t|c_j} \right) \tag{6}$$

which is the Dirichlet distribution with parameter $\alpha = 2$ in order to obtain Laplace smoothing. Laplace smoothing adds "pseudo-counts" that prevent zero probabilities from arising in the final ratio-of-words equations. For example, if we have a test document, and for each class label, we have a word that has not been seen before in documents of that class, then the probability of the document will be zero for all the classes. Using Laplace smoothing resolves this kind of problem.

Given the constraint that word probabilities must sum to 1, we have a constrained optimisation problem that can be solved using Lagrange multipliers: maximise $\log(P(\theta|D))$ subject to $\sum_{w_t} P(w_t|c_j;\theta) = 1$. (See the Mathematical Details section below for mathematical details of the use of Lagrange multipliers.)

The solution of the optimisation is the following:

$$\hat{\theta}_{w_t|c_j} = \frac{1 + \sum_{i=1}^{|D|} N(w_t, d_i) P(y_i = c_j|d_i)}{|V| + \sum_{s=1}^{V} \sum_{i=1}^{|D|} N(w_s, d_i) P(y_i = c_j|d_i)} \tag{7}$$

$$\hat{\theta}_{c_j} = \frac{1 + \sum_{i=1}^{|D|} P(y_i = c_j|d_i)}{|C| + |D|} \tag{8}$$

where $N(w_t|c_j;\theta)$ is the number of times w_t appears in d_i, and $P(y_i = c_j|d_i) \in \{0,1\}$ depends on the class label. These equations in essence tell us that we estimate the word probabilities and the class probabilities by simply using ratio of counts, supplemented by Laplace smoothing.

As we mentioned previously, the naive Bayes classifier works by calculating the probability that a document was generated by each component, and assigning the label that corresponds to the most likely component. These probabilities can be calculated as follows:

$$\begin{aligned} & P(y_i = c_j|d_i;\hat{\theta}) \\ = & \frac{P(d_i|c_j;\hat{\theta})P(c_j|\hat{\theta})}{P(d_i|\hat{\theta})} \\ = & \frac{P(c_j|\hat{\theta}) \prod_{k=1}^{|d_i|} P(w_{d_{i,k}}|c_j;\hat{\theta})}{\sum_{c_r \in C} P(c_r|\hat{\theta}) \prod_{k=1}^{|d_i|} P(w_{d_{i,k}}|c_r;\hat{\theta})} \end{aligned} \tag{9}$$

where the first equality uses Bayes' rule, and the second equality uses equation (4) on the numerator, and equations (1) and (4) on the denominator.

3 EM Framework

EM is a class of algorithms that can be used for finding maximum likelihood estimates of parameters in problems with incomplete data. In this case, the data is considered incomplete because some of them come without class labels. The EM algorithm begins with an initial set of model parameters, and uses this to compute the probability that each document belongs to a particular class. These probabilities are then used to calculate a new estimate of the parameters, which are then used to compute probabilities over the documents. This is performed iteratively until the parameters do not change, or algorithmically, until the parameters change below a certain threshold.

Firstly, we will mathematically describe the EM algorithm before the incorporation of the naive Bayes classifier. Let y be the incomplete data, and let z be the data that is missing from y. The probability function for the complete data is $P(y,z|\theta)$. Let the first value of the parameters be θ_0, and the algorithm then computes the estimates θ_1, θ_2 and so on. Hence, one iteration of the EM algorithm from θ_n to θ_{n+1} takes the following form:

$$\theta_{n+1} = \arg\max_{\theta} Q(\theta) \tag{10}$$

where $Q(\theta)$ is the expected value of the log-likelihood. In other words,

$$Q(\theta) = \sum_z P(z|y,\theta) \log P(y,z|\theta) \tag{11}$$

This comes about as follows. We begin by considering that we want to maximise the probability of the given data and the missing data:

$$\begin{aligned} l(\theta) &= \log \sum_z P(y,z|\theta) \\ &= \log \sum_z P(z|y,\theta) \frac{P(y,z|\theta)}{P(z|y,\theta)} \\ &\geq \sum_z P(z|y,\theta) \log \frac{P(y,z|\theta)}{P(z|y,\theta)} \\ &= \sum_z P(z|y,\theta) \log P(y,z|\theta) \\ &\quad - \sum_z P(z|y,\theta) \log P(z|y,\theta) \end{aligned} \tag{12}$$

where the inequality comes about from Jensen's inequality ($\log E[x] \geq E[\log(x)]$). By maximising the value of the expression on the last line, we lower bound the log-likelihood, and thus we perform hill climbing.

In the E step, we calculate the value of $P(z|y,\theta)$; for example, we calculate the missing class labels. We obtain equation (11) for the M step, where we maximise θ, because there is no θ term in $\sum_z P(z|y,\theta) \log P(z|y,\theta)$.

