On the Semi-Supervised Learning of Multi-Layered Perceptrons

Jonathan Malkin, Amarnag Subramanya, Jeff Bilmes

Department of Electrical Engineering
University of Washington, Seattle, WA
{jsm,asubram,bilmes}@ee.washington.edu

Abstract

We present a novel approach for training a multi-layered perceptron (MLP) in a semi-supervised fashion. Our objective function, when optimized, balances training set accuracy with fidelity to a graph-based manifold over all points. Additionally, the objective favors smoothness via an entropy regularizer over classifier outputs as well as straightforward ℓ_2 regularization. Our approach also scales well enough to enable large-scale training. The results demonstrate significant improvement on several phone classification tasks over baseline MLPs.

Index Terms: semi-supervised learning, neural networks, phone classification

1. Introduction

Multi-layer perceptrons (MLPs) [3] have amassed a solid record in speech recognition as conceptually simple but consistently effective discriminative classifiers. And in addition to static classification, MLPs can play an important role in time series classification when used to generate a nonlinear feature transformation for Tandem acoustic modeling [21, 5]. Such features are then used as inputs to a hidden Markov model (HMM) for time series modeling.

Although machine learning methods such as support vector machines (SVMs) [18] have been in vogue more recently, MLPs still demonstrate their strength by handling large data sets where many kernelized non-parametric methods such as SVMs scale poorly if one wants to use a nonlinear kernel. Despite the utility of MLPs, however, there has been surprisingly little work extending them to semi-supervised learning (SSL). Especially for speech, where training data is very cheap to acquire but expensive to label (especially at the frame level), extending MLPs to take advantage of unlabeled data holds much promise.

In this paper, we propose a new graph-based SSL training objective that is suitable for training parametric classifiers via stochastic gradient descent [11]. We apply this objective to the training of MLPs, making them suitable to the case where there are unlabeled as well as labeled training samples. This gives us a simple and tractable algorithm.

As with most graph-based SSL methods, we assume that the data, both labeled and unlabeled, are embedded within a low-dimensional manifold expressed by the graph. Despite this we do *not* need to learn either an explicit representation of a low-dimensional manifold (unlike, e.g., [19]) or an explicit mapping between any manifold space and our feature space.

In this work, our experiments focus exclusively on frame-by-frame phone classification. We expect, however, that our method will generalize to other uess of MLPs as multi-class classifiers and may provide improvements to Tandem acoustic modeling or other forms of hybrid MLP/HMM systems.

Although we have examined only MLPs in this work, our proposed objective can be applied to models other than MLPs, for instance the Ratio Semi-Definite Classifier (RSC) family [15, 16] or even Gaussian mixture classifiers. In doing so, we have created an elegant and natural way to turn almost any differentiable model into a semi-supervised learner.

This paper is based on work supported by the National Science Foundation under grant IIS-0326382 and by the Office of Naval Research under MURI grant N000140510388.

2. A Novel Objective for SSL MLP Training

Let $\mathcal{D}^\ell = \{(\mathbf{x}_i, y_i)\}_{i=1}^{\ell}$ be labeled training data and $\mathcal{D}^u = \{\mathbf{x}_i\}_{i=\ell+1}^{\ell+u}$ be unlabeled points, where $n = u + \ell$ so we have n points in total. Additionally, we use $\mathbf{p}_\theta(\mathbf{x}_i)$ to denote the vector of posterior probabilities output by a classifier governed by parameters θ given point $\mathbf{x}_i$ — thus, the k^{th} entry of $\mathbf{p}_\theta(\mathbf{x}_i)$ is the value $p_\theta(k|\mathbf{x}_i)$ for a conditional model governed by parameters θ. We use $\mathbf{t}_i$, $1 \leq i \leq l$, to denote a probabilistic label vector for the i^{th} training sample. When the input $\mathbf{x}_i$ has a single output y_i, $\mathbf{t}_i$ is a vector with zeros except for a one in position y_i ("hard-labels"). We also assume that $\{\mathbf{x}_i\}_{i=1}^{n}$ are embedded in a weighted undirected graph $\mathcal{G} = (V, E)$ where $V = \{1, \ldots, n\}$ and $E = V \times V$ are the set of edges between vertices. We use $\omega_{ij} \in \mathbf{W}$ to denote the weight of the edge between vertices i and j. For information on how to construct the graph, see section 4.1.

