A Model for Handling Approximate, Noisy or Incomplete Labeling in Text Classification

Ganesh Ramakrishnan GANRAMKR@IN.IBM.COM
Krishna Prasad Chitrapura KCHITRAP@IN.IBM.COM
Raghu Krishnapuram KRAGHURA@IN.IBM.COM
IBM India Research Lab, IIT, New Delhi, India

Pushpak Bhattacharyya PB@CSE.IITB.AC.IN
Department of C.S.E, IIT Bombay, Mumbai, India

Abstract

We introduce a Bayesian model, BayesANIL, that is capable of estimating uncertainties associated with the labeling process. Given a labeled or partially labeled training corpus of text documents, the model estimates the joint distribution of training documents and class labels by using a generalization of the Expectation Maximization algorithm. The estimates can be used in standard classification models to reduce error rates. Since uncertainties in the labeling are taken into account, the model provides an elegant mechanism to deal with noisy labels. We provide an intuitive modification to the EM iterations by re-estimating the empirical distribution in order to reinforce feature values in unlabeled data and to reduce the influence of noisily labeled examples. Considerable improvement in the classification accuracies of two popular classification algorithms on standard labeled data-sets with and without artificially introduced noise, as well as in the presence and absence of unlabeled data, indicates that this may be a promising method to reduce the burden of manual labeling.

1. Introduction

Text classification has been studied extensively for information retrieval and other knowledge management applications. Some commonly used supervised techniques in the literature for text classification are naïve Bayes (Lewis, 1998; McCallum & Nigam, 1998), support vector machines (Joachims, 1998), k-nearest neighbor (Yang, 1999) and maximum entropy (Nigam et al., 1999).

The task of text classification is hampered by the lack of large amounts of correctly-labeled examples. The generation of training data is typically achieved by manual assignment of class labels to documents by experts. Manual annotations inherently exhibit a certain level of *approximation or uncertainty*. When faced with the challenge of selecting a class label from a set of similar or confusing class labels for a document, the expert often chooses a class label that seems the most plausible. It is desirable that a supervised learning algorithm accounts for the approximation involved in manual labeling by using an estimate of the uncertainty with which a document was labeled.

The tedious and uncertain nature of manual labeling further aggravates the requirement for large amounts of labeled data by classifiers in order to generalize well. This has resulted in considerable amount of research on methods that learn with a very small number of labeled instances along with large amounts of unlabeled data (Nigam et al., 2000; Ando & Zhang, 2004). An important issue to be addressed while designing learning algorithms that generalize from a very small number of labeled examples is how to account for features that are poorly represented in the labeled training examples but are prominent in the unlabeled examples. Probabilistic latent semantic indexing (Hofmann, 1999) performs unsupervised smoothing of words, based on their co-occurrence under common *aspects* or concepts. It is however tailored to information retrieval task and seems less appropriate for text classification. Feature smoothing techniques such as Laplace, Lidstone and Jeffrey-Perks smoothing (Grif-

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

fiths & Tenenbaum, 2001) take away some probability mass from seen features and distribute this mass amongst unseen features. However, these methods do not account for the empirical distribution of features in unlabeled documents.

In many scenarios, it is easy to generate a labeled dataset with some amount of noise in the labeling. For instance, this can be achieved by querying the Web or some document collection. A practical text learning algorithm needs to be resilient to such noisy labeling. Recently, Lee and Liu (2003) modeled the problem of learning from a mixture of positive and unlabeled examples as a two-class learning problem with noisy negative examples. They use weighted logistic regression along with an estimate for the proportion of noise, to learn the separation of positive and negative classes. In practice, the proportion of noise in the labeled collection can be approximated by the proportion of noise in a random sample from the collection.

Brodley and Friedl (1996) and Domingos (1999) have proposed methods of wrapping a filter or a meta-learner around a standard classifier. The former uses the filter to make the native classifier immune to mislabeled training instances by iteratively removing the training instances that can be potentially misclassified under many models. However, this method cannot counter the issues arising from the ambiguous labeling of training documents. The latter meta-learner makes the learning algorithm cost-sensitive when applied to data-sets with imbalanced classes.

