Phonetic Classification Using Controlled Random Walks

Katrin Kirchhoff

Department of Electrical Engineering
University of Washington, Seattle, WA, USA
katrin@ee.washington.edu

Andrei Alexandrescu†

Facebook
andrei.alexandrescu@fb.com

Abstract

Recently, semi-supervised learning algorithms for phonetic classifiers have been proposed that have obtained promising results. Often, these algorithms attempt to satisfy learning criteria that are not inherent in the standard generative or discriminative training procedures for phonetic classifiers. Graph-based learners in particular utilize an objective function that not only maximizes the classification accuracy on a labeled set but also the global smoothness of the predicted label assignment. In this paper we investigate a novel graph-based semi-supervised learning framework that implements a *controlled* random walk where different possible moves in the random walk are controlled by probabilities that are dependent on the properties of the graph itself. Experimental results on the TIMIT corpus are presented that demonstrate the effectiveness of this procedure.

Index Terms: phonetic modeling, classification, phonetic similary, graph-based learning

1. Introduction

A persistent problem in automatic speech processing is the need for adaptation of speech processing systems to varying test conditions for which labeled data is often not available. At the acoustic level, a number of successful adaptation algorithms have been developed (e.g. [1, 2]) that transform model parameters to better fit the test data. Another line of research that is beginning to emerge attempts to include unlabeled data during the training process [3, 5, 6, 7, 8]. Graph-based training methods in particular [5, 8] add a complementary training criterion that guides model parameters towards producing *globally smooth* outputs, i.e. speech samples that are in close proximity in the feature space are encouraged to receive the same output label. In applying this criterion, similarities between training and test samples are taken into account *as well as similarities between different test samples*; in this way, the training algorithm takes into consideration the global underlying structure of the test set and can influence the model parameters to fit this structure.

In this paper we build on previous work on graph-based learning for acoustic classification and test a novel, recently proposed graph-based learning algorithm termed Modified Adsorption (MAD) [9]. Unlike previously proposed graph-based algorithms that can be considered simple random-walk algorithms over a data graph, MAD introduces different probabilities for individual random walk operations such as termination, continuation, and injection. The probabilities for these operations are set depending on properties of the graph itself, notably (as explained below in Section 2.2) the neighbourhood entropy of a given node. When building a graph representing a speech database, the neighbourhood sizes and entropies often exhibit large variance, due to the uneven distribution of phonetic classes (e.g. vowels are more frequent than laterals). MAD offers a unified framework for accommodating such effects without the need for downsampling or upsampling the training data. In the past, MAD has been applied to class-instance extraction from text [10] but it has to our knowledge not been applied to speech classification tasks before. In this paper we contrast MAD to the most widely used graph-based learning algorithm, viz. label propagation.

†Work was performed while the author was a student at the University of Washington.

2. Graph-Based Learning

In graph-based learning, the training and test data are jointly represented as a graph $G(V, E, W)$, where V is a set of vertices representing the data points, E is a set of edges connecting the vertices, and W is a weight function labeling the edges with weights. An edge $e = (v_1, v_2) \in V \times V$ indicates that similarity exists between the data points represented by v_1 and v_2, and the weight w_{12} expresses the degree of similarity between these points. The graph can be densely or sparsely connected, and a wide range of similarity functions can be used for W, though typically non-negative functions are used. A weight of 0 implies absence of an edge between two vertices. The graph is usually represented as an $n \times n$ weight matrix, where n is the number of data points (composed of l labeled and u unlabeled data points; $l + u = n$). Additionally, we have labels $y_1, ..., y_l$ for the set of labeled points $x_1, ..., x_l$. Assuming that there are m different labels in total, we can define an $n \times m$ matrix Y with entries for the l labeled points and zeros for the u unlabeled points. The goal is to utilize the graph in inferring labels for the unlabeled points. A number of different learning algorithms have been proposed in the past, including spectral graph transducer [11], label propagation [12], and measure propagation [8].

The key advantage of graph-based learning is that similarities between training and test points as well as similarities between different test points are taken into account. This is particularly useful when the data to be classified lives on an underlying low-dimensional manifold. It has been shown in the past [13] that the phonetic space exhibits such a manifold structure, and graph-based techniques have been used successfully for phonetic classification in previous work [5, 8].