We propose a novel objective function $J(\theta)$ to be minimized for semi-supervised training of MLPs:

$$J(\theta) = \sum_{i=1}^{\ell} D(\mathbf{t}_i \parallel \mathbf{p}_\theta(\mathbf{x}_i)) + \gamma \sum_{i,j=1}^{n} \omega_{ij} D(\mathbf{p}_\theta(\mathbf{x}_i) \parallel \mathbf{p}_\theta(\mathbf{x}_j)) + \kappa \sum_{i=1}^{n} D(\mathbf{p}_\theta(\mathbf{x}_i) \parallel \mathbf{u}) + \lambda \|\theta\| . \quad (1)$$

Here, $\mathbf{u}$ is the uniform distribution, $D(p \parallel q)$ is the Kullback-Leibler (KL) divergence between probability distributions p and q, and $\|\theta\|$ is an ℓ_2 parameter regularizer (e.g., Frobenius norms in the matrix case). The choice of trade-off parameters, which include $\gamma, \kappa, \lambda \geq 0$, is discussed in section 5. Here, we describe each of the terms in $J(\theta)$ in detail.

The first term in Equation 1 optimizes towards producing distributions close to the target distributions. If $\gamma, \kappa = 0$, $J(\theta)$ is a standard fully-supervised MLP training criterion. In the case of hard labels we have that $D_{KL}(\mathbf{t}_i \parallel \mathbf{p}_i) = -\log p_\theta(y_i|\mathbf{x}_i)$ which is the standard conditional maximum likelihood criterion.

The second term in Equation 1, often called a graph regularizer, favors smooth solutions over the graph. That is, nearby points in the graph — those with large weights ω_{ij}, or more generally geodesically close — should have similar posterior distri-

butions. This term captures the manifold assumption mentioned earlier. Note that the graph regularizer applies to all points, both labeled and unlabeled, which allows $\mathbf{p}_\theta(\mathbf{x}_i)$ to potentially diverge from $\mathbf{t}_i$ in the presence of noisy labels.

The third term in Equation 1 is an entropy regularizer encouraging *higher* entropy output distributions. We use the KL divergence between the posterior distribution and a uniform distribution for consistency in our objective and so the optimization always has a lower bound of 0. To discourage degenerate solutions where all unlabeled points are given the same label, and because MLPs are often very confident in their predictions, favoring higher entropy can be important, especially near decision boundaries. This is especially true if the graph contains separate connected components, some of which may have few or no labeled samples.

The last term in Equation 1 is a standard ℓ_2 regularizer on the model parameters (often called weight decay in the traditional MLP literature). Along with the entropy regularizer, we include this term in our objective as it gives some insurance against the case when too many parameters are available for a given amount of training data. This is especially important at the first layer of the MLP.

2.1. Relationship to other work

While our objective function is novel, this is not the first paper to provide a SSL extension for MLPs. In [20], the authors propose a squared-loss objective, which is not as well suited to probabilistic classification [9]. Our objective, by contrast, starts with a cross-entropy loss function on softmax outputs (equivalent to using the Kullback-Leibler (KL) divergence) and also uses a KL divergence-based graph term to incorporate information about nearby points. Another key difference is the amount of data used: while [20] makes use of one or two unlabeled points for each labeled point, our method always uses all the unlabeled data. [20] can, however, apply its graph regularization to hidden weight layers instead of only to the output layer.

The idea of using a graph regularizer while training supervised algorithms was first proposed in [1] and referred to as *Manifold Regularization* (MR). The Harmonic Mixtures algorithm [22] and also [20] fall into this category.

If $\mathbf{x}_i \in \mathcal{X}$ is the input space and $y_i \in \mathcal{Y}$ the output space, a classifier is a mapping $f : \mathcal{X} \rightarrow \mathcal{Y}$. Given $\mathcal{D}^\ell$ and $\mathcal{D}^u$, MR defines the optimal mapping as:

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}_k} \left(\frac{1}{l} \sum_{i=1}^{l} \mathcal{L}(\mathbf{x}_i, y_i, f) + \gamma_A \parallel f \parallel_k^2 + \frac{\gamma_I}{n^2} \sum_{i=1}^{n} \omega_{ij} \big(f(\mathbf{x}_i) - f(\mathbf{x}_j)\big)^2 \right). \quad (2)$$