In this paper, we present BayesANIL, a Bayes Network-based model that is capable of dealing with approximate, noisy or incomplete labeling of text documents. BayesANIL imposes a multinomial naïve Bayes model on the documents and learns a joint probability distribution, $Pr(d, z)$, between document ids d and the class labels z. $Pr(d, z)$ indicates how well each document d fits into the class z, thus effectively providing a way of relabeling the documents. This is because $Pr(d)$ is an estimate of how correctly the document was labeled. We show in the later sections that these probabilities can be used by standard classifier learners such as naïve Bayes and SVM to learn more accurate classifiers. Given an estimate of the proportion of noise, our model can counter the occurrence of *class noise* (Brodley & Friedl, 1996). We also show that our model can be used in a labeled/unlabeled setting. In this setting, we illustrate how BayesANIL folds in feature evidence from unlabeled documents, while simultaneously labeling the unlabeled documents. This is especially important, given that the features are often poorly represented in the labeled set.

The layout of the rest of the paper is as follows. In Section 2, we describe the role of BayesANIL in a classification setting. Section 3 describes in details the BayesANIL model and associated algorithms for parameter estimation. Section 4 suggests how BayesANIL could be used in conjunction with existing classification algorithms. We finally present experimental results in Section 5 and our conclusions and future work in Section 6.

2. A Model for Learning

2.1. Notation

We consider input space $\mathcal{D}$ of documents. D is a random variable that takes on values d in the set $\mathcal{D}$. We let $\mathcal{Z}$ denote the space of class labels and let Z denote a random variable that takes on values z from $\mathcal{Z}$. The document collection consists of a set of labeled documents represented by $\langle d, z\rangle$ and a set of unlabeled documents. In our setting, we use this document collection with noisy or incomplete labels to train standard classifiers. Each document is composed of words that belong to a vocabulary space $\mathcal{W}$. W is a random variable which takes on values w from the vocabulary $\mathcal{W}$. We have used lower-case alphabets, d, w and z as indices into the document collection $\mathcal{D}$, vocabulary set $\mathcal{W}$ and set of class labels $\mathcal{Z}$ respectively. For e.g., $Pr(w, d', z)$ denotes the joint probability of the word indexed by w, the document with id d' and the class label indexed by z.

2.2. Role of BayesANIL

We propose BayesANIL as an intermediary between the document collection and the classifier learning algorithm. BayesANIL interacts with the collection requesting for instances $\langle d, z\rangle$. BayesANIL uses these instances to learn the distribution $Pr(d, z)$ $\forall\{d, z\} \in \mathcal{D} \times \mathcal{Z}$. In practice, the class labels associated with the documents may be approximate or even noisy. BayesANIL accounts for approximate and noisy labeling by estimating the degree to which the document d belongs to the associated class label z. In addition to learning from labeled instances from $\mathcal{D}$, BayesANIL is also capable of learning from a collection of labeled and unlabeled documents from $\mathcal{D}$.

Once BayesANIL completes the estimation of parameter, the classifier learner gets tuples of $\langle d, z\rangle$ from the document collection and requests BayesANIL for an estimate of $Pr(d, z)$. The classifier learner can use the $Pr(d, z)$ estimates while learning the classification function. For instance, $Pr(d, z)$ can be used as a measure of *support* for the assignment of label z to d and

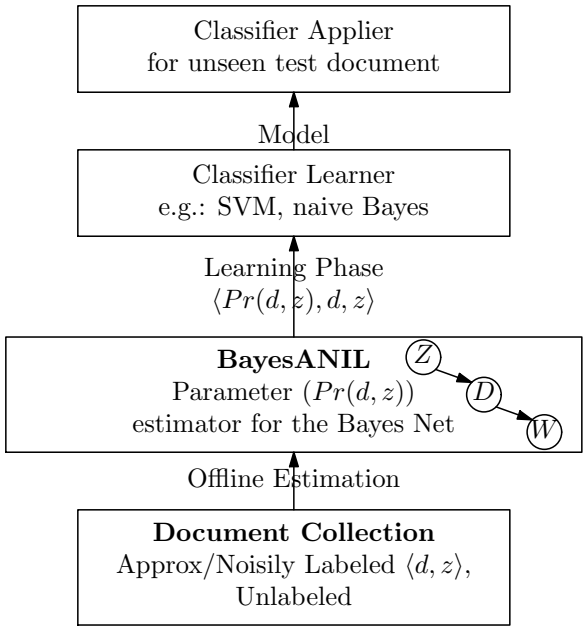

Figure 1. The Role of BayesANIL in text classification.

to weigh the instance $\langle d, z\rangle$ during training.