Note that the probability of the missing data is, by the use of Bayes' theorem and the law of total probability:

$$\begin{aligned} P(z|y,\theta) &= \frac{P(y,z|\theta)}{P(y|\theta)} \\ &= \frac{P(y|z,\theta)P(z|\theta)}{\sum_z P(y|z,\theta)P(z|\theta)} \end{aligned} \tag{13}$$

That is, we can calculate $P(z|y,\theta)$ and $P(y,z|\theta)$ in the Q equation from $P(y|z,\theta)$ and $P(z|\theta)$.

In algorithmic form, the EM algorithm looks like this:

1: Select an initial set of model parameters
2: **repeat**
3: **Expectation step)** Calculate the probability that each object belongs to each distribution, that is, calculate $P(c_j|d_i;\theta)$.
4: **(Maximisation step)** Find the new values of the parameters, θ, that maximise the expected likelihood of the data given the probabilities from the previous step.
5: **until** The parameters do not change, or the change is below a threshold

4 Combining Naive Bayes with EM

4.1 Basic EM

The paper first combines the naive Bayes classifier with the EM algorithm without augmentation. The change from the classic EM algorithm is that there is a priming stage, where the naive Bayes parameters, $\hat{\theta}$ are estimated, instead of randomly assigning the set of model parameters; also, we calculate the probabilities of component membership for the unlabelled documents only, and not the labelled documents. The algorithm now becomes this:

1: Build a naive Bayes classifier using the labelled documents only, with maximum a posteriori parameter estimation.
2: **repeat**
3: **(Expectation step)** Calculate the probability that each document belongs to each component, that is, calculate $P(c_j|d_i;\hat{\theta})$, using the current classifier, $\hat{\theta}$.
4: **(Maximisation step)** Find the new values of the parameters, $\hat{\theta}$, that maximises the expected likelihood of the data given the component membership probabilities from the previous step.
5: **until** The parameters do not change, or the change is below a threshold, as measured by the complete log-likelihood of the labelled and unlabelled data, and the prior

The output of the algorithm is a set of parameters, $\hat{\theta}$, that is locally maximal, given both labelled and unlabelled data.

4.2 Augmented EM

However, the algorithm presented above makes a number of assumptions that are likely to be false in real-world data, namely that the data is produced by a mixture model, and that there is a one-to-one correspondence between mixture components and classes.

Firstly, the paper proposes a scheme whereby the the unlabelled data is weighted. The rationale for doing so is that, because the number of unlabelled documents greatly outweighs the labelled documents, the algorithm is almost performing unsupervised clustering. This is a problem when the mixture model assumptions are not true, meaning that the clustering will not align with the class labels. Furthermore, when there are a large number of labelled documents, the case for having unlabelled data diminishes. The paper introduces a parameter λ, $0 \leq \lambda \leq 1$, which weights the contribution of unlabelled documents at λ, and the new method is termed EM-λ. When $\lambda = 1$, unlabelled documents have the same weight as a labelled document, and the method is the same as the basic EM. When $\lambda = 0$, the unlabelled documents will have no influence at all, and it is equivalent to classifying using purely naive Bayes. In practice, the value of λ can be calculated using leave-one-out cross-validation classification accuracy on the labelled training data, although it can be fine-tuned by hand as well.

Taking into account weighted unlabelled documents, we have:

1: Find λ by cross-validation.
2: Build a naive Bayes classifier using the labelled documents only, with maximum a posteriori parameter estimation.
3: **repeat**
4: **(Expectation step)** Calculate the probability that each document belongs to each component, that is, calculate $P(c_j|d_i;\hat{\theta})$, using the current classifier, $\hat{\theta}$.
5: **(Maximisation step)** Find the new values of the parameters, $\hat{\theta}$, that maximises the expected likelihood of the data given the component membership probabilities from the previous step. When using word and document counts from unlabelled documents, reduce their contribution by a factor of λ.
6: **until** The parameters do not change, or the change is below a threshold, as measured by the complete log-likelihood of the labelled and unlabelled data, and the prior

Secondly, the paper proposes the inclusion of a many-to-one correspondence between mixture components and classes. The example cited is that of a sports class, where subclasses about different sports would have disjoint sets of dependencies between words specific to a particular sport. The change is that some or all of the classes, which can be thought of as "topics", can now have multiple multinomial mixture components. Thus, even for the labelled data, there is an additional parameter that needs to be estimated, namely, which of the components covering a topic generated the particular document.