Here, $\mathcal{L}$ is a loss function, $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a Mercer kernel, and $\mathcal{H}_K$ the associated reproducing kernel Hilbert space with norm $\| \cdot \|_K$. When $\mathcal{L}$ is convex, the overall objective is also convex. The representer theorem may be extended to show that $f^*(x) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}, \mathbf{x}_i)$. Thus, solving for f^* corresponds to computing the optimal α_i^*'s. When $\mathcal{L}$ is squared-loss, the above approach is referred to as *Laplacian Regularized Least Squares* (LapRLS), and when hinge-loss is used, we get *Laplacian Support Vector Machines* (LapSVM). LapRLS admits a closed form solution, i.e., the α_i's may be obtained in a single step that involves inverting a matrix of size $n \times n$.

Despite some similarities between our proposed approach and MR, there are several important differences. First, by using an MLP to learn f, our approach is inherently parametric in nature. In contrast, MR is inherently non-parametric, potentially requiring the data to be stored for evaluation. While any samples for which $\alpha_i = 0$ are not necessary, in practice a large number of α_i are often non-zero, although this is at least somewhat kernel-dependent.

Next, the graph regularization term in the case of MR, as with [20], is based on squared-loss which is theoretically optimal under only a Gaussian assumption over differences in classifier outputs. KL divergence is a more natural measure of similarity between posteriors as it is based on relative rather than absolute error. KL divergence is also asymptotically consistent w.r.t. the underlying probability measures.

A third difference is that, as stated above in the case of MR, a convex $\mathcal{L}$ implies convexity of the overall objective. Because the MLP loss function is not convex, neither is $J(\theta)$. Yet MLPs are still used for many tasks and, as shown in section 5, our model still trains effectively.

Finally, and this is an important goal of this work, MR does not scale to the large data sets commonly seen in speech applications. The closed form solution of MR requires inverting a very large dense $(n \times n)$ matrix, rendering it impractical for many problems.

The Harmonic Mixtures algorithm [22] is similar to the proposed approach in that they use a graph regularizer while training a Gaussian mixture model (GMM). As in MR, the graph regularizer here is squared-loss based but their objective is *not* convex. However, GMMs are generative models, and they use maximum likelihood to learn the parameters while our objective is inherently discriminative.

Finally, we note that, while in this work we use the KL-divergence between distributions, any valid measure of similarity between probability distributions can be used. And to reinforce a point made earlier, the first term in $J(\theta)$ may be replaced by a loss function corresponding to any supervised learner leading to a semi-supervised version of that learner.

3. Model Optimization

Lacking a closed form optimal solution, we use stochastic gradient descent to optimize our MLPs in all cases. Combined with efficient derivatives, this allows the model to easily generalize to large problems. In all cases, $\theta = (w^{ho}, w^{ih})$, a pair of matrices corresponding respectively to the hidden-to-output weights and the input-to-hidden weights of the MLP. We use the symbol w to refer to MLP weights, contrasted with ω to refer to graph edge weights.

Analyzing Equation 1, we can break terms into entropy and cross entropy components as $D(a \parallel b) = H^c(a, b) - H(a)$ where we define cross entropy as $H^c(a, b) = -\sum_i a_i \ln b_i$. For a problem with K classes, doing so gives[1]:

$$J = D(\mathbf{t}_i \parallel \mathbf{p}_i) + \gamma \sum_{j=1}^{n} \omega_{ij} H^c(\mathbf{p}_i, \mathbf{p}_j) - (\kappa + \gamma \sum_{j=1}^{n} \omega_{ij}) H(\mathbf{p}_i) + \kappa \log K + \lambda \|\theta\| . \quad (3)$$

For hard labels and $\gamma = \kappa = 0$, differentiating with respect to MLP weights gives standard back propagation. When $\gamma, \kappa > 0$, the cross entropy term in Equation 3 unsurprisingly requires that we propagate errors not only for the current sample point but for

[1]For notational simplicity, we henceforth use the notation $\mathbf{p}_i = \mathbf{p}_\theta(\mathbf{x}_i)$ where the dependence on the parameters θ and on the i^{th} input sample is implicit.

each of its graph neighbors as well. The use of "error" in this case may be misleading relative to standard MLP training: the value used at each node is not simply the distance to the target value, as shown next. Note that for these derivatives, we assume an extra 1 is appended to both the input vector as well as the hidden layer to accommodate bias shifts.