3. The BayesANIL Model

We assume a model for generation of data, given by the Bayesian Network $Z \rightarrow D \rightarrow W$ (also shown in figure 1). An implication of the model is that, unlike the traditional naïve Bayes generative model, a class generates document instances, each of which is a bag of words. The Bayesian assumption in the model implies that

$$Pr(w|d, z) = Pr(w|d) \tag{1}$$

We compute $Pr(W = w|D = d)$ (abbreviated as $Pr(w|d)$) as the fraction of times word w occurs across all words in document d. The Bayesian assumption also implies that a class is independent of each word given the document, *i.e.* $Pr(z|d, w) = Pr(z|d)$. We treat the $Pr(w|d)$ and $\langle d, z\rangle$ as observables and estimate the parameters $Pr(d, z)$ based on the observations.

3.1. Parameter Estimation

We estimate the model parameters $Pr(d, z)$, by maximizing a log-likelihood objective function - the log of the likelihood of observing N pairs of word-class ids, $\langle w_n, z_n\rangle$, given the model parameters $Pr(d, z)$.

Let $\mathcal{D}_z \subseteq \mathcal{D}$ be the set of documents in the training set that are assigned to class z. Then, $n(w, z)$ is the number of times the word w is seen across all documents in $\mathcal{D}_z$. Let $n(w, d)$ denote the count of w in d and $size(d)$ denote the total count of words in d. Following (McCallum & Nigam, 1998), we scale the count of words in a document, to a common length L (which can be the least common multiple of $size(d)$ across all documents d) and use the scaled word counts instead of the original counts. This way, we avoid the need to account for document lengths. We obtain $n(w, z)$ from the document collection as in (2).

$$\begin{aligned} n(w, z) &= \sum_{d \in \mathcal{D}_z} n_{scaled}(w, d) = \sum_{d \in \mathcal{D}_z} \frac{n(w, d)L}{size(d)} \\ &= L \sum_{d \in \mathcal{D}_z} Pr(w|d). \end{aligned} \tag{2}$$

The objective function is expanded in (3).

$$\begin{aligned} LL &= \sum_{n=1}^{N} \log Pr(w_n, z_n) \\ &= \sum_{w \in \mathcal{W}} \sum_{z \in \mathcal{Z}} n(w, z) \log Pr(w, z). \end{aligned} \tag{3}$$

The objective is to estimate the parameters $Pr(d, z)$, such that the log-likelihood in (3) is maximized, while satisfying the constraint

$$\sum_{d \in \mathcal{D}} \sum_{z \in \mathcal{Z}} Pr(d, z) = 1. \tag{4}$$

We use the Expectation Maximization (Dempster et al., 1976) (EM) algorithm for parameter estimation. The log-likelihood can be written in a more general form in terms of the empirical distribution, $q(w, z)$, of the data (Amari, 1995). Using (2), $q(w, z)$ can be derived as

$$q(w, z) = \frac{n(w, z)}{L|\mathcal{D}_z|} = \frac{1}{|\mathcal{D}_z|} \sum_{d \in \mathcal{D}_z} Pr(w|d). \tag{5}$$

The convergence properties of the EM algorithm hold when $q(w, z)$ is used instead of $n(w, z)$ (Amari, 1995). The modified objective function using $q(w, z)$ is given by

$$LL_{mod} = \sum_{w \in \mathcal{W}} \sum_{z \in \mathcal{Z}} q(w, z) \log Pr(w, z). \tag{6}$$

We express LL_{mod} in terms of the parameters $Pr(d, z)$ and observations $Pr(w|d)$ by expanding the right hand side of (6) and using the assumption in (1).

$$LL_{mod} = \sum_{w \in \mathcal{W}} \sum_{z \in \mathcal{Z}} q(w, z) \log \sum_{d \in \mathcal{D}} Pr(d, z) Pr(w|d) \tag{7}$$

The method of Langrange multipliers can now be used to maximize LL_{mod}, subject to the constraint in (4).