We let t_a be the ath class, or "topic", and then $P(t_a|c_j;\hat{\theta}) \in \{0,1\}$ is the many-to-one mapping between mixture components and classes. For example, if the ath topic is sport, and the jth, kth and lth components refer to hockey, baseball and acquisitions, then $P(t_a|c_j;\hat{\theta}) = P(t_a|c_k;\hat{\theta}) = 1$, but $P(t_a|c_l;\hat{\theta}) = 0$. This mapping is pre-determined; what has to be determined by the algorithm is the assignment of documents to components, not the components to classes. For us to have a consistent model, we clamp all $P(c_j|d_i;\theta)$, for which $P(y_i = t_a|c_j;\theta)$, to 0, and the probabilities over the other components is normalised to 1. Finally, we note that the initialisation of component assignments is random, matching the classic EM algorithm, and the number of mixture components for each class is given by cross-validation.

The algorithm, with multiple components per class, is now:

1: Find the number of mixture components per class by cross-validation.
2: For each labelled document, randomly assign a component probability for mixture components that correspond to the document's class label.
3: Build a naive Bayes classifier using the labelled

documents only, with maximum a posteriori parameter estimation.

4: **repeat**

5: **(Expectation step)** Calculate the probability that each document belongs to each component, that is, calculate $P(c_j|d_i;\hat{\theta})$, using the current classifier, $\hat{\theta}$. Clamp the component probabilities for documents for components not related to the document's class to zero, and renormalise.

6: **(Maximisation step)** Find the new values of the parameters, $\hat{\theta}$, that maximises the expected likelihood of the data given the component membership probabilities from the previous step.

7: **until** The parameters do not change, or the change is below a threshold, as measured by the complete log-likelihood of the labelled and unlabelled data, and the prior

5 Mathematical Details

5.1 Naive Bayes

As mentioned above, in order to find the θ that maximises $P(\theta|D)$, we solve the system of partial derivatives of $\log(P(\theta|D))$. We use Lagrange multipliers to enforce the constraint that the word probabilities must sum to 1, namely $\sum_{w_t} P(w_t|c_j;\theta) = 1$. We therefore have $|C|$ equality constraints of the form:

$$g_j(\theta) = 0, \qquad \forall j = 1,\ldots,|C| \tag{14}$$

where $g_j(\theta) = \sum_{w_t} P(w_t|c_j;\theta) - 1$.

The Lagrangian is therefore:

$$L(\theta,\lambda) = \log(P(\theta|D)) + \sum_{j=1}^{|C|} \lambda_j g_j(\theta) \tag{15}$$

where λ_j is a Lagrange multiplier.

We then find the first-order derivatives of L with respect to θ and λ, and set them to zero:

$$\frac{\partial L}{\partial \theta_{w_t|c_j}} = 0 \qquad \forall w_t \in V, c_j \in C \tag{16}$$

$$\frac{\partial L}{\partial \theta_{c_j}} = 0 \qquad \forall c_j \in C \tag{17}$$

$$\frac{\partial L}{\partial \theta_{\lambda_j}} = 0 \qquad \forall j = 1,\ldots,|C| \tag{18}$$

The second set of equations simply recovers the equality constraints. The other interesting part is the expansion of $\log(P(\theta|D))$. Combining equations (4), (5) and (6):

$$\begin{aligned}
P(\theta|D) &\propto \prod_{d_i\in D}\left(P(|d_i|)\prod_{k=1}^{|d_i|}P(w_{d_{i,k}}|c_j;\theta)\right) \\
&\quad \times \prod_{c_j\in C}\left(\theta_{c_j}\prod_{w_t\in V}\theta_{w_t|c_j}\right) \\
\log(P(\theta|d)) &= \sum_{d_i\in D}\log\left[P(|d_i|)\prod_{k=1}^{|d_i|}P(w_{d_{i,k}}|c_j;\theta)\right] \\
&\quad + \sum_{c_j\in C}\log\left(\theta_{c_j}\prod_{w_t\in V}\theta_{w_t|c_j}\right) + c \\
&= \sum_{d_i\in D}\left[\log P(|d_i|) + \sum_{k=1}^{|d_i|}\log P(w_{d_{i,k}}|c_j;\theta)\right] \\
&\quad + \sum_{c_j\in C}\left[\log\theta_{c_j} + \sum_{w_t\in V}\log\theta_{w_t|c_j}\right] + c \\
&= \sum_{d_i\in D}\log P(|d_i|) \\
&\quad + \sum_{d_i\in D}\sum_{k=1}^{|d_i|}\log P(w_{d_{i,k}}|c_j;\theta) \\
&\quad + \sum_{c_j\in C}\log\theta_{c_j} + \sum_{c_j\in C}\sum_{w_t\in V}\log\theta_{w_t|c_j}
\end{aligned} \tag{19}$$

for some constant c (because the first line is a proportional to relation).

Solving the system of equations (18) will obtain equations (7) and (8).