We start by considering the derivative of the entropy term $H(\mathbf{p}_i)$ with respect to the hidden-to-output weights w^{ho}_{km} and input-to-hidden weights $w^{ih}_{m\ell}$. Because only $\mathbf{x}_i$ is involved in this term, we will drop the subscript i.

$$\frac{\partial H(\mathbf{p})}{\partial w^{ho}_{km}} = -z_m(p_k \log p_k + p_k H(\mathbf{p}))$$
$$\frac{\partial H(\mathbf{p})}{\partial w^{ih}_{m\ell}} = -z_m(1-z_m)x_\ell \sum_k w^{ho}_{km}(p_k \log p_k + p_k H(\mathbf{p}))$$

where z_m is the hidden layer output after applying a sigmoid.

Next we have cross entropy, $H(\mathbf{p}_i, \mathbf{p}_j)$. This derivative itself decomposes into two terms: One depends on the current point $\mathbf{x}_i$ and its hidden layer activations and the other on neighbor $\mathbf{x}_j$ and its hidden layer activations. The updates are:

$$\frac{\partial H^c(\mathbf{p}_i, \mathbf{p}_j)}{\partial w^{ho}_{km}} = (p_{jk} - p_{ik})z_{jm} + \\ - (p_{ik} \log p_{jk} + p_{ik} H^c(\mathbf{p}_i, \mathbf{p}_j))z_{im} \quad (4)$$

$$\frac{\partial H^c(\mathbf{p}_i, \mathbf{p}_j)}{\partial w^{ih}_{m\ell}} = z_{jm}(1-z_{jm})x_{j\ell} \sum_k w^{ho}_{km}(p_{jk} - p_{ik})$$
$$- z_{im}(1-z_{im})x_{i\ell} \sum_k w^{ho}_{km}(p_{ik} \log p_{jk} + p_{ik} H^c(\mathbf{p}_i, \mathbf{p}_j)).$$

Note that the first term in Equation 4 is like the standard backpropagation error term, except that there is a difference between the posteriors for each output rather than between a posterior and its target vector, an intuitively appealing result considering the goal of the graph term. The second term comes about since both $\mathbf{p}_i$ and $\mathbf{p}_j$ are functions of the weights, unlike the target vector in MLP training.

4. Experimental Framework

We performed experiments on two data sets. The first was a portion of the Vocal Joystick (VJ) vowel corpus [8], a database of speakers uttering vowel sounds[2]. The data was collected for The Vocal Joystick project [2], an assistive device to allow individuals with motor impairments to use their voice for continuous control of devices such as a mouse pointer or robotic arm. The other corpus was TIMIT [6], a well-known corpus for phone classification consisting of phonetically balanced read English sentences. In contrast to the VJ corpus, TIMIT has many more classes; we used the standard 39-class variant [12].

The majority of the VJ recordings are monophthongs, a speaker uttering a single vowel in isolation; there is no surrounding linguistic context. Utterances were made with rising, level and falling pitch contours, crossed with quiet, normal and loud amplitudes, and also amplitude sweeps which start loud and become quiet or vice versa. The VJ corpus seems ideal for SSL as it samples points in the vowel triangle [10, 7], defined primarily by the first two formants, which are very hard to estimate accurately. As a result, it is reasonable to expect that the data come from a low-dimensional manifold [1, 19] embedded in the feature space. We use the same training, development and test sets specified in [14, 13].

For VJ, the training, development and test sets had roughly $220k, 41k$ and $90k$ samples, respectively. For TIMIT, those numbers were $1.4M, 124k$ and $515k$, respectively. For TIMIT, the development set was included (with all points unlabeled) in the training set. MLPs on VJ data were trained to 0.1% convergence, and to 1% convergence on TIMIT.

Results on additional corpora and more mathematical background can be found in [17].

4.1. Features and Graph Affinity Values

Features were 26-d Mel frequency cepstral coefficients [14]: 13 coefficients and single deltas. We performed mean and variance normalization assuming a diagonal covariance matrix, and used a sliding window over 7 adjacent frames as the input to the classifiers based on results on the VJ corpus in [13], yielding 182-d input vectors. For consistency, we used the same feature extraction for TIMIT, although we applied per-utterance mean and variance normalization.