2.1. Label Propagation

Label propagation (LP), one of the earliest graph-based learning algorithms [12], minimizes the cost function

$$S = \sum_{i,j=l+1,\ldots,n} W_{ij}(\hat{y}_i - \hat{y}_j)^2 \quad (1)$$

subject to the constraint that

$$y_i = \hat{y}_i \quad \forall i = 1, ..., l \tag{2}$$

where y is the true label, and $\hat{y}$ is the predicted label. Thus, two samples linked by an edge with a large weight should ideally receive the same labels. The cost function has a fixed-form solution but usually an iterative procedure (Algorithm 1) is used. Upon termination, $Y_{l+1,...,n}$ contains entries for the unlabeled data rows in the label matrix. LP can also be viewed from the perspective of random walks on graphs: after convergence, y_{ij} contains the probability that a random walk starting at unlabeled vertex i will terminate in a vertex labeled with j.

Algorithm 1: Iterative Label Propagation Algorithm

1. Initialize the row-normalized matrix P as $P_{ij} = \frac{W_{ij}}{\sum_j W_{ij}}$;
2. Initialize a $n \times m$ label matrix Y with one-hot vectors encoding known labels for the first l rows: $Y_i = \delta_C(y_i)\ \forall i \in \{1, 2, \ldots, l\}$ (the remaining rows of Y can be zero);
3. $Y' \leftarrow P \times Y$;
4. Clamp already-labeled data rows: $Y'_i = \delta_C(y_i) \forall i \in \{1, 2, \ldots, l\}$;
5. If $Y' \cong Y$, stop;
6. $Y \leftarrow Y'$;
7. repeat from step 3

2.2. Modified Adsorption

Modified adsorption (MAD) [9] builds on the Adsorption algorithm [14] in that it introduces three different probabilities for each vertex v in the graph that correspond to the random walk operations of (a) continuing to a neighbouring vertex according to the transition probabilities v ($cont$); (b) stopping and returning the prior ("injected") label probabilities (inj); or (c) abandoning and returning a default label vector ($abnd$), which can be an all-zero vector or a vector giving the highest probability to a default or "dummy" label. Each action is controlled by a corresponding probability that is dependent on the vertex v, i.e. p_v^{cont}, p_v^{inj}, p_v^{abnd}, and all must sum to 1. For unlabeled nodes, p^{inj} is 0.

In order to determine the values of p_v^{cont}, p_v^{inj}, and p_v^{abnd}, we use the procedure in [15], which defines two quantities, c_v and d_v for each vertex v. The first of these is defined as

$$c_v = \frac{log\beta}{log(\beta + e^{H[v]})} \tag{3}$$

$H[v]$ is the so-called neighbourhood entropy of v, i.e. the entropy of the probability distribution of outgoing edges of v:

$$H[v] = -\sum_u P(u|v) log(P(u|v)) \tag{4}$$

d_v is defined as

$$d_v = \begin{cases} (1 - c_v) \times \sqrt{H[v]} & \text{if } v \text{ is labelled} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Finally, a normalization factor $z_v = max(c_v + d_v, 1)$ is defined and the probabilities are set as $p_v^{cont} = c_v/z_v$, $p_v^{inj} = d_v/z_v$, and $p_v^{abnd} = 1 - p_v^{cont} - p_v^{inj}$. The effect of setting c_v and d_v in this manner is to penalize neighbourhoods with a large number of nodes and high entropy to the benefit of small, low-entropy neighbourhoods or prior label information.

MAD optimizes the following objective function:

$$\begin{aligned} S = \sum_i [&\mu_1 \sum_m p_i^{inj} (y_i^m - \hat{y}_i^m)^2 \\ &+ \mu_2 \sum_m \sum_{j \in N(v_i)} p_i^{cont} W_{ij} (\hat{y}_i^m - \hat{y}_j^m)^2 \\ &+ \mu_3 \sum_m p_i^{abnd} (y_i^m - r_i^m)^2] \end{aligned} \tag{6}$$

The optimization criterion thus consists of three parts: (a) correspondence of the predicted labels with the true seed labels; (b) smoothness of the predicted labeling within the graph neighborhood of the current vertex, $N(v)$; and (c) regularization by penalizing divergence from a default value r. Each part is weighted by a coefficient μ which controls its relative importance.