5.2 Basic EM

This section now discusses the integration of naive Bayes and the EM algorithm. We follow the notation in the paper and denote the set of labelled documents by D^l and the set of unlabelled documents by D^u. These sets are disjoint, and the entire set of training documents is $D = D^l \cup D^u$. The details of the maximisation are similar to the previous discussion in the context of naive Bayes; the only real change is that our documents are not all labelled. We will show that it is possible for a hill climbing procedure to find a locally maximum value of $\hat{\theta}$.

Again, we find the θ that maximises $P(\theta)P(D|\theta)$. We represent $P(\theta)$ in the same manner as before,

but we need to find new expressions for $P(D|\theta)$, which again, is simply the product over all the document probabilities. For an unlabelled document, its probability is the sum over all classes, and for a labelled document, we simply consider the probability that the document was generated by the mixture component associated with its class label. Hence:

$$\begin{aligned} P(D|\theta) &= \prod_{d_i \in D^u} \sum_{j=1}^{|C|} P(c_j|\theta) P(d_i|c_j;\theta) \\ &\quad \times \prod_{d_i \in D^l} P(y_i = c_j|\theta) P(d_i|y_i = c_j;\theta) \end{aligned} \tag{20}$$

Noting that log is a strictly monotonically increasing function, we maximise $\log(P(\theta|D))$ instead. Let $l(\theta|D) \sim \log(P(\theta)P(D|\theta)$. Applying the logarithmic laws to equation (20), we arrive at:

$$\begin{aligned} l(\theta|D) &= \log(P(\theta)) \\ &\quad + \sum_{d_i \in D^u} \log \sum_{j=1}^{|C|} P(c_j|\theta) P(d_i|c_j;\theta) \\ &\quad + \sum_{d_i \in D^l} \log(P(y_i = c_j|\theta) P(d_i|y_i = c_j;\theta)) \end{aligned} \tag{21}$$

Now, we suppose for a moment that we knew the class labels for all the documents. We will represent this in a matrix z, where:

$$z_{ij} = \begin{cases} 1 & \text{iff } y_i = c_j \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

Suppose we have an unlabelled document $d_i \in D^u$. Then:

$$\begin{aligned} &\sum_{j=1}^{|C|} P(c_j|\theta) P(d_i|c_j;\theta) \\ &= \sum_{j=1}^{|C|} z_{ij} \log(P(c_j|\theta) P(d_i|c_j;\theta)) \end{aligned} \tag{23}$$

because in the first line, only one of the terms is non-zero (we are assuming that we know for sure what the class labels are). Likewise, in the second line, the use of z_{ij} means that only one term in the summation is non-zero (a document appears in one class only). That is how we can bring the log term inside the summation.

Likewise, for a labelled document, $d_i \in D^l$:

$$\begin{aligned} &\log(P(y_i = c_j|\theta) P(d_i|y_i = c_j;\theta)) \\ &= \sum_{j=1}^{|C|} z_{ij} \log(P(c_j|\theta) P(d_i|c_j;\theta)) \end{aligned} \tag{24}$$

because the z_{ij} zeroes out the terms where $y_i \neq c_j$, and we are still only considering the probability of the document conditional on the class label.

Hence, the complete log-likelihood of the parameters can be written as:

$$\begin{aligned} &l_c(\theta|D;z) \\ &= \log(P(\theta)) \\ &\quad + \sum_{d_i \in D} \sum_{j=1}^{|C|} z_{ij} \log(P(c_j|\theta) P(d_i|c_j;\theta)) \end{aligned} \tag{25}$$

Going back to the incomplete case, we now replace z_{ij} by its expected value, and we obtain:

$$\begin{aligned} &l(\theta|D) \\ &= \log(P(\theta)) \\ &\quad + \sum_{d_i \in D} E_z \left[\sum_{j=1}^{|C|} \log(P(c_j|\theta) P(d_i|c_j;\theta)) \right] \end{aligned} \tag{26}$$

noting that the expectation operator can go outside of the inner sum because the sum of expectations is the expectation of the sum. Applying Jensen's inequality (which is $\log E[x] \geq E[\log x]$; the paper has it the wrong way around, because log is a concave down function):

$$\begin{aligned} &l(\theta|D) \\ &\leq \log(P(\theta)) \\ &\quad + \sum_{d_i \in D} \log E_z \left[\sum_{j=1}^{|C|} P(c_j|\theta) P(d_i|c_j;\theta) \right] \\ &= \log(P(\theta)) \\ &\quad + \sum_{d_i \in D^u} \log \sum_{j=1}^{|C|} P(c_j|\theta) P(d_i|c_j;\theta) \\ &\quad + \sum_{d_i \in D^l} \log(P(y_i = c_j|\theta) P(d_i|y_i = c_j;\theta)) \end{aligned} \tag{27}$$

Hence, we have found a lower bound for the expected value of the log-likelihood (the last line is equation (21)), and we can therefore find a maximum $\hat{\theta}$ by a hill climbing procedure as the EM algorithm.