Graphs were constructed over the training data using $\mathcal{K}$-nearest neighbors ($\mathcal{K}$-NN) based on Euclidean distance. We then applied a radial basis function (RBF) kernel with width σ so that our affinity matrix is constructed with values $\omega_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{2\sigma^2}}$. Here $\mathcal{K}$ and σ are hyperparameters; we used $\mathcal{K} = 20$ for VJ and $\mathcal{K} = 10$ for TIMIT. The value of σ was tuned on the development set over the set $d_i/3,\ i \in \{1, 2, 3, 4, 5\}$ where d_i is the average distance between each node and its i^{th} nearest neighbor. The values for γ (in Equation 1) were obtained by a search over $\left[10^{-6}, 10^{1}\right]$ in multiplicative steps of 10^1; κ varied over 0 and the range $\left[10^{-6}, 10^{0}\right]$ in multiplicative steps of 10^2. To limit the size of the hyperparameter search while tuning models, we used ℓ_2 regularization coefficients and hidden layer sizes tuned on the fully supervised training set with no graph. For the VJ corpus, we used an MLP hidden layer of size 50, also based on tuning experiments. For TIMIT, the hidden layer was of size 500, chosen to give reasonable performance but still allow for reasonable training time.

5. Results

In both cases, we created a SSL problem by randomly dropping labels from samples in the training set. We always used all samples; the only difference is the number of *labels* used. Samples whose labels were dropped are simply treated as unlabeled. This gives a better view of how our algorithm compares to a baseline MLP at various labeled to unlabeled data ratios.

Although we would have preferred to test our model against LapRLS or LapSVM, training and evaluation time of those models was prohibitive for corpora of this size.

5.1. Vowel Classification

Results from the Vocal Joystick corpus appear in Figure 1. In all cases here, error rates were calculated via a variant on 6-fold cross validation. The SSL-MLP shows an improvement over the baseline MLP for all values of l (all statistically significant at the $p < 0.0001$ level). The margin between models increases as more labels are removed on the development set. And despite a significant degree of mismatch between development and test sets, the gains still hold up on the test set. Also note that the SSL-MLP provides a win even when all labels are used.

[2]The corpus is freely available online: web search for "Vocal Joystick vowel corpus"

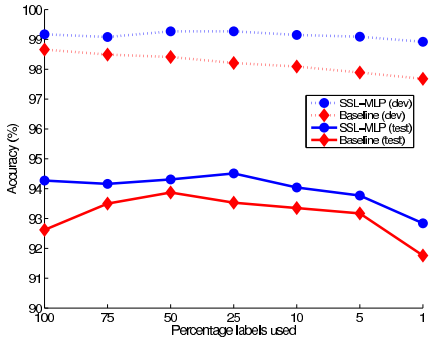

Figure 1: Vocal Joystick vowel data (4-class).

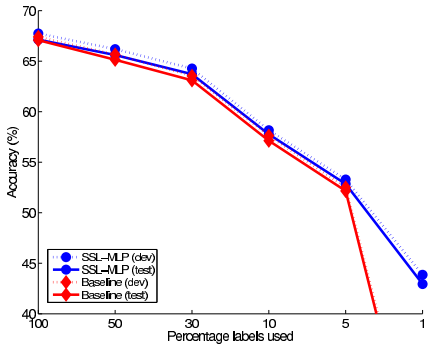

Figure 2: TIMIT data (39-class), differences are statistically significant everywhere ($p < 0.0001$) except the fully labeled case.

5.2. TIMIT Phone Classification

TIMIT results (frame-based) appear in Figure 2. We see a small and significant improvement in accuracy over most of the range on the development set. The gap becomes much larger when using only 1% of the labels. Shifting to test results, we see that the difference holds up except when using all labels. In the test case, the difference is statistically significant at $p < 0.0001$ in all cases (owing to the large number of test examples even though it is not obvious from the plot), except the fully labeled case. The results show a clear benefit from the SSL-MLP, especially when we have a small number of labeled points, a typical scenario for the SSL setting.

6. Conclusions and Discussion

We have introduced a novel objective function that expresses the cost of the current state of a classifier over both labeled and unlabeled data and have applied it to training multi-layered perceptrons. Our results are encouraging especially when using only a small percentage of the labels in the training set, and results on the VJ corpus shows how the graph can help provide performance gains even on fully labeled data.

