Multi-Label Text Classification with a Mixture Model Trained by EM

Andrew Kachites McCallum
Just Research
Pittsburgh, PA 15213
mccallum@justresearch.com

Abstract

In many important document classification tasks, documents may each be associated with multiple class labels. This paper describes a Bayesian classification approach in which the multiple classes that comprise a document are represented by a mixture model. While the labeled training data indicates which classes were responsible for generating a document, it does not indicate which class was responsible for generating each word. Thus we use EM to fill in this missing value, learning both the distribution over mixture weights and the word distribution in each class's mixture component. We describe the benefits of this model and present preliminary results with the Reuters-21578 data set.

1 Introduction

Text classification is the problem of assigning a text document into one or more topic categories or classes. In multiclass document classification, as distinguished from binary document classification, there are more than two classes. In multi-label classification each document may have more than one class label. For example, given classes N. America, S. America, Europe, Asia and Australia, a news article about U.S. troops in Bosnia may be labeled with both the N. America and Europe classes.

This paper describes a Bayesian approach to multiclass, multi-label document classification. We define a probabilistic generative model that represents the multi-label nature of a document by indicating that the words in a document are produced by a mixture of word distributions, one for each topic. The generative process begins by selecting the set of classes (instead of a single class) that will be the labels for this document; then producing a set of mixture weights for those classes; finally, each word in the document is generated by first selecting a class according to these mixture weights, then letting that class generate a single word. Classification uses Bayes rule and selects the class set that is most likely given the document.

Parameters of the model are learned by maximum a posteriori estimation from labeled training data. The labels indicate which classes were involved in generating the document, but do not indicate which class generated which individual words. Thus we use Expectation-Maximization (EM) to fill in this missing value, avoiding over-fitting by performing the E-step in leave-one-document-out fashion. EM sets both the mixture weights and the word distributions in each class. We also add an extra class to which all documents belong: the "English" class. This naturally captures background language in common to all documents—in essence automatically extracting a task-specific stop list.

There are two common alternatives to this approach. The first is to build a binary classifier for each class (*e.g.* [Yang, 1998; Joachims, 1998; Nigam *et al.*, 1999]). The second is to build a mapping from class-document pairs to real-valued scores, and use the scores to provide a ranking of classes—the top most of which can be assigned to the document by choosing a threshold (*e.g.* [Schapire and Singer, 1999]).

The key features of the approach in this paper are four-fold: **(1)** We conjecture that our mixture model can represent more expressive decision boundaries than can the binary classifier. In other words, the binary classifier's large and multi-faceted negative class is not well-represented by a single word distribution model. Our approach models "other" classes with a mixture of their own individual models. **(2)** Note that binary classifiers tacitly assume that labels can be assigned independently. When one label provides information about another, the binary classifier fails to capture this. Our generative model can represent the correlations between class labels. **(3)** There are no thresholds to be tuned or learned. Threshold learning can typically be quite problematic because document length and other attributes affect the score, making it difficult to relate scores across different documents. **(4)** We have the advantages of a formal probabilistic approach, with a well-defined generative model, making available future enhancements based on the large tool chest of powerful statistical parameter estimation techniques, (such as shrinkage and unlabeled data [McCallum *et al.*, 1998; Nigam *et al.*, 1999]).

This paper presents preliminary experimental results on a subset of the Reuters-21578 data set. We find that the mixture model outperforms the approach based on a collection of binary classifiers, reducing classification error on almost all labels, reducing error by more than 50% on one label, and reducing overall error by one third. Further experiments are needed to fully evaluate the technique.

2 Multi-label Classification with a Mixture Model

The Generative Model. The generative model contains a set of classes, $\mathcal{C} = \{c_1, c_2, ...c_n\}$. Each document is associated with some subset of these classes, which can be thought of as a binary bit-vector, $\vec{c}$, with ones indicating the classes in the subset. The power set of all possible class subsets is written $\mathcal{C}^\star$. Associated with each individual class, c_j is a word distribution $\mathrm{P}(w_k|c_j)$, for all words in the vocabulary, $\mathcal{V} = \{w_1, w_2, ...w_m\}$. Given a set of classes, each document is generated by a mixture of these word distributions, with mixture weights $\vec{\lambda}$, where components of $\vec{\lambda}$ associated with classes not in $\vec{c}$ are forced to be zero. The mixture weights themselves are selected from a distribution over mixture weights written $\mathrm{P}(\vec{\lambda}|\vec{c})$. Labeled training data consists of a set of documents $\mathcal{D} = \{d_1, d_2, ...d_N\}$. The vector of class labels associated with document d_i is written $\vec{l_i}$.