Another difference to the LP algorithm is that seed labels are allowed to change during the training process and need not stay fixed. This may be useful when the labeling of the training data is noisy.

In [15] it was shown that the function in Equation 6 can be described as a linear system that can be optimized by the Jacobi method [4], leading to the training procedure shown below:

Algorithm 2: Iterative MAD Algorithm

Input: $G = (G, E, W); Y_{1,...,l}; p^{inj}, p^{cont}, p^{abnd}; \mu_1, \mu_2, \mu_3$
Output: $\hat{Y}_{1,...,n}$

1. Initialize $\forall i = 1, \ldots, l$:
2. $\hat{y}_i = y_i$
   $M_{ii} \leftarrow \mu_1 \times p_i^{inj} + \mu_2 \times p_i^{cont} \times \sum_{j \in N(v_i)} W_{ij} + \mu_3$;
3. **repeat**
4. $\quad D_i \leftarrow \sum_{j \in N(v_i)} (p_i^{cont} W_{ij} + p_j^{cont} W_{ji}) \hat{Y}_j$;
5. $\quad$ **for** $i \in 1, \ldots, n$ **do**
6. $\quad\quad \hat{Y}_i \leftarrow \frac{1}{M_{ii}} (\mu_1 \times p_i^{inj} \times Y_i + \mu_2 \times D_i + \mu_3 \times R_i)$
7. $\quad$ **end**
8. **until** *convergence* ;

3. MAD for Acoustic-Phonetic Classification

In order to apply MAD to acoustic-phonetic classification, we need to first construct the data graph. To this end we need to define an appropriate similarity measure in acoustic-phonetic space. One common approach is to use Euclidean distance between MFCC feature vectors, and to subsequently convert distance values into similarity values by means of a Gaussian kernel. In contrast, we follow our earlier work [5] and first apply a neural-network classifier to the data (trained on only a limited subset of the training data) and use the probability distributions output by the first-level classifier as a new feature space. In our experience this procedure has yielded superior performance. The reason is that the first-pass classifier (if well-trained) removes noise in the data and produces a new feature space where classes are clustered more tightly, thus yielding superior graphs. Another advantage is that distance measures with well-defined

probability semantics can be used in order to comnpute similarity values for the graph edges, whereas distance measures such as Euclidean distance that are used with traditional speech features often make unwarranted assumptions about the underlying distribution of features. We compute distances in the probability space using Jensen-Shannon divergence:

$$d_{JS}(a,b) = \frac{d_{KL}(a,m) + d_{KL}(b,m)}{2} \quad (7)$$

where m is the equal-weight interpolation of a and b:

$$mi_{[i]} = \frac{a_{[i]} + b_{[i]}}{2} \quad (8)$$

and d_{KL} is the Kullback-Leibler divergence

$$d_{KL}(a,b) = \sum_i a[i] \log \frac{a_{[i]}}{b_{[i]}} \quad (9)$$

The resulting values are passed through a Gaussian kernel:

$$W_{ij} = \exp\left[-\frac{d(x_i,x_j)^2}{\alpha^2}\right] \quad (10)$$

where $d(x_i, x_j)$ is the distance between x_i and x_j and α is a bandwidth hyperparameter. This parameter has a significant influence on the performance of the graph-based learner since it determines the structure of the graph. There is no optimal method for setting this parameter; practical applications have used a variety of heuristic methods. Entropy minimization has been proposed [17], based on the intuition that a good prediction is a confident prediction. Under this approach, α is optimized to yield a labeling of minimum entropy, subject to the restriction that the labeling is consistent with the known training labels. The problem with this approach is scalability since α needs to be optimized by gradient descent for each test utterance, which adds considerably to the total run-time of the algorithm. A faster method that has been proposed is based on the 3σ rule of the Normal distribution: samples lying beyond 3σ from average have a negligible probability. The technique selects the smallest distance $d(x_i, x_j)$ between samples with different labels, and selects α as one-third of that minimum distance; however, this method is extremely sensitive to noise and outliers.