The results on TIMIT are a clear demonstration that the model scales well — and when using little labeled data, and even with widely varying class priors, the graph regularizer is able to help the model learn more effectively. In that case, we see the real power of this model: smoothing over the graph-induced manifold helps the classifiers retain higher accuracy with many fewer labels, the ultimate goal of SSL.

While our work here focused exclusively on MLPs, preliminary experiments with a very different classifier, the Ratio Semi-definite Classifier (RSC) mentioned earlier, show positive results on the VJ corpus as well. Consequently, our objective is general, and perhaps more importantly, it is convex in the collection of distributions $\{\mathbf{p}_i\}_i$. When applied to an MLP the training procedure is not convex of course, but there are many composition rules that preserve convexity [4], and in such cases it would be relatively easy to apply the objective to a different family of classifier so as to obtain a convex parametric SSL algorithm. The objective could be applied to any classifier that expresses its classification preference as a posterior probability distribution, even if not convex.

In future work, we would like to explore the sensitivity of the hyperparameters to the amount of labeled and unlabeled data. Additionally, by allowing the use of unlabeled data, we can perform a better analysis of the diphthongs included in the Vocal Joystick vowel corpus, examining the path of a diphthong through the vowel space. Finally, the potential improvement to Tandem acoustic modeling with our SSL-MLP seems worth pursuing. There are many options with which to experiment and we expect improved performance as we become more familiar with this new model.

7. References

[1] M. Belkin, P. Niyogi, and V. Sindhwani. On manifold regularization. In *Proc. Int'l Workshop on Artificial Intelligence and Statistics*, 2005.

[2] J. Bilmes et al. The Vocal Joystick: A voice-based human-computer interface for individuals with motor impairments. In *HLT-EMNLP*, 2005.

[3] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[4] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2006.

[5] D. P. W. Ellis, R. Singh, and S. Sivadas. Tandem acoustic modeling in large-vocabulary recognition. In *IEEE ICASSP*, pages I–517–520, May 2001.

[6] J. Garofolo et al. *TIMIT Acoustic-Phonetic Continuous Speech Corpus*. Linguistic Data Consortium, Philadelphia, PA, 1993.

[7] K. Johnson. *Acoustic & Auditory Phonetics*. Blackwell Publishing, 2nd edition, 2003.

[8] K. Kilanski et al. The Vocal Joystick data collection effort and vowel corpus. In *Interspeech*, Pittsburgh, PA, Sept. 2006.

[9] D. M. Kline and V. L. Berardi. Revisiting squared-error and cross-entropy functions for training neural network classifiers. *Neural Computing and Applications*, 14(4):310–318, Dec. 2005.

[10] P. Ladefoged and I. Maddieson. *The Sounds of the World's Languages*. Blackwell Publishers, 1996.

[11] Y. LeCun, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[12] K.-F. Lee and H.-W. Hon. Speaker-independent phone recognition using hidden Markov models. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 37(11), Nov. 1989.

[13] X. Li. *Regularized Adaptation: Theory, Algorithms and Applications*. PhD thesis, University of Washington, Seattle, WA, 2007. in preparation.

[14] X. Li, J. Bilmes, and J. Malkin. Maximum margin learning and adaptation of MLP classifiers. In *Interspeech*, Lisbon, Portugal, 2005.

[15] J. Malkin and J. Bilmes. Ratio semi-definite classifiers. In *IEEE Int'l Conf. on Acous., Speech and Signal Processing*, Las Vegas, NV, 2008.

[16] J. Malkin and J. Bilmes. Multi-layer ratio semi-definite classifiers. In *IEEE Int'l Conf. on Acous., Speech and Signal Processing*, Taipei, Taiwan, 2009.

[17] J. Malkin, A. Subramanya, and J. Bilmes. A semi-supervised learning algorithm for multi-layered perceptrons. Technical Report UWEETR-2009-0003, U. Washington Dept. of Electrical Engineering, 2009.

[18] B. Schölkopf and A. Smola. *Learning With Kernels*. MIT Press, Cambridge, MA, 2002.

[19] J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290, 2000.

[20] J. Weston, F. Ratle, and R. Collobert. Deep learning via semi-supervised embedding. In *Intl. Conf. on Machine Learning*, 2008.

[21] J. Zheng. Combining discriminative feature, transform, and model training for large vocabulary speech recognition. In *IEEE ICASSP*, May 2007.

[22] X. Zhu and J. Lafferty. Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In *Int'l Conf. on Machine Learning*, 2005.