The generative process for a document is as follows. First select a set of classes from distribution $\mathrm{P}(\vec{c})$. Then select mixture weights among these classes according to $\mathrm{P}(\vec{\lambda}|\vec{c})$. Then start generating words in the document. For each word select a class according the mixture weights in $\vec{\lambda}$, (the components of which are written λ_c, which can be thought of as $\mathrm{P}(c|\vec{\lambda})$), then let that class generate a single word according to its multinomial word probability distribution $\mathrm{P}(w_k|c_j)$. Thus the probability of a document is $\mathrm{P}(d) = \sum_{\vec{c} \in \mathcal{C}^*} \int d\vec{\lambda}\, \mathrm{P}(\vec{\lambda}|\vec{c}) \prod_{w \in d} \sum_{c \in \mathcal{C}} \vec{\lambda}_c \mathrm{P}(w|c)$.

Once this generative model is defined, multi-label Bayesian classification falls out naturally. Given a test document we wish to select the set of classes that is most probable: $\vec{c}^{+} = \arg\max_{\vec{c}} \mathrm{P}(\vec{c}|d)$. To express $\mathrm{P}(\vec{c}|d)$ in terms of the generative model parameters, we begin by applying Bayes' rule: $\mathrm{P}(\vec{c}|d) = \mathrm{P}(\vec{c})\mathrm{P}(d|\vec{c})/\mathrm{P}(d)$. The generative process for the document d is not defined in terms of $\vec{c}$ only, but requires the mixture weights. We introduce the mixture weights by a sum of total probability, (and also drop the $\mathrm{P}(d)$, which is just a normalization factor that makes the $\mathrm{P}(\vec{c}|d)$'s sum to one):

$$\mathrm{P}(\vec{c}|d) \propto \mathrm{P}(\vec{c}) \int d\vec{\lambda}\, \mathrm{P}(\vec{\lambda}|\vec{c})\mathrm{P}(d|\vec{\lambda}, \vec{c}). \tag{1}$$

This is an integral over an infinite number of possible real-valued vectors $\vec{\lambda}$. Finding the closed-form solution to this integral is the subject of future work. For now, we approximate the integral using the single a priori most probable $\vec{\lambda}$.[1] (See the M-step below.) Let $\vec{\lambda}^{(\vec{c})}$ be the single most likely mixture weights given the set of classes $\vec{c}$: $\vec{\lambda}^{(\vec{c})} \equiv \arg\max_{\lambda} \mathrm{P}(\lambda|\vec{c})$, then we approximate Equation 1 by $\mathrm{P}(\vec{c}|d) \approx \mathrm{P}(\vec{c})\mathrm{P}(d|\vec{\lambda}^{(\vec{c})}, \vec{c})$. Assuming word independence, and then expanding the mixture model gives: $\mathrm{P}(\vec{c}|d) \approx \mathrm{P}(\vec{c}) \prod_{w \in d} \sum_{c \in \mathcal{C}} \vec{\lambda}_c^{(\vec{c})} \mathrm{P}(w|c)$.

Classification is performed by calculating the posterior probability of each class set $\vec{c}$ and selecting the set that is most probable. When the size of the class set is allowed to be large, there will be too many sets to evaluate exhaustively, and efficient yet accurate search will be necessary. Here are three methods for efficient exploration of class set space: **(1)** Begin by restricting $\vec{c}$ to size one, and finding $\arg\max_{\vec{c}} \mathrm{P}(\vec{c}|d)$, then restrict $\vec{c}$ to size two in which one of the classes must be the class selected in the first step. Continue this process, greedily adding classes until all have been added, and then select the combination that is most likely. **(2)** Find the mixture weights λ that are most likely to have produced the document (described as the E-step below), and then evaluate $\mathrm{P}(\vec{c}|d)$ for those sets resulting from adding classes one at a time in order of highest mixture weight. (Note that this is analogous to automatically determining a threshold for the second method described in the introduction.) For both of the methods above, additional accuracy may be obtained by performing a limited amount of exploration around the greedy paths. **(3)** Explore by simulated