We have previously proposed an efficient alternative method of setting α that does not need to be tuned at run-time but can be carried out in advance since it only uses the training data [5]. First, the average between-class distance $\overline{d}_b$ and the average within-class distance $\overline{d}_w$ are computed:

$$\overline{d}_b = \frac{1}{N_b} \sum_{y_i \neq y_j} d(x_i, x_j) \quad (11)$$

$$\overline{d}_w = \frac{1}{N_w} \sum_{y_i = y_j} d(x_i, x_j) \quad (12)$$

where N_b and N_w are the corresponding sample counts. Once d_b and d_w have been computed, we set α such that two samples that are distanced at $\frac{d_g + d_w}{2}$ have a similarity of 0.5:

$$\exp\left[-\frac{(d_b + d_w)^2}{4\alpha^2}\right] = \frac{1}{2} \Rightarrow \alpha = \frac{d_b + d_w}{2\sqrt{\ln 2}} \quad (13)$$

Our motivation is that two samples with a maximally ambiguous distance should be maximally ambiguous in terms of similarity as well.

One of the drawbacks of graph-based techniques is their computational complexity, especially during the process of graph construction. In order to compute pairwise similarities between all data instances, n^2 operations need to be carried out, which is prohibitive for large n. Since our training data is typically larger than our test data (even when using a subset of all the data available), most of the computation is incurred by comparing unlabeled to labeled data instances. However, since we are working with a feature space defined by probability distributions, we can eliminate this step and simply define the ideal feature vectors for all classes: each m-dimensional feature vector has a 1 at the position corresponding to the correct label and 0 elsewhere. We define m such vectors and use them as labeled (or 'seed') nodes. We then compute the Jensen-Shannon divergence between each of the test nodes (that are associated with the soft probability distributions output by the first-pass classifier) and the seed nodes. In order to speed up the pairwise comparisons between unlabeled-labeled and unlabeled-unlabeled nodes, we use kd-trees [18] in combination with the Jensen-Shannon divergence as a fast k-nearest neighbour search method. We do not set k to a fixed value but allow all neighbours within a given divergence value range r ($r = 7$ for the labeled set and $r = 0.2$ for the unlabeled set). Different values are used for the labeled vs. unlabeled set since the feature representation of the labeled points consists of 1/0 vectors whereas the feature vectors for the unlabeled points contain soft probability distributions. The divergence values therefore lie in different ranges.

4. Data and Experiments

We perform experiments on *frame-based* acoustic classification on the TIMIT corpus. We use a training set of 2544 sentences (1124823 frames), a development set of 1152 sentences (353001 frames) (taken from the original training set), and the core test set of 192 sentences (57919 frames). The data was preprocessed into vectors of 26 MFCC coefficients (12 MFCC features plus energy plus first derivatives), at a frame rate of 10ms, with a window of 25ms As a baseline classifier we use a multi-layer perceptron (MLP). The input to the MLP consists of nine windows of 26 coefficients and thus has 234 units. The number of hidden units is 400. The output layer has 48 units: as is common with TIMIT, we perform training on 48 phone classes that are then collapsed into 39 classes for the purpose of evaluation. Note that classification is on a frame-by-frame basis; there is no higher-level segmental modeling involved. We investigate various training conditions that differ in the amount of training data; we utilize randomly sampled subsets of 10%, 30%, 50%, respectively, as well as the full training set. Both the MLP and the graph-based classifier are optimized on the development set. The MLP was trained using back-propagation and cross-validation on the development set. The parameters of the graph-based learners were optimized separately for each training condition. The optimal values were obtained by using a grid search over possible settings and evaluating the performance on the development set. The graph-based learners were run for 10 iterations.

For our experiments we use the implementation of MAD provided in the Junto software package [1].

[1] www.sourceforge.net/projects/junto

Training set	Baseline (MLP)	LP	MAD
10%	59.39	59.16	60.21
30%	62.65	61.69	63.38
50%	63.00	62.24	63.69
full	65.95	64.39	64.76

Table 1: Frame classification accuracies (%) for baseline supervised learner (MLP) vs. graph-based semi-supervised classifiers.