We convert this problem into an equivalent problem of maximizing the objective function O over the parameters $Pr(d, z)$ and the Lagrange multiplier γ as follows:

$$O = \sum_{w\in\mathcal{W}}\sum_{z\in\mathcal{Z}} q(w,z)\log\left(\sum_{d\in\mathcal{D}} Pr(w|d)Pr(d,z)\right) - \gamma\left(\sum_{d\in\mathcal{D}}\sum_{z\in\mathcal{Z}} Pr(d,z) - 1\right) \quad (8)$$

The condition for maximum value of the objective function O is obtained by partially differentiating it with respect to each of the parameters $Pr(d', z')$ as well as γ and setting the partial derivatives to 0 as shown in (9). $\forall d' \in \mathcal{D}, z' \in \mathcal{Z}$,

$$\frac{\partial O}{\partial Pr(d',z')} = \sum_{w\in\mathcal{W}} q(w,z')\frac{Pr(w|d')}{\sum_{d\in\mathcal{D}} Pr(w|d)Pr(d,z')} - \gamma = \sum_{w\in\mathcal{W}} q(w,z')\frac{Pr(d'|w,z')}{Pr(d',z')} - \gamma = 0 \quad (9)$$

$$Pr(d',z') = \frac{\sum_{w\in\mathcal{W}} q(w,z')Pr(d'|w,z')}{\gamma} \quad (10)$$

Solving (4) and (10) and eliminating γ, we obtain, $\forall d' \in \mathcal{D}$ and $z' \in \mathcal{Z}$

$$Pr(d',z') = \frac{\sum_{w\in\mathcal{W}} q(w,z')Pr(d'|w,z')}{\sum_{d\in\mathcal{D}}\sum_{z\in\mathcal{Z}}\sum_{w\in\mathcal{W}} q(w,z)Pr(d|w,z)} \quad (11)$$

Further, using the Bayes rule and the independence assumption, $\forall w \in \mathcal{W}$, we obtain

$$Pr(d'|w,z') = \frac{Pr(d',z')Pr(w|d')}{\sum_{d\in\mathcal{D}} Pr(d,z')Pr(w|d)} \quad (12)$$

Putting together the *Expectation step* in (12) with the *Maximization step* in (11), we can state the algorithm as in figure 2. We initialize parameters $Pr(d, z)$ as a uniform distribution. The *Expectation* and *Maximization* steps are stopped when there is no significant change in the values of the model parameters across two successive iterations as detailed in Section 3.3.

Input: Empirical distribution $q(w, z)$, $Pr(w|d)$.
Initialize the parameters $Pr(d, z)$ as a uniform distribution: $Pr(d,z) = \frac{1}{|\mathcal{D}||\mathcal{Z}|} \forall d \in \mathcal{D}, z \in \mathcal{Z}$
while the stopping criterion is not met **do**
  E Step:
  for Each document $d' \in \mathcal{D}$ **do**
    for Each word $w \in \mathcal{W}$ **do**
      for Each class $z' \in \mathcal{Z}$ **do**
        $Pr(d'|w,z') = \frac{Pr(d',z')Pr(w|d')}{\sum_{d\in\mathcal{D}} Pr(d,z')Pr(w|d)}$
      end for
    end for
  end for
  M Step:
  for Each document $d' \in \mathcal{D}$ **do**
    for Each class $z' \in \mathcal{Z}$ **do**
      $Pr(d',z') = \frac{\sum_{w\in\mathcal{W}} q(w,z')Pr(d'|w,z')}{\sum_{d\in\mathcal{D}}\sum_{z\in\mathcal{Z}}\sum_{w\in\mathcal{W}} q(w,z)Pr(d|w,z)}$
    end for
  end for
end while
Output: Estimated distribution $Pr(d', z')$, $\forall d' \in \mathcal{D}, z' \in \mathcal{Z}$

Figure 2. The Expectation Maximization Algorithm for estimating parameters of the Bayesian Network in Figure 1.

3.2. Re-estimating the Empirical Distribution

The empirical distribution $q(w, z)$ is initially obtained only from the labeled set of documents in the training set. Across iterations, as the $Pr(d, z)$ parameters for the labeled and the unlabeled documents are estimated, it is desirable that distribution of words in the unlabeled documents supplement the empirical distribution for the subsequent iterations. Simultaneously, the effect of words from the correctly labeled documents needs to be reinforced and those from the incorrectly labeled documents needs to be reduced from the empirical distribution. A possible way of re-estimating the empirical distribution $q_n(w, z)$ in the n^{th} iteration from the empirical distribution $q_{(n-1)}(w, z)$ and probability estimates $Pr_{(n-1)}(d, z)$ from the $(n-1)^{th}$ iteration is given below.

$$q_n(w,z) = (1-\lambda)q_{(n-1)}(w,z) + \lambda Pr_{(n-1)}(w,z) = (1-\lambda)q_{(n-1)}(w,z) + \lambda\sum_{d\in\mathcal{D}} Pr(w|d)Pr_{(n-1)}(d,z) \quad (13)$$

We employ λ as a smoothing parameter. In the case of learning in the presence of classification noise, λ serves as an estimate of the proportion of classification noise in the training data.

