Discriminative Model Selection for Density Models

Bo Thiesson and Christopher Meek
Microsoft Research
Redmond, WA 98052-6399
{thiesson,meek}@microsoft.com

Abstract

Density models are a popular tool for building classifiers. When using density models to build a classifier, one typically learns a separate density model for each class of interest. These density models are then combined to make a classifier through the use of Bayes' rule utilizing the prior distribution over the classes. In this paper, we provide a discriminative method for choosing among alternative density models for each class to improve classification accuracy.

1 INTRODUCTION

Methods for constructing models for classification are often described as either discriminative methods or non-discriminative methods. A discriminative method constructs a model by using both positive and negative examples for each class while constructing the classifier. While discriminative methods often perform better than non-discriminative methods, the construction of discriminative models can be problematic in situations in which the number of classes is very large and when some of the classes have small counts. In such situations, one turns to non-discriminative model construction techniques. A non-discriminative method constructs a separate model for each class using only positive examples for that class.

The most common non-discriminative technique for classifier construction is the density model approach. In this approach, one builds a separate density model for each class of interest. These density models are then combined to make a classifier through the use of Bayes' rule by combining the density models with the prior distribution over the classes. Typically, the density models for a class is chosen on the basis of its ability to represent the density of the features for that class. Thus, the choice for each class is independent of the choice made for other classes despite the fact that the final use is a classification task in which the alternative classes are compared. In this paper, we describe a method for choosing among a set of alternative density models for the classes. This method can be used to improve the resulting classifier in a variety of ways. For instance, it can be used to improve overall classification accuracy or it can be used to reduce the size of the resulting classifier while maintaining classification accuracy.

In our approach, we assume that we are given a data set and a set of alternative density models for each class, and we choose a single model for each class, what we term a *configuration*. The configuration is chosen on the basis of the accuracy of a density model classifier constructed from the models in the configuration for the data set and the cost associated with these models. Clearly, the number of configurations grows quickly and, as the number of classes grows large, it becomes infeasible even to enumerate all of the configurations. We show that the problem of identifying the configuration of models that has the best accuracy can be solved using standard propagation algorithms in a (non-probabilistic) graphical model. The structure of the graphical model is determined by the types of mistakes that are made on the data set. If the resulting model is sparse then the selection can be computed effectively. In many cases, the resulting model is not sparse and we need to approximate inference to identify good configurations of models.

In Section 2, we describe the density model selection problem. In Section 3, we describe the graphical model approach to solving the problem. We also describe a simple approximation method for situations in which the method is computationally infeasible. In Section 4, we provide some preliminary experimental evidence using mixtures of Gaussian models containing different numbers of components. In these experiments, our goal is to identify the number of mixture components needed for each class to achieve optimal classification accuracy. We compare our model selection technique with a standard (non-discriminative) approach to model selection for density models. In Section 5, we discuss future work.

2 DENSITY MODEL SELECTION

In this section, we describe the problem of density model selection for classification. The variables in this problem are the class variable $C = \{c_1, \ldots, c_J\}$, where c_j is one of the possible classes of interest, and a set of variables $\mathbf{X} = \{X_1, \ldots, X_N\}$ that are to be used in making the class determination. We are given a training set $D = \mathbf{y}^1, \ldots, \mathbf{y}^L$ of L cases that have the values for both C and $\mathbf{X}$. For convenience we use D_j $(1 \leq j \leq J)$ to denote the subsets of D for which the correct class is c_j. We denote the l^{th} case of the data set for class c_j by $\mathbf{x}_j^l$ where the value for C is given by the subscript j.

We assume that we are given a set of alternative models for each class. We denote the i^{th} model for class c_j by M_j^i. To simplify the notation, we assume that each class has the same number of alternative models I. The alternative models are assumed to be given in this paper and could be learned using the training set above or some separate training set.