In the E-step, we calculate the $P(c_j|d_i;\theta)$ for the unlabelled documents using equation (9) with the current parameter estimate, $\hat{\theta}$. These probabilities are then used to estimate a new value of $\hat{\theta}$ in the M-step. The priming M-step calculates an initial value of $\hat{\theta}$ using equations (7) and (8). The termination condition is given by thresholding the change in equation (25).

5.3 Augmented EM

We modify equation (25) to obtain:

$$\begin{aligned} & l_c(\theta|D;z) \\ = \; & \log(P(\theta)) \\ & + \sum_{d_i \in D^l} \sum_{j=1}^{|C|} z_{ij} \log(P(c_j|\theta)P(d_i|c_j;\theta)) \\ & + \lambda \left(\sum_{d_i \in D^u} \sum_{j=1}^{|C|} z_{ij} \log(P(c_j|\theta)P(d_i|c_j;\theta)) \right) \end{aligned} \tag{28}$$

Jensen's inequality can be applied to the two sub-expressions:

$$\sum_{d_i \in D^l} \sum_{j=1}^{|C|} z_{ij} \log(P(c_j|\theta)P(d_i|c_j;\theta))$$

and

$$\sum_{d_i \in D^u} \sum_{j=1}^{|C|} z_{ij} \log(P(c_j|\theta)P(d_i|c_j;\theta))$$

as before (replacing D by D^l and D^u) to create a lower bound.

Define a weighting factor:

$$\Lambda(i) = \begin{cases} \lambda & \text{if } d_i \in D^u \\ 1 & \text{if } d_i \in D^l \end{cases} \tag{29}$$

in order to reduce the effect of unlabelled documents on the word counts from equations (7) and (8). The following two equations follow intuitively from this desire to weight the unlabelled documents differently:

$$\hat{\theta}_{w_t|c_j} = \frac{1 + \sum_{i=1}^{|D|} \Lambda(i) N(w_t, d_i) P(y_i = c_j|d_i)}{|V| + \sum_{s=1}^{V} \sum_{i=1}^{|D|} \Lambda(i) N(w_s, d_i) P(y_i = c_j|d_i)} \tag{30}$$

$$\hat{\theta}_{c_j} = \frac{1 + \sum_{i=1}^{|D|} \Lambda(i) P(y_i = c_j|d_i)}{|C| + \left|D^l\right| + \lambda |D^u|} \tag{31}$$

For the multiple mixtures case, we modify equation (9) to obtain:

$$\begin{aligned} & P(t_a|d_i;\hat{\theta}) \\ = \; & \sum_{c_j \in C} P(t_a|c_j;\hat{\theta}) \frac{P(c_j|\hat{\theta}) \prod_{k=1}^{|d_i|} P(w_{d_{i,k}}|c_j;\hat{\theta})}{\sum_{c_r \in C} P(c_r|\hat{\theta}) \prod_{k=1}^{|d_i|} P(w_{d_{i,k}}|c_r;\hat{\theta})} \end{aligned} \tag{32}$$

Instead of classifying straight into class probabilities, we first "classify" the documents into the mixture components, and then add up the probabilities for a particular topic/class.

6 Evaluation

6.1 Datasets and Measures

A number of datasets are considered in the paper for use in evaluating the proposed algorithms. The three are 20 NEWSGROUPS, WEBKB and REUTERS. For all three data sets, some documents are set aside to be the test set, and a large proportion of documents are set aside to be the unlabelled documents. The remainder become the labelled documents, and are divided into a number of non-overlapping subsets, which allows the authors of the paper to perform each experiment multiple times. The size of the subsets can be changed, and the number of subsets is affected accordingly (since the total number of labelled documents available is constant). The average over all trials (that share the same number of labelled documents) is reported. A summary of the datasets appears in Table 1.

The 20 NEWSGROUPS dataset consists of 20017 articles divided among 20 different UseNet discussion groups, which are easily confusable. The feature selection is as follows: the text is tokenised by taking contiguous alphabetic characters, and words in a stoplist are removed; the word counts are then normalised with respect to the document length. The vocabulary consists of 62258 unique words. Because in practice, a classifier would be asked to classify future articles based on the ones already seen, the test set consisted of the last 20% of articles by date, which is 4000 documents. The unlabelled set consists of 10000 documents cho-

Dataset	# Docs	# Topics	# Vocab	# Test	# Unlabelled	# Labelled	Metric
20 NEWSGROUPS	20017	20	62258	4000	10000	6000	Accuracy
WEBKB	4199	4	300	3/4 unis	2500	(remainder)	Accuracy
REUTERS	12902	90	†	3299	7000	2603	P-R

Figure 1: A summary of the datasets. † The number depends on the category being experimented upon.

sen at random from the remainder, and the remaining 6000 are available as labelled documents. For this dataset, the metric used was accuracy, which is the proportion of correct class assignments divided by the total number of assignments.