An implication of (13) is that a word w in an unla-

beled document d gets associated with a class label z based on the degree to which d belongs to z. The word w may have been previously unseen in the training documents of class z, *i.e.* $q_0(w,z) = q(w,z) = 0$. However, the maximization step in (11) ensures a non-zero value for $Pr_{(n-1)}(d,z)$, if this document has some other words in common with documents labeled with class z. Hence, the re-estimated value $q_1(w,z)$, as per (13) will be non-zero and weighed by $Pr_{(n-1)}(d,z)$, which reflects the support for membership of document d in class z. This effect of word co-occurrence on the empirical distribution has a transitive effect as the iterations proceed. Further, this way of re-estimating the empirical distribution results in denser matrix of $Pr(d,z)$ values across iterations.

3.3. Practical Considerations

The BayesANIL classification model has a small number of parameters to be estimated, *viz.*, $O(|\mathcal{D}||\mathcal{Z}|)$. However, it needs to represent some amount of sparse information such as $Pr(w|d)$, $q(w,z)$, *etc.* Further, the algorithm in Figure 2 iterates over each feature, document and class to estimate the required parameters. In this section, we propose methods for efficiently storing the sparse representation and methods for speeding up the computations.

We used a Java-based sparse matrix implementation[1] to represent our data structures for $Pr(w|d)$ and $q(w,z)$. The need for a multi-dimensional matrix representation for $Pr(d'|w,z')$ was completely eliminated by introducing a minor change to the structure of the algorithm in Figure 2. We merged the E and M steps, to estimate the new values of $Pr(d',z')$ from old values $(Pr_o(d',z'))$ as shown in Figure 3.

$Pr_o(d',z') = Pr(d',z')$; $Pr(d',z') = 0$
for Each word $w \in \mathcal{W}$ **do**
  for Each class $z' \in \mathcal{Z}$ **do**
    $Pr(w,z') = \sum_{d \in \mathcal{D}} Pr_o(d,z') Pr(w|d)$
    for $\forall d' \in \mathcal{D} | w$ occurs in d' **do**
      $Pr(d',z') = Pr(d',z') + \frac{q(w,z') Pr_o(d',z') Pr(w|d')}{Pr(w,z')}$
    end for
  end for
end for

Figure 3. EM iterations restructured for efficient storage and computation

In order to speed up the convergence in our experiments, we stop the EM iterations when the change in log-likelihood across two successive iterations, i and $(i+1)$, is less than 0.01% of the log-likelihood computed in the i^{th} iteration. As an exception to this, in the experiment described in section 5.2.2, we use convergence of the classification accuracy on a held-out validation set as a stopping criterion, to be consistent with the work with which we compare therein.

[1] http://www.math.uib.no/~bjornoh/jmp/

4. Using BayesANIL's parameters in Classifiers

4.1. Weighted naïve Bayes

Multinomial naïve Bayes classifier uses the Bayes rule to compute for each class z an estimate of the probability of z given the unseen document based on the parameters, $Pr(w|z)$, estimated *a posteriori* from the training documents (McCallum & Nigam, 1998). Traditionally, the classifier uses an estimate of $Pr(w|z)$ based simply on the number of times word w occurs in the training data for class z, divided by the total number of word occurrences in the training data for that class. However, BayesANIL has an estimate of $Pr(w|z)$ which takes into account the degree to which the training documents belong to a particular class z. Based on the model assumptions, $Pr(w|z)$ can be derived in terms of parameters $Pr(d,z)$ as in (14). We call the classifier that uses these $Pr(w|z)$ estimates, the *WeightedNB* classifier.

$$Pr(w|z) = \sum_{d \in \mathcal{D}} Pr(w,d|z) = \sum_{d \in \mathcal{D}} Pr(w|d) Pr(d|z) \quad (14)$$

To classify a completely new document d', with words $\{w_i | 1 \le i \le |d'|\}$, which are *i.i.d. random samples*, we compute for each class z', the *posterior* $Pr(z'|d')$ from the distribution $Pr(w_i|z')$, as detailed in (McCallum & Nigam, 1998). We did not perform any explicit feature smoothing, because from (14) we can infer that a dense matrix of $Pr(d,z)$ values ensures that values of $Pr(w|z)$ are smooth.

