Efficient Bayesian Task-Level Transfer Learning

Daniel M. Roy and **Leslie P. Kaelbling**
Massachusetts Institute of Technology
Computer Science and Artificial Intelligence Laboratory
{droy, lpk}@csail.mit.edu

Abstract

In this paper, we show how using the Dirichlet Process mixture model as a generative model of data sets provides a simple and effective method for transfer learning. In particular, we present a hierarchical extension of the classic Naive Bayes classifier that couples multiple Naive Bayes classifiers by placing a Dirichlet Process prior over their parameters and show how recent advances in approximate inference in the Dirichlet Process mixture model enable efficient inference. We evaluate the resulting model in a meeting domain, in which the system decides, based on a learned model of the user's behavior, whether to accept or reject the request on his or her behalf. The extended model outperforms the standard Naive Bayes model by using data from other users to influence its predictions.

1 Introduction

In machine learning, we are often confronted with multiple, related data sets and asked to make predictions. For example, in spam filtering, a typical data set consists of thousands of labeled emails belonging to a collection of users. In this sense, we have multiple data sets—one for each user. Should we combine the data sets and ignore the prior knowledge that different users labeled each email? If we combine the data from a group of users who roughly agree on the definition of spam, we will have increased the available training data from which to make predictions. However, if the preferences within a population of users are heterogeneous, then we should expect that simply collapsing the data into an undifferentiated collection will make our predictions worse.

The process of using data from unrelated or partially related tasks is known as *transfer learning* or *multi-task learning* and has a growing literature (Baxter, 2000; Guestrin et al., 2003; Thrun, 1996; Xue et al., 2005). While humans effortlessly use experience from related tasks to improve their performance at novel tasks, machines must be given precise instructions on how to make such connections. In this paper, we introduce such a set of instructions, based on the statistical assumption that there exists some partition of the tasks into clusters such that the data for all tasks in a cluster are identically distributed. Ultimately, any such model of sharing must be evaluated on real data, and, to that end, we evaluate the resulting model in a meeting domain. The learned system decides, based on training data for a user, whether to accept or reject the request on his or her behalf. The model that shares data outperforms its no-sharing counterpart by using data from other users to influence its predictions.

When faced with a classification task on a single data set, well-studied techniques abound (Boser et al., 1992; Lafferty et al., 2001). A popular classifier that works well in practice, despite its simplicity, is the Naive Bayes classifier (Maron, 1961). We can extend this classifier to the multi-task setting by training one classifier for each cluster in the latent partition. To handle uncertainty in the number of clusters and their membership, we define a generative process for data sets that induces clustering. At the heart of this process is a non-parametric prior known as the Dirichlet Process. This prior couples the parameters of the Naive Bayes classifiers attached to each data set. This approach extends the applicability of the Naive Bayes classifier to the domain of multi-task learning when the tasks are defined over the same input space.

Bayesian inference under this *clustered* Naive Bayes model combines the contribution of every partition of the data sets, weighing each by the partition's posterior probability. However, the sum over partitions is intractable and, therefore, we employ recent work by Heller and Ghahramani (2005a) to implement an approximate inference algorithm. The result is efficient, task-level transfer learning.

2 Models

In this paper, we concentrate on classification settings where the features X_f, $f = 1, \ldots, F$ and labels Y are drawn from finite sets $\mathcal{V}_f, \mathcal{L}$, respectively. Our goal is to learn the relationship between the input features and output labels in order to predict a label given an unseen combination of features.

Consider U tasks, each with N_u training examples composed of a label Y and a feature vector X. To make the discussion more concrete, assume each task is associated with a different user performing some common task. Therefore, we will write $D^{(u)}$ to represent the feature vectors $X^{(u,i)}$ and corresponding labels $Y^{(u,i)}$, $i = 1, \ldots, N_u$, associated with the u-th user. Then D is the entire collection of data across all users, and $D^{(u,j)} = (X^{(u,j)}, Y^{(u,j)})$ is the j-th data point for the u-th user.