[1] Using the single a priori most probable mixture weights for each class set is one solution, however early experimental evidence indicates that quite different mixture weights are most likely for different documents labeled with the same class set, thus indicating that this uni-modal approximation could be improved. An interesting alternative to the closed form solution is a non-parametric, "memory-based" representation for $\mathrm{P}(\vec{\lambda}|\vec{c})$: Instead of summing over all possible mixture weights $\vec{\lambda}$, sum over the most likely mixture weights associated with each of the training documents labeled with class set $\vec{c}$. This could be even better than a closed-form solution when the parametric form of of $\mathrm{P}(\vec{\lambda}|\vec{c})$ (such as Dirichlet) is a poor match for the data.

annealing or other similar stochastic optimization techniques. Each of these three methods have different biases. This paper uses both (1) and (2) concurrently.

Parameter Estimation. We determine parameters by maximum a posteriori estimation from a collection of labeled training data. The class set prior distribution is currently represented in completely unfactored form and is set by maximum a posteriori estimation with a uniform prior: $\mathrm{P}(\vec{c}) = (m + N(d, \vec{c}))/(m|\mathcal{C}^{\star}| + |\mathcal{D}|)$, where $N(d, \vec{c})$ is the number of labeled training documents labeled with exactly the vector $\vec{c}$, and m is a smoothing parameter determining the strength of the uniform prior. Setting $m = 1$ results in Laplace smoothing.

All other parameters cannot be estimated directly from training data since all aspects of the generative process are not provided by the given labels. Specifically, the labels on the training document indicate which classes were involved in generating each document, but not which classes were responsible for generating each individual word. Thus we turn to Expectation-Maximization (EM) [Dempster *et al.*, 1977] in order to fill in this missing value. In the E-step we estimate for each word in a training document which class was responsible for generating it. In the M-step we use these estimates to straightforwardly determine maximum a posteriori or maximum likelihood estimates of the most likely mixture weight for each class ($\vec{\lambda}^{(\vec{c})}$) and per-class word probability distributions ($\mathrm{P}(w|c)$):

E-step

$$\mathrm{P}(c|w \in d_i) = \frac{\vec{\lambda}_c^{(\vec{l_i})}\mathrm{P}(w|c)}{\sum_{c'} \vec{\lambda}_{c'}^{(\vec{l_i})}\mathrm{P}(w|c')}$$

M-step

$$\vec{\lambda}_c^{(\vec{c})} = \frac{1+\sum_{d_i \in \vec{c}} N(w,d_i)\mathrm{P}(c|w \in d_i)}{|\mathcal{C}|+\sum_{c' \in \vec{c}} \sum_{d_i \in \vec{c}} N(w,d_i)\mathrm{P}(c'|w \in d_i)}$$

$$\mathrm{P}(w_i|c) = \frac{\sum_{d \in \mathcal{D}} N(w_i,d)\mathrm{P}(c|w_i)}{\sum_{w' \in \mathcal{V}} \sum_{d \in \mathcal{D}} N(w',d)\mathrm{P}(c|w')}$$

where $N(w, d)$ is the count of w occurs in d. If there are no training documents for a particular set of classes, then we backoff to the average mixture weights of class sets in the training data, where the average is weighted by class set overlap.

Additional Features. To guard against over-fitting, the E-step above is performed in leave-one-out fashion. That is, when estimating which classes generated the words in a particular training document, the influence of that document is temporarily removed from the parameter estimates used to perform the E-step. Because the parameters are based on (probabilistically weighted) counts, this removal is quite easy. The leave-one-out E-step is also used in deleted interpolation [Jelinek and Mercer, 1980] and in shrinkage for document classification [McCallum *et al.*, 1998].

In addition to the classes in $\mathcal{C}$, we also add an extra class to which all documents belong. This can be thought of as the "English" class. Due to the leave-one-out E-step, this class gathers the words that are common to all classes—in essence automatically finding the task-specific stop list.