4.2. Weighted SVM

Support Vector Machines are learning algorithms that are traditionally used in two class settings to maximally separate regions of homogeneous labels, by building a hyperplane that separates the positive and the negative classes. The support vectors that span up the hyperplane can then be used to classify the test data points based on which side and how far they lie from the hyperplane separating the two classes. Most SVM implementations use cost-based optimization techniques and provide handles to specify the various costs associated with misclassifying the data points. One such MATLAB-based implementation (Schwaighofer, 2002) allows the user to associate an

upper bound cost for misclassifying any data point, upper bound cost for misclassifying positive and/or negative data-points and cost for misclassifying each of the data-points. The learning algorithm tries to minimize the overall misclassification cost.

We use (Schwaighofer, 2002) to utilize the estimate of $Pr(d)$ provided by BayesANIL as the cost of misclassifying the document d and call the resulting classifier *WeightedSVM*. The intuition behind this setting is that we require the SVM learner not to mis-classify the labeled examples that we have more trust in. In fact, Our observations recorded in Section 5 show that $Pr(d)$ reflects the probability that a training document d is correctly labeled.

5. Experiments and Results

We ran experiments on two standard data-sets, to demonstrate that BayesANIL is adept in facilitating better performance of two text classifiers, *viz.*, naïve Bayes and SVM. We created scenarios to demonstrate learnability in the presence of noise and in a labeled-unlabeled setting. We also provide insights into the performance of our method, particularly the effect of the smoothing parameter λ on classification accuracy.

5.1. Data Sets

We used some standard text classification data-sets to evaluate our model. We depend on rainbow (McCallum, 1996) to parse, tokenize, index and print the sparse matrix representation of the documents. We chose not to perform any feature selection such as stop-word removal, de-tagging, pruning by info-gain, stemming, *etc.*

The **WebKB** data-set[2] contains Web pages gathered from the computer science departments of several US universities. As in (Nigam et al., 1999), we chose the four most populous and popular categories: *student, faculty, course* and *project*, altogether consisting of 4,199 documents. We removed the server headers before tokenizing the documents and found that rainbow's tokenizer reports $54,948$ unique words in this data-set.

The **Newsgroups**[2] data set contains 19,997 articles evenly divided among twenty UseNet discussion groups. While tokenizing, we discarded the subject and newsgroup category along with the complete UseNet header. The total number of unique words for this data set was $111,935$.

[2] http://www-2.cs.cmu.edu/~TextLearning/datasets.html

5.2. Experiments

5.2.1. Supervised Learning

We used BayesANIL along with standard classification methods, naïve Bayes and SVM, in order to label previously unseen test documents. The results on Newsgroups and WebKB datasets are as tabulated in Table 1. The accuracies (in percentage) reported are averages over 20 random splits of the data-sets into training and testing parts in the ratio 60%:40%. The stopping criterion for the EM iteration was based on convergence of log-likelihood as described in Section 3.3. Laplace smoothing was used in the plain naïve Bayes classifier and SVM was used with a linear kernel. Error correcting output code was used to handle multi-classes in SVM. The weightedNB and weightedSVM were set up as described in Section 4.

Dataset	NB	Weighted NB	SVM	Weighted SVM
WebKB	80.51	86.2	83.8	86.47
Newsgroup	77.65	84.8	79.8	85.42

Table 1. Accuracies (in %) on two commonly-used data sets by vanilla naïve Bayes and SVM against versions of the respective classifiers weighed with BayesANIL.

5.2.2. Noisy Labels

In order to show the ability of BayesANIL to learn in the presence of noise, we used the setup similar to the one in (Lee & Liu, 2003). We used the 20 Newsgroups data-set in one class vs. the rest setting. We held out a random sample of 50% documents for training, 20% for validation and 30% for testing. We introduced noise artificially by mutating the labels of a random subset of documents in the positive class of the training and validation sets. We experimented by varying this noise parameter α ($\alpha = 0$, 0.3 and 0.7). BayesANIL was used to learn the degree to which each of the training documents belongs to the positive and negative classes. We set the λ parameter in our EM algorithm to α to aid the correction of noise in the initial empirical distribution of the features. The documents in the validation set were used to determine the stopping criterion for the EM runs. Finally, we built a SVM classifier over the training data, using the class labels predicted by BayesANIL, instead of the original noisy class labels. We set up the weightedSVM as described in the Section 4. Table 2 shows the F measure for the classification of test documents by the weightedSVM. The results convincingly exceed the ones reported in (Lee & Liu, 2003).