Let $m_{u,y}$ denote the number of data points labeled $y \in \mathcal{L}$ in the data set associated with the u-th user and let $n_{u,y,f,x}$ denote the number of instances in the u-th data set where the f-th feature takes the value $x \in \mathcal{V}_f$ when its parent label takes value $y \in \mathcal{L}$. Let $\theta_{u,y,f,x} \triangleq \mathbb{P}(X_f^{(u)} = x \mid Y^{(u)} = y)$, and $\phi_{u,y} \triangleq \mathbb{P}(Y^{(u)} = y)$. We can now write the general form of the probability mass function of the data conditioned on the parameters θ and ϕ of the Naive Bayes model:

$$p(D|\theta,\phi) \overset{(a)}{=} \prod_{u=1}^{U} p(D^{(u)}|\theta_u,\phi_u) \overset{(b)}{=} \prod_{\substack{1\le u\le U\\ 1\le n\le N_u}} p(D^{(u,n)}|\theta_u,\phi_u)$$

$$\overset{(c)}{=} \prod_{u=1}^{U}\prod_{n=1}^{N_u} p(Y^{(u,n)}|\phi_{u,y}) \prod_{f=1}^{F} p(X_f^{(u,n)}|Y^{(u)},\theta_{u,y,f})$$

$$\overset{(d)}{=} \prod_{u=1}^{U}\prod_{y\in\mathcal{L}} \phi_{u,y}^{m_{u,y}} \prod_{f=1}^{F}\prod_{x\in\mathcal{V}_f} \theta_{u,y,f,x}^{n_{u,y,f,x}},$$

In (a), $p(D|\theta,\phi)$ is expanded into a product of terms, one for each data set, reflecting that the data sets are independent conditioned on the parameterization. Step (b) assumes the data points are exchangeable; in particular, the label/feature pairs are independent of one another given the parameterization. In step (c), we have made use of the Naive Bayes assumption that the features are conditionally independent given the label. Finally, in step (d), we have used the fact that the distributions are multinomials. The maximum likelihood parameterizations are

$$\phi_{u,y} = \frac{m_{u,y}}{\sum_{y\in\mathcal{L}} m_{u,y}}, \quad \theta_{u,y,f,x} = \frac{n_{u,y,f,x}}{\sum_{x\in\mathcal{V}_f} n_{u,y,f,x}}.$$

Because each data set is parameterized separately, it is no surprise that the maximum likelihood parameterization for each data set depends only on the data in that data set. In order to induce sharing, we must somehow constrain the parameterization across users. In the Bayesian setting, the prior distribution $F(\theta,\phi)$ can be used to enforce such constraints. Given a prior, the resulting joint distribution on the data is

$$p(D) = \int_{\Theta\times\Phi} p(D|\theta,\phi)\, dF(\theta,\phi).$$

Both models introduced in this section are completely specified by a prior distribution over the parameters of the Naive Bayes model. As we will see, different priors result in different types of sharing.

2.1 No-Sharing Baseline Model

We have already seen that a ML parameterization of the Naive Bayes model ignores related data sets. In the Bayesian setting, any prior density f over the entire set of parameters that factors into densities for each user's parameters will result in no sharing. In particular,

$$f(\theta,\phi) = \prod_{u=1}^{U} f(\theta_u,\phi_u),$$

is equivalent to the statement that the parameters for each user are independent of the parameters for other users. Under this assumption of independence, training the entire collection of models is identical to training each model separately on its own data set. We therefore call this model the *no-sharing* model.

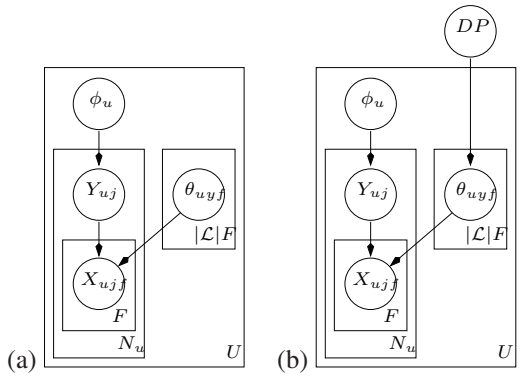

Figure 1: Graphical models for the (a) *No-Sharing* and (b) *Clustered Naive Bayes* (right) models. Each user has its own parameterization in the no-sharing model. The parameters of the Clustered Naive Bayes model are drawn from a Dirichlet Process. Here, the intermediate measure G has been integrated out.