The WEBKB dataset contains web pages from computer science departments in seven categories. The paper uses the four-most populous non-other categories, resulting in 4199 pages. The feature selection is as follows: numbers are converted into specific token-types, no stoplist is used (because common words tend to have good information gain in this case), and the 300 most informative words, as measured by average mutual information with the class variable, constituted the vocabulary. Again, to mimic the supposed real-world use, four test sets were created by leaving one of the four universities out of the dataset. Of the remainder, 2500 were randomly selected for use as the unlabelled set, and the rest were available as labelled documents. Again, the metric used for this dataset was accuracy.

Finally, the REUTERS dataset consists of 12902 articles and 90 topics. The feature selection is as follows: all the words except for certain common tags are used, filtered through a stoplist. The vocabulary size needs to be optimised for a particular class, and the appropriate value was found via leave-one-out cross validation. This is possible, because the REUTERS dataset was subjected to binary classifiers, unlike the other two datasets. Again, because of time dependencies, the authors chose to use the last 3299 documents as the test set. 7000 documents from the remainder are used as the unlabelled set, and the remainder are available as labelled documents. Because of the disparity between the positive and negative class in this particular case, labelled document sets are weighted with four times more negative samples than positive samples. For this dataset, the metric used was precision-recall breakeven points, which is standard for binary classification, as opposed to accuracy. The precision-recall breakeven point is the the precision and recall value at which the two are equal, and the classifier can trade-off between the two by shifting the boundary between the positive and the negative class. This metric was used instead of accuracy, because the vast number of negative documents means that high accuracy is easy to obtain by always predicting the negative class.

The authors make a note about their use of leave-one-out cross-validation that is interesting. In order to be more computationally efficient, when testing a particular labelled document, the word counts of that document are removed before testing that document against the model, and thus only one run of EM is required instead of one run per document. However, some residual effects of the held-out document may remain to affect the optimality of the cross-validation. After the experiments have been conducted, the authors hypothesise that certain problems were due to this residual effect.

6.2 Experiments

The authors performed several experiments to test their algorithms. The experiments were:

1. Using 20 NEWSGROUPS, the authors looked at the accuracy of the classifier when it was given varying amounts of labelled documents (from 20 up to 5500 documents), and either 0 or 10000 unlabelled documents.

2. Using 20 NEWSGROUPS, the authors looked at the effect of varying the amount of unlabelled data. For 40, 140, 300, 600 and 3000 labelled documents, they varied the number of unlabelled documents (from 0 up to 13000).

3. Using WEBKB, the authors considered the accuracy when the classifier was given varying amounts of labelled documents (from 4 to 400 documents), and either 0 or 2500 unlabelled documents. This is essentially Experiment 1 repeated for WEBKB.

4. Using WEBKB, the authors moved to EM-λ. For 3 different numbers of labelled documents (40, 80, 200), and a fixed number of unlabelled documents (2500), they changed the value of λ and reported the accuracy.

5. Using WEBKB, the authors varied the number of labelled documents (from 4 to 400) and varied the contribution of the unlabelled data: none at all, 2500 with basic EM, 2500 with λ chosen via cross-validation, and 2500 with a perfectly chosen λ. This is an extension of Experiment 3.

6. Using REUTERS, the authors allowed introduced multiple mixture components per class, and compare the results to that of basic EM, naive Bayes, and naive Bayes with multiple mixture components per class. The authors also compare the effect of choosing the number of mixture components via cross-validation instead of choosing it optimally. Note that the measure changes to precision-recall breakeven points.

6.3 Results and Analysis

Note that the experiment numbers are my own and refer to the numbered list in the previous section.

6.4 Experiment 1

6.4.1 Results

- The EM classifier performs better than the naive Bayes classifier. Here are some results for selected values of number of labelled documents:

# Labelled Documents	NB (%)	EM (%)
20	20	35
300	52	66
5500	76	78

- The results are statistically significant, and at 300 documents, it represents a 30% reduction in classification error.

6.4.2 Analysis

- The introduction of unlabelled data improved the accuracy of the performance of the system.

- However, the benefit of unlabelled data decreases as the number of labelled documents increases.

- There is another way to look at the results. When unlabelled documents are added, it takes far fewer labelled documents to achieve the same accuracy.

6.5 Experiment 2

6.5.1 Results

- In general, the more unlabelled data there is, the better the performance. With 40 labelled documents, accuracy goes up from under 30% to over 40%, when going from 0 to over 10000 unlabelled documents.

- Small amounts of unlabelled data together with labelled data hurts performance.

6.6 Analysis

- The benefit of adding large volumes of unlabelled data is small when there are a large number of labelled documents (for example, by considering the line in the Figure 3 for 3000 labelled documents).