Positive class	$\alpha = 0$	$\alpha = 0.3$	$\alpha = 0.7$
atheism	0.782	0.779	0.739
autos	0.840	0.832	0.806
space	0.891	0.890	0.865
graphics	0.698	0.656	0.648
motorcycles	0.915	0.927	0.880
christian	0.833	0.855	0.829
ms-windows	0.717	0.728	0.683
baseball	0.904	0.883	0.879
guns	0.796	0.814	0.771
pc	0.728	0.722	0.687
hockey	0.921	0.913	0.840
mideast	0.862	0.848	0.847
mac	0.839	0.855	0.799
crypt	0.894	0.902	0.862
politics	0.666	0.661	0.623
xwindows	0.802	0.788	0.764
electronics	0.727	0.731	0.714
religion	0.590	0.588	0.545
forsale	0.745	0.729	0.679
med	0.913	0.909	0.858
Average	0.803	0.801	0.766

Table 2. F scores, using BayesANIL in conjunction with SVM, with rate of mislabeled positive class being α

5.2.3. Access to unlabeled examples

In a separate setting, we use our model for classification, given labeled and unlabeled documents. In accordance with the experimental setup described in (Nigam et al., 2000), we selected the latest (by date) 20% documents from each class of Newsgroups for testing. In the case of WebKB, web pages from *utexas.edu* were held out for testing. For each of the data-sets, we randomly sampled 1% of the total from the remaining collection to form the labeled training set. The unlabeled document collection was constructed by randomly sampling from the remaining document collection in each case. Figures 4 and 5 show variations in the testing accuracies with increasing amount of unlabeled data and with two different values of the smoothing parameter λ for the WebKB and Newsgroups data-sets respectively. It is evident that increasing the amount of unlabeled data helps the classifier in each case. Also, having a non-zero value of smoothing parameter λ for the empirical distribution improves the accuracy over a value of $\lambda = 0$.

Figure 6 shows the plot of various documents (training/unlabeled) and their probabilities, after a fixed number of EM iterations (10), on the WebKB and Newsgroup data-sets. For training, we used 10% of the documents at random as the labeled set and retained the rest as unlabeled. We also introduced noise by mutating 30% of the class labels of the labeled data. After 10 EM iterations, we labeled each document instance d with $z = argmax_z Pr(z, d)$. Let z' be the original label of the document d. In the plot legends, "correctly labeled" means $argmax_z Pr(z, d) = z'$. Similarly, "incorrectly labeled" means $argmax_z Pr(z, d) \neq z'$. An interesting observation in these plots is that correctly labeled training documents have the highest $Pr(d)$, followed by the incorrectly labeled training documents and then by the correctly labeled documents from the unlabeled set. The incorrectly labeled document from the unlabeled set have the least $Pr(d)$. This is the reason why using SVM weighed by BayesANIL gives more accurate results than plain SVM. Setting the cost of misclassifying an example d to to $Pr(d)$ forces the SVM learner to generate the least mis-classifications on the most confidently labeled examples.

6. Summary and Conclusions

We proposed a Bayesian model that accounts for uncertainty incurred by the manual labeling process causing approximate, noisy and incomplete labelings. We showed that this model could be used with standard learners such as naïve Bayes and SVM to significantly improve the accuracy of text classification. Further, our results improve over a state-of-the-art result reported in the literature of classification in the presence of noise. This method when used in a labeled/unlabeled setting, can learn from features in the unlabeled data that were unseen in the labeled data. BayesANIL provides both conditional probabilities $Pr(z|d)$ and *a priori* probabilities $Pr(d)$, which can be used in the setting of active labeling (given a set of unlabeled documents). One could consider these probabilities as a measure of confidence and support, respectively, for the label assigned to a given document. In the future, we plan to extend the implementation of our model to links modeled as a set of inward-linking and outward-linking neighbors, and other hypertext features.