We use $p(c_j)$ to denote the prior on the classes and $p(\mathbf{x}|M_j^i)$ to denote the likelihood of the values $\mathbf{X} = \mathbf{x}$ given the i^{th} model for the j^{th} class. We denote a *configuration* of models by $\mathbf{s} = (s_1, \ldots, s_J)$ where $s_j \in \{1, \ldots, I\}$ indexes the model selected for class j.

We evaluate the performance for a configuration of models by the overall classification accuracy obtained for the associated test data sets $D_1, \ldots, D_J$. We let n_j be the number of cases in data set D_j. The total number of errors and overall classification accuracy are computed as follows:

$$\begin{aligned}&Errors(D_1, \ldots, D_J | s_1, \ldots, s_J)\\ &\quad = \sum_{j=1}^{J} \sum_{l=1}^{n_j} 1 - \chi^j(\mathrm{argmax}_k[p(\mathbf{x}_j^l | M_k^{s_k}) p(c_k)]) \quad (1)\end{aligned}$$

$$\begin{aligned}&Accuracy(D_1, \ldots, D_J | s_1, \ldots, s_J)\\ &\quad = 1 - \frac{Errors(D_1, \ldots, D_J | s_1, \ldots, s_J)}{\sum_{j=1}^{J} n_j} \quad (2)\end{aligned}$$

where $\chi^a(b)$ is one if $a = b$ and zero otherwise. Note, when computing the total number of errors, we are selecting the model with the highest posterior probability through the simple application of Bayes rule; $p(M_k^{s_k} | \mathbf{x}) \propto p(\mathbf{x} | M_k^{s_k}) p(c_k)$.

The goal of *discriminative density model selection* for classification is to identify the configuration $\mathbf{s}$ of models that minimizes

$$Errors(\mathbf{s}, D) + \alpha Cost(\mathbf{s}) \quad (3)$$

where α is given and $Cost(\mathbf{s})$ is some cost associated with a particular configuration. In this paper, we will choose $Cost(\mathbf{s}) = \sum_j Size(M_j^{s_j})$ where the $Size(M_j^i)$ is the number of parameters in the i^{th} model for class j.

3 ALGORITHM

We use the junction tree algorithm to identify the configuration that minimizes Equation 3. We construct a junction tree in two phases. In the first phase, we construct an error graph. The error graph captures the confusability between classes. Each class is represented by a vertex in the graph. The lack of an edge between the vertices for class c_j and c_k indicates that there is no configuration in which we can mistakenly classify a data point belonging to one of the classes as the other class. In the second phase, we begin by triangulating the error graph. The (maximal) cliques of the triangulated error graph form the junction tree. With each clique in the junction tree we associate a potential which has values indexed by the model configurations for the classes that appear in the clique. We set the values of the potentials to represent the costs (e.g., model sizes) and errors associated with classification mistakes. We use the resulting junction tree and potentials to find the optimal configuration of models by applying the min-sum algorithm (e.g., Aji and McEliece, 2000). In situations where the error graph is highly connected the computations will be intractable and one must either use approximate propagation techniques or prune the error graph.

Construction of error graph

In this phase of the algorithm we construct a representation for the confusable classes. We do this by the Construct-Error-Graph procedure. In this procedure, for each case, we rank alternative models on the basis of the likelihood of the case. This implicitly assumes that the prior for the classes is uniform. The procedure can be adapted to the case in which one does not choose to assume a uniform prior on the classes.

```
Construct-Error-Graph(DataSet D)

1) Let G be an empty graph with vertices
   corresponding to classes.

2) For each case in data set D
```

```
3)    Create a set S of all classes
      that have models that are ranked
      above any model for the correct class.

4)    Add an undirected edge between each
      pair of classes in S and an undirected
      edge between the correct class and
      each class in S.

5) Return G
```

The complexity of the min-sum procedure is related to the connectivity of the error graph for the data set. If one includes models that are extremely poor models for a class then the size of the cliques will be very large. In an effort to reduce the complexity of the procedure one can modify the algorithm to count the number of mistakes between a pair of classes and prune all of the edges between classes for which there are fewer error than some threshold.