Furthermore, the uniform word distribution is also added as another mixture component. In conjunction with the leave-one-out E-step, this serves to automatically determine the "optimal" amount of smoothing with uniform distribution, in the sense that it maximizes the probability of all the training data in leave-one-out fashion. For example, a word that occurs only once in the training data will have a leave-one-out probability of zero in all mixtures except the uniform distribution; its mixture weight λ will thus be increased, properly set to maximize the probability of the left-out document.

grain	wheat	corn	ship	trade	crude	*root*
$d^\star$	$d^\star$	$d^\star$	$d^\star$	trade	oil	$d^\star$
grain	wheat	corn	gulf	billion	$d^\star$	mln
tonnes	tonnes	tonnes	ships	$d^\star$	crude	dlrs
mln	mln	mln	strike	japan	opec	march
agriculture	export	maize	port	deficit	prices	pct
year	agriculture	pct	shipping	year	bpd	year
usda	offer	usda	seamen	japanese	barrels	billion
crop	department	acres	march	surplus	mln	april
farmers	soviet	year	union	exports	production	blah
farm	march	production	iran	countries	pct	company

Table 1: The ten most probable words in multinomial word distributions of several topic classes from the ten used in the Reuters-21578 data set. The notation $d^\star$ refers to a string of digits. Note that the model has correctly placed the word "blah" in the *root* word distribution; in a small percentage of the documents across all classes, it is the sole word in the document body.

3 Experimental Results

The Reuters 21578 Distribution 1.0 was used to evaluate the performance of the multi-label mixture model. As in several other studies, only the ten most populous classes were used (*e.g.* [Joachims, 1998; Nigam *et al.*, 1999]). The text was tokenized by removing SGML tags, gathering strings alphabetic characters, downcasing them, and removing words on the SMART stoplist. Additionally all strings of digits were mapped to a single common token. The multi-label mixture model was trained with the ModApte training set. EM was run to convergence, which took ten iterations and about 4 minutes of wall clock time.

The performance of this model is compared with the traditional method consisting of a collection of binary classifiers, one for each class, as used in several other studies [Yang, 1998; Joachims, 1998; Nigam *et al.*, 1999]. Each binary classifier is a multinomial naive Bayes with Laplace smoothing [McCallum and Nigam, 1998].

The results of EM estimation for the word probability distribution in each of the classes are quite interesting. As expected, words associated with the different class topics gather in the appropriate classes. Even classes that never occur as the sole label for a document, such as wheat, tend to gather those words that are in common among the wheat documents, excluding words that are better explained by other topics. Several word distributions are given in Table 1. Notice that some high-probability words still overlap among classes, such as "tonnes" in in grain, wheat and corn. This suggests that a hierarchical or directed acyclic graph structure among classes could capture some of this redundancy and perhaps aid generalization; see the discussion of future work in the last section of this paper.

A comparison of classification accuracy is shown in Table 2. This paper's multi-label mixture model outperforms the traditional collection of binary classifiers for all classes except one (ship), sometimes reducing error by more than half (*e.g.* wheat). An unfair advantage is given to the traditional binary classifier in that we select the vocabulary size with which it does best on the test set. The mixture model involves no such tuning. When credit is given only for class sets that are an exact match with label set (full set), the multi-label mixture model reduces error by one third.

class	binary NB classifier accuracy (vocab size)	multi-label mixture accuracy
acq	0.9707 (23433)	0.9783
corn	0.9751 (20)	0.9834
crude	0.9751 (23433)	0.9807
earn	0.9718 (2000)	0.9823
grain	0.9627 (20)	0.9795
interest	0.9639 (23433)	0.9827
money	0.9536 (23433)	0.9689
ship	0.9867 (23433)	0.9815
trade	0.9524 (20)	0.9654
wheat	0.9618 (23433)	0.9811
full set	0.7569	0.8392

Table 2: Classification accuracy on the ten largest classes in Reuters-21578. The multi-label mixture model sometimes reduces error by half (wheat). When credit is given only for class sets that are an exact match with label set (last line), the multi-label mixture model reduces error by one third.

In preliminary results on the entire set of 90 Reuters classes, our mixture model performs better than the binary naive Bayes classifier on classes with a moderate and small number of training documents. On those classes with extremely large training sets recall is higher, but precision is lower. Over all classes, macroaverage F_1 is improved by 7 points; microaverage F_1 is the same [Yang, 1998]. Analysis suggests that a more factored class set prior $\mathrm{P}(\vec{c})$ may improve precision.