Having specified that the prior factors into parameter distributions for each user, we must specify the actual parameter distributions for each user. A reasonable (and tractable) class of distributions over multinomial parameters are the Dirichlet distributions which are conjugate to the multinomials. Therefore, the distribution over ϕ_u, which takes values in the $|\mathcal{L}|$-simplex, is

$$f(\phi_u) = \frac{\Gamma(\sum_{y\in\mathcal{L}} \alpha_{u,y})}{\prod_{y\in\mathcal{L}} \Gamma(\alpha_{u,y})} \prod_{y\in\mathcal{L}} \phi_{u,y}^{\alpha_{u,y}-1}.$$

Similarly, the distribution over $\theta_{u,y,f}$, which takes values in $|\mathcal{V}_f|$-simplex, is

$$f(\theta_{u_y,f}) = \frac{\Gamma(\sum_{x\in\mathcal{V}_f} \beta_{u,y,f,x})}{\prod_{x\in\mathcal{V}_f} \Gamma(\beta_{u,y,f,x})} \prod_{x\in\mathcal{V}_f} \theta_{u,y,f,x}^{\beta_{u,y,f,x}-1}.$$

We can write the resulting model compactly as a generative process:[1]

$$\phi_u \sim \text{Dir}([\alpha_{u,y} : y \in \mathcal{L}]) \tag{1}$$

$$Y^{(u,n)} \mid \phi_u \sim \text{Discrete}(\phi_u)$$

$$\theta_{u,y,f} \sim \text{Dir}([\beta_{u,y,f,x} : x \in \mathcal{V}_f])$$

$$X_f^{(u,n)} \mid Y^{(u,n)}, \{\theta_{u,y,f} : \forall y \in \mathcal{L}\} \sim \text{Discrete}(\theta_{u,(Y^{(u,n)}),f})$$

The No-Sharing model will function as a baseline against which we can compare alternative models that induce sharing. We turn now to specifying a model that induces sharing.

[1]The $\sim$ symbol denotes that the variable to the left is distributed according to the distribution specified on the right. It should be noted that the Dirichlet distribution requires an ordered set of parameters. We therefore define an arbitrary ordering of the elements of $\mathcal{L}, \mathcal{V}_f$.

2.2 The Clustered Naive Bayes Model

A plausible assumption about a collection of related data sets is that they can be clustered into closely-related groups. To make this more precise, we will consider two tasks t_1 and t_2 over some space $X \times Y$ to be related if the data associated with both tasks are identically distributed. While this is a very coarse model of relatedness, it leads to improved predictive performance with limited training data.

As a first step we must specify a distribution over partitions of tasks. There are several properties we would like this distribution to have: first, we want exchangeability of tasks (users); e.g., the probability should not depend on the ordering (i.e. identity) of the tasks (users); second, we want exchangeability of clusters; e.g., the probability should not depend on the ordering/naming of the clusters; finally, we want consistency; e.g., a priori, the (hypothetical) existence of an unobserved task should not affect the probability that any group of tasks are clustered together.

The Chinese Restaurant Process (CRP) is a stochastic process that induces a distribution over partitions that satisfies all these requirements (Aldous, 1985). The following metaphor was used to define the process: imagine a restaurant with a countably infinite number of indistinguishable tables. The first customer sits at an arbitrary empty table. Subsequent customers sit at an occupied table with probability proportional to the number of customers already seated at that table and sit at an arbitrary, new table with probability proportional to a parameter $\alpha > 0$. The resulting "seating chart" partitions the customers. It can be shown that, in expectation, the number of occupied tables after n customers is $\Theta(\log n)$ (Antoniak, 1974; Navarro et al., 2006).

The tasks within each cluster of the partition will share the same parameterization. Extending our generative model, imagine that when a new user enters the restaurant and sits at a new table, they draw a complete parameterization of their Naive Bayes model from some base distribution. This parameterization is then associated with their table. If a user sits at an occupied table, they adopt the parameterization already associated with the table. Therefore, everyone at the same table uses the same rules for predicting.

This generative process corresponds with the well known Dirichlet Process mixture model (DPM) and has been used very successfully to model latent groups (Ferguson, 1973). The underlying Dirichlet process has two parameters, a mixing parameter α, which corresponds to the same parameter of the CRP, and a base distribution, from which the parameters are drawn at each new table. It is important to specify that we draw a complete parameterization of all the feature distributions, $\theta_{y,f}$, at each table. We have decided not to share the marginal distributions, ϕ, because we are most interested in knowledge relating the features and labels.