Creating the potentials

We represent the costs and error information in a junction tree. We construct the junction tree from the error graph by triangulating the (possibly pruned) error graph. Note that it is NP-hard to identify the triangulation that minimizes the size of the maximal cliques of the resulting graph (Arnborg, Corneil, and Proskurowski 1987). The cliques of the triangulated error graph provide us with the junction tree. We associate a potential with each of the (maximal) cliques in the junction tree. The potential contains a value for each of the possible configurations of the variables in the clique with which it is associated. Recall that for our application, the variables are the classes and the possible values for the variables index the alternative models. We assign values to the potentials of the junction tree with the Fill-Potentials procedure.

```
Fill-Potentials(DataSet D, Join-Tree T)

1) For each class J find the smallest
   clique in T that contains J. For
   each configuration of the potential
   add alpha times the size of the model
   for class J that is indexed by that
   configuration.

2) For each case L in the data set

3)    Identify the correct class X for L

4)    For each model I from class X

5)       Get the list of models from the other
         classes that have better likelihoods
         for case L.

5)       Let Y be the set of classes
         represented in this list.

6)       Find a clique CL that contains both
         X and Y (Note that there might not be
         one if we have pruned; in that case
         find the clique that contains the most
         variables in X union Y and remove the
         variables from Y not in clique CL.)

7)       Let Z be the set of model
         configurations for Y in which one
         of the models in Y is better than
         model I for class X.

8)       Add one to each configuration of the
         potential for CL that has the I'th
         model for class X and is consistent
         with a configuration in Z.
```

Min-Sum Propagation

After the potentials have been created and filled we can apply the Min-Sum propagation algorithm (e.g., Aji and McEliece, 2000). If one wants to identify the best configuration in terms of accuracy one sets $\alpha = 0$. If one wants to identify a configuration that is smaller but maintains the highest accuracy possible one can adjust α to be larger than 0. As one increases the size of α the size of the resulting configurations will decrease.

Example

For this example, we assume that there are four classes and that we are given two density models and a data set for each class. We denote the density models for the classes by $M_1 = \{M_1^1, M_1^2\}$, $M_2 = \{M_2^1, M_2^2\}$, $M_3 = \{M_3^1, M_3^2\}$, and $M_4 = \{M_4^1, M_4^2\}$ and we denote the associated training data sets for the classes by $D_1 = \{\mathbf{x}_1^1, \mathbf{x}_1^2\}$, $D_2 = \{\mathbf{x}_2^1, \mathbf{x}_2^2\}$, $D_3 = \{\mathbf{x}_3^1, \mathbf{x}_3^2\}$, and $D_4 = \{\mathbf{x}_4^1, \mathbf{x}_4^2\}$.

Each column of Table 1 contains a case and a list of models ranked according to likelihood for that case. For instance, for data case $\mathbf{x}_1^1$ the model with highest likelihood is model M_1^2 followed by models M_3^1 and M_1^1. Hence, selecting a model configuration involving M_3^1 and M_1^1 would result in a wrong classification for case $\mathbf{x}_1^1$. Note that no models ranked below the lowest ranked correct model are included in the list; these models have no effect on the model configuration selection method.

When constructing the error graph, case $\mathbf{x}_2^1$ leads the

Case	$\mathbf{x}_1^1$	$\mathbf{x}_1^2$	$\mathbf{x}_2^1$	$\mathbf{x}_2^2$	$\mathbf{x}_3^1$	$\mathbf{x}_3^2$	$\mathbf{x}_4^1$	$\mathbf{x}_4^2$
Model ranking	M_1^2 M_3^1 M_1^1	M_1^1 M_3^2 M_1^2	M_2^2 M_3^1 M_3^2 M_1^1 M_2^1	M_2^2 M_2^1	M_3^1 M_4^2 M_3^2	M_1^2 M_3^2 M_4^1 M_4^2 M_3^1	M_4^2 M_4^1	M_4^2 M_3^1 M_4^1

Table 1: Models ranked according to likelihood for each individual case in our example.