5. Results

Results obtained by the supervised baseline classifier (the MLP described above) and the two semi-supervised learners on the test set are shown in Table 1. The evaluation measure provided is the percentage of correctly classified frames. Differences in accuracy of 0.5% or larger are statistically significant at the 0.05 level using a difference of proportions test. As can be seen from the table, the MAD classifier outperforms both the baseline and the LP classifier in the first three training conditions; it falls short of the baseline when using the full training set though still outperforms the supervised MLP classifier. We hypothesize that the improved performance of MAD is due to the controlled nature of the random walk implemented in MAD, particularly the discounting of vertices with a large number of edges. These are often associated with the silence label or one of the more frequent phone classes, which tend to dominate the graph. The LP-based learner, by contrast, treats all edges equally; we hypothesize that this is the cause of its inferior performance.

6. Conclusions

We have investigated the application of a recently proposed graph-based learning algorithm, MAD, to acoustic-phonetic classification. Results show that MAD is successful at utilizing unlabeled data, outperforming a standard MLP classifier significantly when less than the full training set is used, and despite using only a single seed node per class. It also outperforms the standard label propagation algorithm. In the future we will investigate incorporating dependencies between labels as well as applications of graph-based learning to phone recognition as opposed to frame-level phonetic classification, and the integration of graph-based phonetic classifiers into fully-fledged speech recognition systems.

7. Acknowledgements

This work was supported by NSF grant IIS-0812435. Any opinions, findings, and conclusions or recommendations expressed in this material belong to the authors and do not necessarily reflect the views of this agency.

8. References

[1] M. Gales and P.C. Woodland, "Mean and Variance Adaptation within the MLLR framework", *Computer Speech and Language 10*, 1996, 249-264.

[2] J.L. Gauvain and C.H. Lee, "Maximum A-Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains", *IEEE Transactions on Speech and Audio Processing 2*, 1994, 291-298.

[3] G. Levow, "Unsupervised and semi-supervised learning of tone and pitch acceent", *Proceedings of HLT-NAACL*, 2006

[4] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Society for Industrial Mathematics, 2003

[5] A. Alexandrescu and K. Kirchhoff, "Graph-based learning for phonetic classification", *Proceedings of ASRU*, 2007

[6] J.-T. Huang and M. Hasegawa-Johnson, "Semi-supervised training of Gaussian mixture models by conditional entropy minimization", *Proceedings of Interspeech*, 2010, 1353-1356.

[7] J.H. Jeon and Y. Liu, "Semi-supervised Learning for Automatic Prosodic Event Detection Using Co-training Algorithm", *Proceedings of ACL*, 2009

[8] A. Subramanya and J. Bilmes, *Semi-supervised Learning with Measure Propagation*, Technical Report UWEE-TR-2010-0004, University of Washington

[9] P.P. Talukdar and K. Crammer, "New regularized algorithms for transductive learning", *Proceedings of ECML-PKDD*, 2009

[10] P.P. Talukdar and F. Pereira, "Experiments in Graph-Based Semi-Supervised Learning Methods for Class Instance Acquisition", *Proceedings of ACL*, 2010

[11] T. Joachims, "Transductive Learning via Spectral Graph Partitioning, International Conference on Machine Learning", *Proceedings of ICML*, 2003

[12] X. Zhu and Z. Ghahramani, *Learning from labeled and unlabeled data with label propagation*, Technical Report CMU-CALD-02, Carnegie Mellon University, 2002

[13] A. Jansen and P. Niyogi, "Semi-supervised learning of speech sounds", *Proceedings of Interspeech*, 2007, 86-89.

[14] S. Baluja et al., "Video suggestion and discovery for YouTube: taking random walks through the view graph", *Proceedings of WWW*, 2008

[15] P.P. Talukdar, *Graph-based Weakly Supervised Methods for Information Extraction and Integration*, PhD thesis, University of Pennsylvania, 2010

[16] A. Alexandrescu, *Scalable Graph-Based Learning for Human Language Technology*, PhD thesis, University of Washington, CSEE Department, 2009

[17] X. Zhang and W.S. Lee, "Hyperparameter learning for graph-based semi-supervised learning algorithms", *Proceedings of NIPS*, 2006

[18] J. L. Bentley, "Multidimensional binary search trees used for associative searching", *Communications of the ACM 18(9)*, 1975, 509-517