In future work we will evaluate the methods on richer data sets, where we expect our leave-one-out EM to shine (unlike Reuters where several classes, such as corn and wheat can be predicted by a single word). We will also compare with Support Vector Machines [Joachims, 1998] and Boosting [Schapire and Singer, 1999]. In relation to these methods, we expect our model to have advantages of computational efficiency and robustness to sparse data through shrinkage (see Section 4).

4 Related Work and Conclusions

Various related work has used mixture models for modeling text. The Nymble system from BBN uses an HMM with uniform transition probabilities to perform information extraction of "named entities" [Bikel *et al.*, 1997]. Similar models have been used for text segmentation and topic tracking [Yamron *et al.*, 1998]. The most related work is that of Imai *et al.* [1997], in which a mixture model is used for multi-label document classification. However, their work does not have a class-set conditional mixture distribution, does not use leave-one-out evaluation in the E-step, does not do "optimal smoothing" by mixing with the uniform distribution, and classifies test documents by searching the space of class sets using a method based on a heuristic threshold. BoosTexter is a collection of enhancements to AdaBoost that enable its application to multiclass, multi-label document classification problems [Schapire and Singer, 1999]. Unlike traditional Boosting, the labels as well as the training instances are re-weighted. BoosTexter aims to predict all the correct labels by ranking them so that the correct labels receive the highest ranks. Their work does not evaluate any method of tuning the threshold on the ranks. The mixture model described in this paper is similar to the "aspect" clustering model described by Hofmann and Puzicha [1998] where a document is also generated by a mixture of per-topic models. Instead of supervised classification, their model is used for unsupervised clustering.

In future work we will combine this model with one that uses shrinakge to take advantage of a hierarchy of topics [McCallum *et al.*, 1998], or a directed acyclic graph of topics. This model also lends itself to improving parameter estimates by incorporating unlabeled data, as in the work by Nigam *et al.* [1999]. We will also explore more complex functions and priors for the $\mathrm{P}(\vec{c})$ and $\mathrm{P}(\vec{\lambda}|\vec{c})$ functions. The model may additionally be improved by using multiple mixture components for each topic, the word distribution of each learned in a completely unsupervised fashion.

References

[Bikel *et al.*, 1997] Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. Nymble: a high-performance learning name-finder. In *Proceedings of ANLP-97*, pages 194–201, 1997.

[Dempster *et al.*, 1977] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(B):1–38, 1977.

[Hofmann and Puzicha, 1998] T. Hofmann and J. Puzicha. Statistical models for co-occurrence data. Technical Report AI Memo 1625, AI Lab, MIT, 1998.

[Imai *et al.*, 1997] Toru Imai, Richard Schwartz, Francis Kubala, and Long Nguyen. Improved topic discrimination of broadcast news using a model of multiple simultaneous topics. In *Proceedings of the IEEE ICASSP*, volume 2, pages 727–730, Munich, Germany, April 1997.

[Jelinek and Mercer, 1980] Fred Jelinek and Robert Mercer. Interpolated estimation of Markov source parameters from sparse data. In S. Gelsema and L. N. Kanal, editors, *Pattern Recognition in Practice*, pages 381–402, 1980.

[Joachims, 1998] Thorsten Joachims. Text categorization with Support Vector Machines: Learning with many relevant features. In *Machine Learning: ECML-98, Tenth European Conference on Machine Lea rning*, pages 137–142, 1998.

[McCallum and Nigam, 1998] Andrew McCallum and Kamal Nigam. A comparison of event models for naive Bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998. http://www.cs.cmu.edu/∼mccallum.

[McCallum *et al.*, 1998] Andrew McCallum, Ronald Rosenfeld, Tom Mitchell, and Andrew Ng. Improving text clasification by shrinkage in a hierarchy of classes. In *International Conference on Machine Learning (ICML)*, pages 359–367, 1998.

[Nigam *et al.*, 1999] Kamal Nigam, Andrew McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 1999. To appear.

[Schapire and Singer, 1999] Robert E. Schapire and Yoram Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning Journal*, 1999.

[Yamron *et al.*, 1998] J.P. Yamron, I. Carp, L. Gillick, S. Lowe, and P. van Mulbregt. A hidden Markov model approach to text segmentation and event tracking. In *Proceedings of the IEEE ICASSP*, Seattle, Washington, May 1998.

[Yang, 1998] Yiming Yang. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, 1998. To appear.