Again, we can represent the model compactly by specifying the generative process:

$$\phi_u \sim \text{Dir}([\alpha_{u,y} : y \in \mathcal{L}])$$

$$Y^{(u,n)} \mid \phi \sim \text{Discrete}(\phi_u)$$

$$G \sim \text{DP}(\alpha, \prod_{f=1}^{F} \prod_{y \in \mathcal{L}} \text{Dir}([\beta_{y,f,x} : x \in \mathcal{V}_f])$$

$$\theta_u = (\theta_{u,y,f})_{f=1,2,\ldots,F}^{y \in \mathcal{L}} \sim G$$

$$X_f^{(u,n)} \mid Y^{(u,n)}, \{\theta_{u,y,f} : \forall y \in \mathcal{L}\} \sim \text{Discrete}(\theta_{(u,Y^{(u,n)}),f})$$

The discrete measure G is a draw from the Dirichlet Process; in practice, this random variable is marginalized over. Because the tasks are being clustered, we have named this model the Clustered Naive Bayes model (and denote its distribution function over the parameters as F_{CNB}). We now explain how to use the model to make predictions given training data.

3 Approximate Inference

Given labelled training data $D^{(u)} = \{Y^{(u,i)}, X^{(u,i)}\}_{1 \le i \le N_u}$ for all tasks $u \in \{1, 2, \ldots, U\}$ and an unlabeled feature vector $X^* \triangleq X^{(v,N_v+1)}$ for some task v, we would like to compute the posterior distribution of its label $Y^* \triangleq Y^{(v,N_v+1)}$. Using Bayes rule, and ignoring normalization constants,

$$p(Y^* = y | X^*, D) \propto p(X^* | Y^* = y, D)\, p(Y^* = y, D)$$

$$\propto \int_{\Theta \times \Phi} p(D' | \theta, \phi)\, dF_{\text{CNB}}(\theta, \phi),$$

where D' is the data set where we imagine that the $(v, N_v + 1)$-th data point has label y. Therefore, Bayesian inference requires that we marginalize over the parameters, including the latent partitions of the Dirichlet process. Having chosen conjugate priors, the base distribution can be analytically marginalized. However, the sum over all partitions makes exact inference under the Dirichlet Process mixture model intractable. While Markov Chain Monte Carlo and variational techniques are the most popular approaches, this paper uses a simple, recently-proposed approximation to the DPM known as Bayesian Hierarchical Clustering (BHC) (Heller and Ghahramani, 2005a), which approximates the sum over all partitions by first greedily generating a hierarchical clustering of the tasks and then efficiently summing over the exponential number of partitions "consistent" with this hierarchy. This approach leads to a simple, yet efficient, algorithm for achieving task-level transfer.

Consider a rooted binary tree T where each task is associated with a leaf. It will be convenient to identify each internal node, T_n, with the set of leaves descending from that node. A tree-consistent partition of the tasks is any partition such that each subset corresponds exactly with a node in the graph (Figure 2). It can seen that, given any rooted tree over more than two objects, the set of all tree-consistent partitions is a strict subset of the set of all partitions. Exact inference under the DPM requires that we marginalize over the latent partition, requiring a sum over the super-exponential number of partitions. The BHC approximation works by efficiently computing the sum over the exponential number of tree-consistent partitions, using a divide-and-conquer approach to combine the results from each subtree. Intuitively, if the tree is chosen carefully, then the set of tree-consistent partitions will capture most of the mass in the posterior. BHC tries to find such a tree by combining Bayesian model selection with a greedy heuristic.