- With small amounts of unlabelled data, it is hypothesised that the overly-confident estimates in the E-step causes the unlabelled data to be assigned to different classes too sharply, thus causing the drop in accuracy.

6.7 Experiment 3

6.7.1 Results

- Here are some accuracy measurements for varying numbers of labelled documents:

# Labelled Documents	NB (%)	EM (%)
4	50	55
240	81	79

- These results are statistically significant.

6.7.2 Analysis

- The EM can hurt performance when the data does not fit the assumptions of the model. The large proportion of unlabelled data to labelled data can throw off parameter estimation.

6.8 Experiment 4

6.8.1 Results

- For 40 labelled documents, the optimal value of λ is high (around 0.9). For 80 labelled documents, the optimal value of λ is mid-range (around 0.45), and for 200 labelled documents, the optimal value of λ is low (around 0.1).

- The peaks were in the middle, and not at the end, and thus EM-λ performed better than either naive Bayes or basic EM.

6.8.2 Analysis

- We can conclude that there are few labelled documents, the value of unlabelled documents is high, and therefore the algorithm weights the unlabelled documents higher; when there are more labelled documents, the value of unlabelled documents is lower.

- The experiment shows an inverse relationship between the labelled dataset size and the best weighting factor.

6.9 Experiment 5

6.9.1 Results

- When λ was chosen by cross-validation, it ran with a higher accuracy than basic EM with a large number of labelled documents.

- When the number of labelled documents was low, choosing λ with cross-validation tended to result in fluctuating accuracies with respect to the basic EM curve.

- Choosing λ perfectly resulted in accuracies significantly higher than basic EM.

6.9.2 Analysis

- It shows that while cross-validation helps with choosing a good value of λ, there is still some room for improvement, because the perfectly chosen one outperforms it, especially for small numbers of labelled documents, and is more stable in its results.

- Cross-validation seems to be problematic when faced with smaller numbers of labelled data; this is because there are fewer documents to perform cross-validation with, and the authors suggest using other methods of selecting λ.

6.10 Experiment 6

6.10.1 Results

- The precision-recall breakeven values for a number of Reuters categories were presented for the various classification methods examined. NB1 and EM1 are the basic naive Bayes and EM respectively. The asterisk denotes the use of multiple mixture components. In general, we can see that using multiple mixture components helps both NB and EM, but selecting the number of components via cross-validation degraded performance with respect to choosing the value optimally. We quote a few results:

Category	NB1	NB*	EM1	EM*	EM*CV
acq	69.4	74.3	70.7	83.9	75.6
corn	44.3	47.8	44.6	52.8	47.1
crude	65.2	68.3	68.2	75.4	68.3
earn	91.1	91.6	89.2	89.2	87.1
grain	65.7	66.6	67.0	72.3	67.2

- The median number of components used for the negative class across the trials was also reported. On average, the authors found that EM* uses more mixture components than NB*, while the number of components chosen for NB*CV was low. We quote a few results:

Category	NB*	EM*	EM*CV
acq	4	10	1
corn	3	5	3
crude	2	8	1
earn	1	1	1
grain	2	8	1

6.10.2 Analysis

- It is interesting to see that adding multiple mixture components helps with improving the performance of the naive Bayes classifier as well as the EM classifier.
- Basic EM is often worse than NB, and this could be because of the significantly violated assumptions (the negative class cannot be modelled with just one mixture component).
- The results for EM* show that combining EM with multiple mixture components is superior to just using multiple mixture components.
- EM* uses more mixture components than NB* on average, which the paper suggests is due to the fact that unlabelled data adds expressiveness to the model and reduces variance (which would come about because having multiple mixtures allows the data to fit the model better).
- The authors also present results that show that having too few, or too many, mixture components hurts performance; it is hypothesised that for small number of mixture components, the model is too restrictive, while with too many mixture components, the degradation in performance comes about because the model needs to predict more parameters than it can support.
- Finally, the authors reproduced the results in terms of accuracy. The point to take away here is that accuracy tends to be high for this particular dataset in general because of the large number of documents in the negative class. Using multiple mixture components still helps, because its inclusion makes the modelling for the negative class better, which is then reflected in accuracy.

7 Related Works

We outline how the paper differs from previous work.

Classical EM work The paper notes that using EM to classification is not a new idea, and the idea that the unclassified data can be treated as the incomplete data was mentioned by R. J. A. Little. The novelty in this paper would appear to be the combination of EM and a naive Bayes classifier.

Combining labelled and unlabelled data using EM Shahshahani and Landgrebe (1994) use a mixture of Gaussians, and Miller and Uyar (1997) use Mixtures of Experts, which differs from the naive Bayes classifier used in this paper. The difference also lies in the testing; these papers used up to 40 features on non-text data, whereas in this paper, they used many three orders of magnitude more features with textual data.