		M_3^1	M_3^2
M_1^1	M_2^1	2	1
	M_2^2	1	0
M_1^2	M_2^1	2	3
	M_2^2	1	2

	M_4^1	M_4^2
M_3^1	2	1
M_3^2	0	1

Table 2: Potential tables after the Fill-Potentials procedure.

algorithm to add all edges between model classes M_1, M_2, and M_3. Adding edges to the error graph for all cases in the example results in a graphical structure as shown in Figure 1. This graph is triangulated.

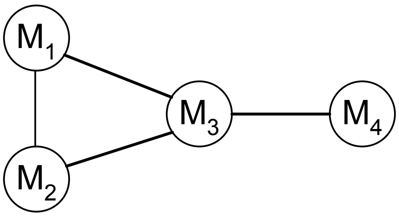

Figure 1: Error graph for the example.

We begin by initializing the value for all potentials to zero for every configuration. For this example, we set $\alpha = 0$, that is, we effectively have no difference in the costs for alternative configurations. Next we consider each case and increment the potentials appropriately. When considering case $\mathbf{x}_4^2$, we add a count of one to the configuration (M_3^1, M_4^1) in the potential table for M_3 and M_4. For a slightly more complicated example, when considering case $\mathbf{x}_2^1$, we add a count of one to the configurations (M_1^1, M_2^1, M_3^1), (M_1^1, M_2^1, M_3^2), (M_1^2, M_2^1, M_3^1), and (M_1^2, M_2^1, M_3^2) in the potential table for M_1, M_2, and M_3. Adding counts for all the cases results in potential tables as shown in Table 2.

Table 3 shows the consistent marginal potential tables as obtained after min-sum propagation. From these potential tables it can be seen that the configuration $(M_1^1, M_2^2, M_3^2, M_4^1)$ with a total of $0+0-\min\{0,1\} = 0$ errors is the best configuration for the classification task.

		M_3^1	M_3^2
M_1^1	M_2^1	3	1
	M_2^2	2	0
M_1^2	M_2^1	3	3
	M_2^2	2	2

	M_4^1	M_4^2
M_3^1	3	2
M_3^2	0	1

Table 3: Consistent marginal potential tables after min-sum propagation.

Data set	#Train	#Test	#Variables
M54	928	618	33
M56	1388	925	33
M64	986	656	33
M78	3720	2482	33
N86	5160	3440	33
N99	6000	4000	33
N146	2843	1894	33
N158	1062	708	33

Table 4: Statistics for sub-phonetic data sets used in our experiments.

4 Experiments

In this section, we describe preliminary experiments with our approach for discriminative model selection for density model classification.

4.1 Data sets

We evaluate our method on real-world speech data, which has been partitioned into data sets associated with individual sub-phonetic events observed for continuous speech. We have selected eight highly confusable sub-phonetic events for the classification task. Each case in a data set represent an observation on 33 continuous variables, which are 12 mel-scale frequency cepstrum coefficients (MFCCs), log-energy and their first and some second order dynamics (Huang, Acero, Alleva, Hwang, Jiang, and Mahajan 1995). The data set for each individual sub-phonetic event is partitioned by a 60/40 split into training and test data. Characteristics for the data sets are summarized in Table 4.

4.2 Models

We investigate finite mixture models in which each component model encodes the mutual independence of the variables $\mathbf{X} = (X_1, \ldots, X_N)$—that is, mixtures of multivariate Gaussians with diagonal covariance matrices.

For each data set we learn twenty different mixture models with one through twenty components. The individual models are learned by applying the EM algorithm to perform MAP estimation using diffuse priors similar to those described in Thiesson, Meek, Chickering, and Heckerman (1999). We use the discriminative model selection method, described in this paper, to select the configuration of models which maximizes the overall classification accuracy on the training data. For comparison, we consider a trivial method that selects the configuration containing only one component models and another trivial method that selects the configuration containing only twenty component models. We also consider the method that selects the configuration where the model for a class is selected to maximize the Cheeseman-Stutz score (Cheeseman and Stutz, 1995), as suggested in Thiesson *et al.* (1999).