Just as in classic, agglomerative clustering (Duda et al., 2001), BHC starts with all objects assigned to their own cluster and then merges these clusters one by one, implicitly

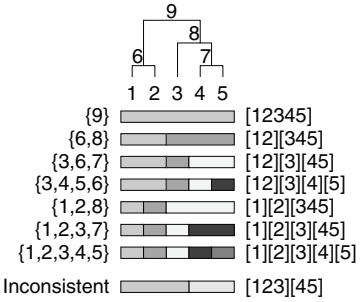

Figure 2: All tree-consistent partitions represented both as sets of nodes (left) and collection of leaves (right), and one partition that is not tree-consistent (the sets of leaves [123] is not representable by an internal node).

forming a tree that records which merges were performed. However, instead of using a distance metric and merging the nearest clusters, BHC merges the two clusters that maximize a statistical hypothesis test. At each step, the algorithm must determine which pair in the set of clusters $T_1, T_2, \ldots, T_m$ to merge next. Consider two particular clusters T_i and T_j and let D_i and D_j be the set of tasks in each respectively. The BHC algorithm calculates the posterior probability that these two clusters are in fact a single cluster $T_k = T_i + T_j$. Specifically, the hypothesis $\mathcal{H}_k$ is that the data in $D_k = D_i \cup D_j$ are identically distributed with respect to the base model (in this case, some Naive Bayes model). The probability of the data in this new cluster under $\mathcal{H}_k$, $p(D_k|\mathcal{H}_k)$ is simply the marginal likelihood of the data.

The alternative hypothesis, $\bar{\mathcal{H}}_k$ is that the data D_i and D_j are, in fact, split into two or more clusters. Computing the probability associated with this hypothesis would normally require a sum over the super-exponential number of partitions associated with the tasks in D_i and D_j. However, the clever trick of the BHC algorithm is to restrict its attention to tree-consistent partitions. Therefore, the probability of the data $D_k = D_i \cup D_j$ under $\bar{\mathcal{H}}_k$, $p(D_k|\bar{\mathcal{H}}_k) = p(D_i|T_i)\, p(D_j|T_j)$, where $p(D_i|T_i)$ is the probability of the data associated with the tree T_i. Let $\pi_k = p(\mathcal{H}_k)$ be the prior probability of the cluster T_k. Then, we can write $p(D_k|T_k)$ recursively

$$p(D_k|T_k) = \pi_k p(D_k|\mathcal{H}_k) + (1-\pi_k) p(D_i|T_i) p(D_j|T_j). \quad (2)$$

Then the posterior probability of $\mathcal{H}_k$ is

$$p(\mathcal{H}_k|D_k) = \frac{\pi_k p(D_k|\mathcal{H}_k)}{p(D_k|T_k)}. \quad (3)$$

We now present the BHC algorithm, whose output is sufficient for approximate Bayesian predictions under our model.

input data $\mathcal{D} = \{D^{(1)}, \ldots, D^{(n)}\}$
model $p(X|Y, \theta)$ and prior density $f(\theta)$
initialize number of clusters c=n, and
$D_i = \{D^{(i)}\}$ for $i = 1, \ldots, n$
while $c > 1$ **do**
Find the pair $\mathcal{D}_i$ and $\mathcal{D}_j$ with the highest posterior probability of $\mathcal{H}_k$ where $T_k = T_i + T_j$.
Merge $D_k \leftarrow D_k \cup D_j$, $T_k \leftarrow (T_i, T_j)$
Delete D_i and D_j, $c \leftarrow c - 1$
end while

Heller and Ghahramani (2005a) show that a specific choice of the prior $\pi_k = p(\mathcal{H}_k)$ leads to an approximate inference scheme for the DPM. Let α be the corresponding parameter from the DPM. Then, we can calculate the prior probability for each cluster T_j in the tree built by BHC.

initialize each leaf i to have $d_i = \alpha$, $\pi_i = 1$
for each internal node k **do**
$d_k = \alpha\Gamma(n_k) + d_{\text{left}_k} d_{\text{right}_k}$
$\pi_k = \frac{\alpha\Gamma(n_k)}{d_k}$
end for

Having built a tree that approximates the posterior distribution over partitions, we can use the tree to compute the posterior probability of an unseen label. Assume we have an unlabeled example X_k associated with the k-th task. Let A_k be the set of nodes along the path from the node k to the root in the tree generated by the BHC algorithm (e.g. in Figure 2, $A_5 = \{5, 7, 8, 9\}$). Note that the elements $T_i \in A_k$ correspond to all clusters that task k participates in across all tree-consistent partitions. Our predictive distribution for Y will then be the weighted average of the predictive distributions for each partition:

$$p(Y_k|X_k, D, T) = \sum_{T_i \in A_k} \frac{w_i}{\sum_{j \in A_k} w_j} p(Y_k|X_k, D_i), \quad (4)$$

where $w_k = r_k \prod_{i \in A_k/\{k\}} (1 - r_i)$ and $p(Y_k|X_k, D_k)$ is the predictive distribution under the base model after combining the data from all the tasks in cluster k.