Theoretical utility of unlabelled data Shahshahani and Landgrebe (1994) analysed the convergence rate when introducing unlabelled data, but this paper criticises the results by saying that the assumption that unlabelled data can improve classification results in the absence of labelled data is unrealistic.

Using EM to fill in missing values Ghahramani and Jordan (1994) also use EM with mixture models to fill in missing values. In this paper, the focus is on estimating the class label, whereas in that paper, the focus was on features other than the class label.

Combination of EM with naive Bayes model Cheeseman & Stutz (1996), in the AutoClass project, differ from the current paper because their emphasis is in clustering, for unsupervised learning over unlabelled data.

Multiple mixture components Jaakkola and Jordan (1998) considered using mixtures to improve mean field approximations (such as naive Bayes), but the emphasis seems to be of a general nature.

Active learning Recent work by some of the authors of this paper, McCallum & Nigam (1998), have combined EM with active learning, which asks a human labeller for the label of an unlabelled document repeatedly as part of the model building process. This appears to be an extension of the work in the current paper.

Unlabelled data in support of supervised learning Blum and Mitchell (1998) consider problems where each instance can be partitioned into two sets such that each is sufficient to learn the target function; they can then learn two classifiers, which train each other over unlabelled data. This paper does not have the luxury of having such a redundancy in the data.

Statistical techniques for text classification The authors of this paper note that many other techniques other than naive Bayes have been used. Various authors in the literature have considered techniques such as Support Vector Machines, k nearest neighbour and TFIDF/Rocchio.

8 Weaknesses

We outline some weaknesses of the approach on the paper and potential ways of overcoming them.

8.1 Large Number of Assumptions

The use of a naive Bayes classifier entails that we make a large number of assumptions, even after we relaxed some of the assumptions required for the generative model later. The problem with having a large number of assumptions (or any assumptions at all) is that the model does not directly fit the text, which results in a decline in quality of classifications. For example, the assumptions that the words of a document are generated independently of context, and that the probability of a word is independent of its position in the document, are clearly false, because of such things as syntax. This can be overcome by introducing more features/parameters into the model, by parameterising, say, the probability of a word w_t being found at position k in a document of class c_j, instead of just the existence of that word in such a document. In addition, there is the assumption that the length of a document is independent of the class, which is also not necessarily true. Again, a resolution to this problem is by introducing a new parameter to do with document length. These new parameters must be estimated by the EM algorithm as well.

8.2 Use of Laplace Smoothing

The use of Laplace smoothing is problematic, and there are measures that perform better than it. The problem with Laplace smoothing is that it places too much of the probability mass on the words that are not seen, especially for short documents and a large vocabulary. This results in a significant distortion of the word counts. Other smoothing methods such as the M-estimate, or Good-Turing Estimation, may be more appropriate, although some methods may be more computationally expensive.

8.3 Finding Parameters using Cross-Validation

With the augmented EM method, a number of parameters must be put in by hand or discovered by cross-validation. This is time-consuming and potentially error-prone (as discovered by the authors when their leave-one-out cross-validation resulted in less-than-optimal results), and in a production system, this may bias the classifier to the training data, as opposed to being as general as possible. Furthermore, if you have a small number of labelled documents to start with (and this is the problem the paper tries to address), then cross-validation cannot be performed reliably because of the small number of available documents. A way around this would be to perform further experiments that seek to find relationships for the parameters that require cross-validation (such as the number of components per class) without the use of cross-validation. For example, one might come up with a rule that gives the number of components as a function of the size of the vocabulary. Another alternative is to use active learning to give more information to the classifier.

8.4 Practical Application

The paper notes that one motivation for using unlabelled data is that there are instances when users are not prepared to label so many documents as to make the classifier accurate enough. What would be useful then is if the classifier learnt from its mistakes, so that the pool of labelled documents grows over time, which can only be a good thing. This appears to have been addressed in their later work with active learning. However, in general, one can question the need for unlabelled data, despite the results that they present in this paper. In most research domains, collecting several hundred labelled documents is not overly onerous, and this paper's approach only adds a marginal amount of improvement for increased complexity when you go beyond a small number of documents. As an alternative to learning joint probabilities via the use of unlabelled data, dictionaries and thesauri, or other compilations of linguistic knowledge, may be more useful, at least when we are dealing with textual data.

9 Conclusion

We considered the combination of the naive Bayes classifier with the EM algorithm, to allow for the use of unlabelled data in text classification. Although this increases accuracy in general, performance is affected when the assumption underlying the model are challenged. Two amendments to the basic algorithm were proposed, that ameliorate the affects of violated assumptions and remove the source of one of the problems, and this results in improvements in the classification.

References and Bibliography

[1] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Mach. Learn.*, 39(2-3):103–134, 2000.

[2] P. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining.* Addison Wesley, 2005.