Acknowledgments

We would like to use this opportunity to thank Sachindra Joshi for the discussions while framing the problem solved in this paper and Sreeram Balakrishnan for his comments on the writeup.

References

Amari, S.-I. (1995). Information geometry of the EM and em algorithms for neural networks. *Neural Net-*

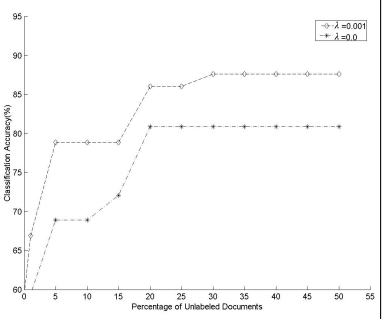

Figure 4. Plot for WebKB of accuracy on 20% test data as a function of % unlabeled data. 1% of the data was kept as labeled training data

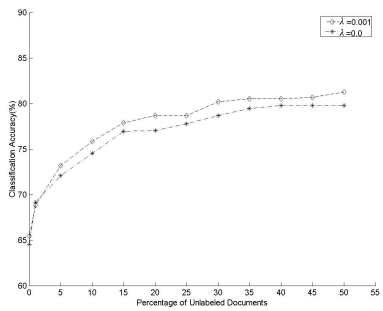

Figure 5. Plot for 20 NewsGroups of accuracy on 20% test data as a function of % unlabeled data. 1% of the data was kept as labeled training data

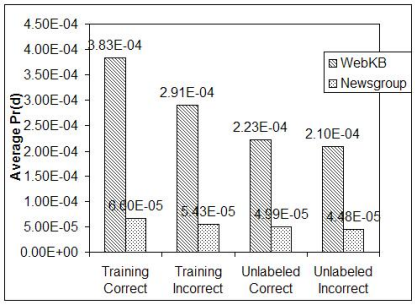

Figure 6. The average $Pr(d)$ plotted against documents in WebKB and Newsgroups. Training/unlabeled documents with correctly and incorrectly predicted labels are shown.

works, *8*, 1379–1408.

Ando, R. K., & Zhang, T. (2004). A framework for learning predictive structures from multiple tasks and unlabeled data. *Technical Report RC23462, IBM T.J. Watson Research Center*.

Brodley, C. E., & Friedl, M. A. (1996). Identifying and eliminating mislabeled training instances. *Proceedings of the 13th National Conference on Artificial Intelligence* (pp. 799–805). Portland, Oregon.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1976). Maximum likelihood from incomplete data via the EM algorithm. *Proceedings of the Royal Statistical Society*, 1–38.

Domingos, P. (1999). Metacost: A general method for making classifiers cost-sensitive. *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 155–164). New York, NY, US.

Griffiths, T. L., & Tenenbaum, J. B. (2001). Exploiting weak prior knowledge in bayesian parameter estimation. *Proceedings of NIPS*.

Hofmann, T. (1999). Probabilistic latent semantic indexing. *Proceedings of the 22nd International ACM SIGIR conference on Research and Development in Information Retrieval* (pp. 50–57). Berkeley, US.

Joachims, T. (1998). Text categorization with support vector machines: learning with many relevant features. *Proceedings of 10th European Conference on Machine Learning* (pp. 137–142). Chemnitz, DE: Springer Verlag, Heidelberg, DE.

Lee, W. S., & Liu, B. (2003). Learning with positive and unlabeled examples using weighted logistic regression. *Proceedings of the 20th International Conference on Machine Learning* (pp. 448–455). Washington DC, US.

Lewis, D. (1998). Naive (bayes) at forty: The independence assumption in information retrieval. *Proceedings of 10th European Conference on Machine Learning* (pp. 137–142).

McCallum, A., & Nigam, K. (1998). A comparison of event models for naive bayes text classification. *Proceedings of AAAI-98 Workshop on Learning for Text Categorization*.

McCallum, A. K. (1996). Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/˜mccallum/bow.

Nigam, K., Lafferty, J., & McCallum, A. (1999). Using maximum entropy for text classification. *Proceedings of Workshop on Machine Learning for Information Filtering, IJCAI*, 61–67.

Nigam, K., McCallum, A. K., Thrun, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning.*, *39*, 103–134.

Schwaighofer, A. (2002). SVM toolbox for matlab. http://www.igi.tugraz.at/aschwaig/software.html.

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Information Retrieval*, *1*, 69–90.