While the computational complexity of posterior computation is quadratic in the number of tasks, Heller and Ghahramani (2005b) have proposed $O(n \log n)$ and $O(n)$ randomized variants.

4 Results

The utility of the type of sharing that the Clustered Naive Bayes supports can only be assessed on real data sets. To that end, we evaluated the model's predictions on a meeting classification data set collected by Rosenstein et al. (2005). The data set is split across 21 users from multiple universities, an industry lab and a military training exercise. In total, there are 3966 labeled meeting requests, with 100-400 meeting requests per user. In the meeting acceptance task, we aim to predict whether a user would `accept` or `reject` an unseen meeting request based on a small set of features that describe various aspects of the meeting.

To evaluate the clustered model, we assessed its predictive performance in a transfer learning setting, predicting labels for a user with sparse data, having observed all the labeled data for the remaining users. In particular, we calculated the receiver-operator characteristic (ROC) curve having trained on 1,2,4,8,16,32,64, and 128 training examples for each user (conditioned on knowledge of all labels for the remaining users). Each curve was generated according to results from twenty random partitions of the users' data into training and testing sets. Figure 3 plots the area under the ROC curve as a measure of classification performance versus the number of training examples.

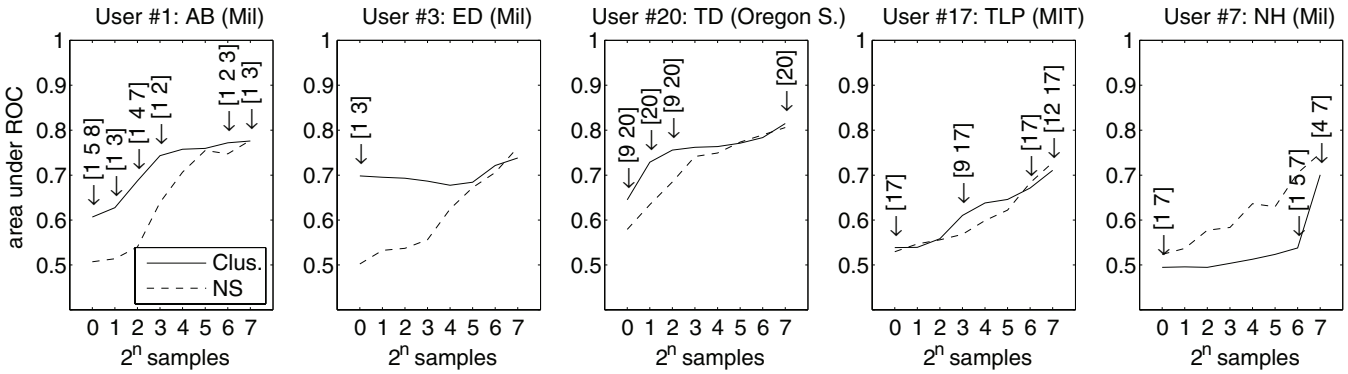

Figure 3: Area under the curve (AUC) vs. training size for five representative users. The AUC varies between 1 (always correct), 0 (always wrong), and 0.5 (chance). For each experiment we label the (MAP) cluster of users to which the user belongs. If the cluster remains the same for several experiments, we omit all but the first mention. The first three examples illustrate improved predictive performance. The last two examples demonstrate that it is possible for performance to drop below that of the baseline model.

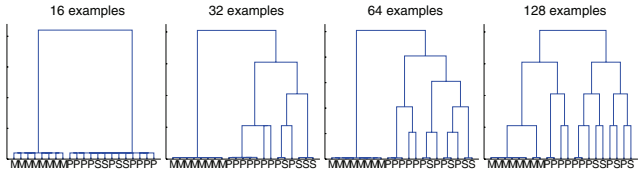

Figure 4: Progression of trees found by BHC for 16, 32, 64 and 128 examples per user. Short vertical edges indicate that two tasks are strongly related. Long vertical edges indicate that the tasks are unrelated. Key: (M)ilitary, (P)rofessor, (S)RI researcher.