4.3 Results

Table 5 shows the overall classification accuracy for the configurations obtained by our selection methods. The highest accuracy is obtained for the configuration selected by the discriminative method suggested in this paper. In addition, with the exception of the method of choosing the smallest configuration, all of the methods produce configurations that yield good classifiers.

Table 6 shows the number of mixture components for the selected configurations. The Cheeseman-Stutz and the discriminative model selection methods choose configurations that are significantly smaller than the most complex configuration. It is interesting to note that the discriminative selection improves the classification accuracy while reducing the size of the resulting classifier even when not using a cost term to penalize model size (i.e., $\alpha = 0$). By comparison, the Cheeseman-Stutz method yields an even smaller configuration but one that has slightly worse accuracy.

For large scale classification tasks where many thousand models are used for classification, as is the case for the speech recognition system described in Huang *et al.* (1995), storage or memory constraints may force us to choose a model configuration that is smaller than the optimal configuration. By adding a cost term to the model configuration selection criterion, such as $Cost(s) = \sum_j (Size(M_j^{s_j})$, we can reduce the size of the selected configuration while maintaining accuracy.

Selection Method	Accuracy
One component models	0.585
Twenty component models	0.650
Cheeseman-Stutz selection	0.643
Discriminative model selection	0.653

Table 5: Classification accuracies for the smallest, largest, Cheeseman-Stutz selected, and demonstratively selected configurations.

Data set	One	Twenty	CS	DMS
M54	1	20	3	5
M56	1	20	2	5
M64	1	20	3	2
M78	1	20	6	10
N86	1	20	8	9
N99	1	20	10	12
N146	1	20	7	8
N158	1	20	3	5
Total	8	160	42	56

Table 6: Number of mixture components in each model for the smallest (One), largest (Twenty), Cheeseman-Stutz selected (CS), and discriminatively selected model configuration (DMS).

Finally it should be noted that we have seen situations where the most complex model configuration significantly degrade performance, so even if the size is not a constraint, selecting the most complex model configuration is not, in general, a good choice.

5 DISCUSSION

In this paper, we have described a method for choosing among a set of density models for a set of classes of interest. Our method allows one to choose a set of density models that can achieve good accuracy while allowing one to limit the cost (e.g., size) of the resulting set of models. We plan on applying this methods to alternative data sets and alternative classes of density models. Alternative approximations such as loopy min-sum propagation is a natural alternative to the approach of pruning the error-graph. In addition, more work is needed to understanding how to reuse computations for multiple runs when adjusting the tuning parameter α.

Acknowledgements

We would like to thank Max Chickering for helpful comments on a previous version of this paper.

References

[Aji and McEliece, 2000] Aji, S. and McEliece, R. (2000). The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343.

[Arnborg et al., 1987] Arnborg, S., Corneil, D. G., and Proskurowski, A. (1987). Complexity of finding embeddings in a k-tree. *SIAM Journal of Algebraic Discrete Methods*, 8(2):277–284.

[Cheeseman and Stutz, 1995] Cheeseman, P. and Stutz, J. (1995). Bayesian classification (AutoClass): Theory and results. In Fayyad, U., Piatesky-Shapiro, G., Smyth, P., and Uthurusamy, R., editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI Press, Menlo Park, CA.

[Huang et al., 1995] Huang, X., Acero, A., Alleva, F., Hwang, M.-Y., Jiang, L., and Mahajan, M. (1995). Microsoft Windows highly intelligent speech recognizer: Whisper. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, 1995. ICASSP-95*, volume 1, pages 93–96.

[Thiesson et al., 1999] Thiesson, B., Meek, C., Chickering, D., and Heckerman, D. (1999). Computationally efficient methods for selecting among mixtures of graphical models, with discussion. In *Bayesian Statistics 6: Proceedings of the Sixth Valencia International Meeting*, pages 631–656. Clarendon Press, Oxford.