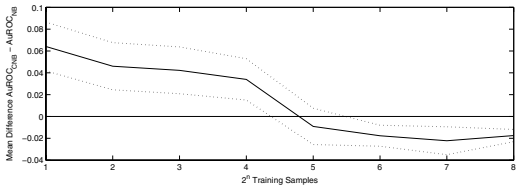

Figure 5: The clustered model has more area under the ROC curve than the standard model when there is less data available. After 32 training examples, the standard model has enough data to match the performance of the clustered model. Dotted lines are standard error.

From the 21 users, we have selected five representative samples. The first three examples (users 1, 3 and 20) show how the model performs when it is able to use related user's data to make predictions. With a single labeled data point, the model groups user 1 with two other military personnel (users 5 and 8). While at each step the model makes predictions by averaging over all tree-consistent partitions, the MAP partition listed in the figure has the largest contribution. For user 1, the MAP partition changes at each step, providing superior predictive performance. However, for the third user in the second figure, the model chooses and sticks with the MAP partition that groups the first and third user. In the third example, User 20 is grouped with user 9 initially, and then again later on. Roughly one third of the users witnessed improved initial performance that tapered off as the number of examples grew.

The fourth example (user 17) illustrates that, in some cases, the initial performance for a user with very few samples is not improved because there are no good candidate related users with which to cluster. Finally, the last example shows one of the four cases where predictions using the clustered model leads to worse performance. In this specific case, the model groups the user 7 with user 1. It is not until 128 samples that the model recovers from this mistake and achieves equal performance.

Figure 4 shows the trees and corresponding partitions recovered by the BHC algorithm as the number of training examples for each user is increased. Inspecting the partitions, they fall along understandable lines; military personnel are most often grouped with other military personnel, and professors and SRI researchers are grouped together until there is enough data to warrant splitting them apart.

Figure 5 shows the relative performance of the clustered versus standard Naive Bayes model. The clustered variant outperforms the standard model when faced with very few examples. After 32 examples, the models perform roughly equivalently, although the standard model enjoys a slight advantage that does not grow with more examples.

5 Related Work

Some of the earliest work related to transfer learning focused on sequential transfer in neural networks, using weights from networks trained on related data to bias the learning of networks on novel tasks (Caruana, 1997; Pratt, 1993). More recently, these ideas have been applied to modern supervised learning algorithms, like support vector machines (Wu and Dietterich, 2004). More work must be done to understand the connection between these approaches and the kind of sharing one can expect from the Clustered Naive Bayes model.

This work is related to a large body of transfer learning research conducted in the hierarchical Bayesian framework, in which common prior distributions are used to tie together model components across multiple data sets. The clustered model can be seen as an extension of the model first presented by Rosenstein et al. (2005) for achieving transfer with the Naive Bayes model. In that work, they fit a Dirichlet distribution for each shared parameter across all users. Unfortu-

nately, because the Dirichlet distribution cannot fit arbitrary bimodal distributions, the model cannot handle more than one cluster, i.e. each parameter is shared completely on not at all. The model presented in this paper can handle any number of users by modelling the density over parameters using a Dirichlet Process prior. It is possible to loosely interpret the resulting Clustered Naive Bayes model as grouping tasks based on a marginal likelihood metric. From this viewpoint, this work is related to transfer-learning research which aims to first determine which tasks are relevant before attempting transfer (Thrun and O'Sullivan, 1996).

Ferguson (1973) was the first to study the Dirichlet Process and show that it can, simply speaking, model any other distribution arbitrarily closely. The Dirichlet Process has been successfully applied to generative models of documents (Blei et al., 2004), genes (Dahl, 2003), and visual scenes (Sudderth et al., 2005). Teh et al. (2006) introduced the Hierarchical Dirichlet Process, which achieves transfer in document modeling across multiple corpora. The work closest in spirit to this paper was presented recently by Xue et al. (2005). They investigate coupling the parameters of multiple logistic regression models together using the Dirichlet Process prior and derive a variational method for performing inference. In the same way, the Clustered Naive Bayes model we introduce uses a Dirichlet Process prior to tie the parameters of several Naive Bayes models together for the purpose of transfer learning. There are several important differences: first, the logistic regression model is discriminative, meaning that it does not model the distribution of the inputs. Instead, it only models the distribution of the output labels conditioned on the inputs. As a result, it cannot take advantage of unlabeled data. Second, in the Clustered Naive Bayes model, the data sets are clustered with respect to a generative model which defines a probability distribution over both the inputs. As a result, the Clustered Naive Bayes model could be used in a semi-supervised setting. Implicit in this choice is the assumption that similar feature distributions are associated with similar predictive distributions. This assumption must be judged for each task: for the meeting acceptance task, the generative model of sharing is appropriate and leads to improved results.

6 Conclusion

The central goal of this paper was to evaluate the Clustered Naive Bayes model in a transfer-learning setting. To evaluate the model, we measured its performance on a real-world meeting acceptance task, and showed that the clustered model can use related users' data to provide better prediction even with very few examples.

The Clustered Naive Bayes model uses a Dirichlet Process prior to couple the parameters of several models applied to separate tasks. This approach is immediately applicable to any collection of tasks whose data are modelled by the same parameterized family of distributions, whether those models are generative or discriminative. This paper suggests that clustering parameters with the Dirichlet Process is worthwhile and can improve prediction performance in situations where we are presented with multiple, related tasks. A theoretical question that deserves attention is whether we can get improved generalization bounds using this technique. A logical next step is to investigate this model of sharing on more sophisticated base models and to relax the assumption that users are exactly identical.

References

D. Aldous. Exchangeability and related topics. *Springer Lecture Notes on Math*, 1117, 1985.

C. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *Annals of Stat.*, 2, 1974.

J. Baxter. A model of inductive bias learning. *JAIR*, 12, 2000.

D. Blei, T. Griffiths, M. Jordan, and J. Tenenbaum. Hierarchical topic models and the nested Chinese restaurant process. *NIPS*, 2004.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. *COLT*, 1992.

R. Caruana. Multitask Learning. *Machine Learning*, 28, 1997.

D. B. Dahl. Modeling differential gene expression using a Dirichlet process mixture model. *JASA*, 2003.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley and Sons, 2001.

T. Ferguson. A Bayesian Analysis of Non-parametric Problems. *Annals of Stat.*, 1, 1973.

C. Guestrin, D. Koller, C. Gearhart, and N. Kanodia. Generalizing Plans to New Environments in Relational MDPs. *IJCAI*, 2003.

K. Heller and Z. Ghahramani. Bayesian Hierarchical Clustering. *ICML*, 2005a.

K. Heller and Z. Ghahramani. Randomized Algorithms for Fast Bayesian Hierarchical Clustering. In *PASCAL Workshop on Statistics and Optimization of Clustering*, 2005b.

J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. *ICML*, 2001.

M. E. Maron. Automatic indexing: An experimental inquiry. *JACM*, 8(3), 1961.

D. Navarro, T. Griffiths, M. Steyvers, and M. Lee. Modeling individual differences using Dirichlet processes. *J. of Math. Psychology*, 50, 2006.

L. Y. Pratt. Non-literal Transfer Among Neural Network Learners. *Artificial Neural Networks for Speech and Vision*. Chapman and Hall, 1993.

M. Rosenstein, Z. Marx, L. P. Kaelbling, and T. G. Dietterich. Transfer Learning with an Ensemble of Background Tasks. *NIPS Workshop on Inductive Transfer: 10 Years Later*, 2005.

E. Sudderth, A. Torralba, W. Freeman, and A. Willsky. Describing Visual Scenes using Transformed Dirichlet Processes. *NIPS*, 2005.

Y. W. Teh, M. Jordan, M. Beal, and D. Blei. Hierarchical Dirichlet Processes. *JASM*, 2006.

S. Thrun. Is learning the n-th thing any easier than learning the first? *NIPS*, 1996.

S. Thrun and J. O'Sullivan. Discovering Structure in Multiple Learning Tasks: The TC algorithm. *ICML*, 1996.

P. Wu and T. G. Dietterich. Improving SVM accuracy by training on auxiliary data sources. *ICML*, 2004.

Y. Xue, X. Liao, L. Carin, and B. Krishnapuram. Learning multiple classifiers with Dirichlet process mixture priors. *NIPS*, 2005.