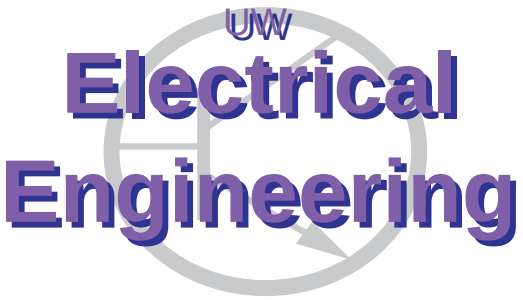
A Semi-Supervised Learning Algorithm for Multi-Layered Perceptrons

Jonathan Malkin, Amarnag Subramanya, Jeff Bilmes
{jsm,asubram,bilmes}@ee.washington.edu

Dept. of EE, University of Washington
Seattle, WA 98195-2500



UWEE Technical Report
Number UWEETR-2009-0003
May 2009

Department of Electrical Engineering
University of Washington
Box 352500
Seattle, Washington 98195-2500
PHN: (206) 543-2150
FAX: (206) 543-3842
URL: http://www.ee.washington.edu

A Semi-Supervised Learning Algorithm for Multi-Layered Perceptrons

Jonathan Malkin, Amarnag Subramanya, Jeff Bilmes
{jsm,asubram,bilmes}@ee.washington.edu

Dept. of EE, University of Washington
Seattle, WA 98195-2500

University of Washington, Dept. of EE, UWEETR-2009-0003
May 2009

Abstract

We address the issue of learning multi-layered perceptrons (MLPs) in a discriminative, inductive, multiclass, parametric, and semi-supervised fashion. We introduce a novel objective function that, when optimized, simultaneously encourages 1) accuracy on the labeled points, 2) respect for an underlying graph-represented manifold on all points, 3) smoothness via an entropic regularizer of the classifier outputs, and 4) simplicity via an ℓ_2 regularizer. Our approach provides a simple, elegant, and computationally efficient way to bring the benefits of semi-supervised learning (and what is typically an enormous amount of unlabeled training data) to MLPs, which are one of the most widely used pattern classifiers in practice. Our objective has the property that efficient learning is possible using stochastic gradient descent even on large datasets. Results demonstrate significant improvements compared both to a baseline supervised MLP, and also to a previous non-parametric manifold-regularized reproducing kernel Hilbert space classifier.

1 Introduction

Graph-based methods have been highly successful for semi-supervised learning (SSL). Initial graph-based methods were primarily *non-parametric* and *transductive* [29], learning to classify unlabeled test points simultaneously with learning a model for labeled points. While such transductive models can be extended to handle novel points using, for example, a weighted combination of nearest neighbors [31], this requires that we keep the entire training set around at all times. This is undesirable on two fronts: Computing the nearest neighbors at run-time can be computationally expensive, and in large applications it might not be feasible to retain the entire training set. As a result, there is a desire for *inductive* models which can be more compactly represented and which have a straightforward ability to generalize to novel points.

Another division between learning models is that of parametric vs. nonparametric. Nonparametric models are such that the number of parameters may vary with the amount of training data. Many nonparametric algorithms are used with kernel methods [26], and tend to return sparse results, but a poor match between the kernel and the decision surface can lead to results that are much denser. And in many cases, the sorts of nonlinear kernels we would like to use make solving the model prohibitively expensive on very large data sets. Nevertheless, nonparametric graph-based approaches to SSL such as [2] often perform quite well and are backed by useful theory.

Many of the initial SSL approaches, although not graph-based, were parametric [4]. A more recent example [9] used discriminatively trained generative parametric models for semi-supervised learning. There has been less work on combining parametric SSL with graph-based models, with [33] being an obvious exception which used a generative mixture model to capture local behavior and a graph to provide global structure. [31] and [6] present more thorough discussions of both these issues and SSL in general.

In this paper, we propose a new SSL training objective that is suitable for training parametric classifiers via numerical optimization. We apply this objective to the training of standard *Multi-Layer Perceptrons* (MLP) [3, 13], making

them suitable to the case where there are unlabeled as well as labeled training samples. We optimize the objective using stochastic gradient descent[18, 19], thereby yielding a simple and computationally tractable algorithm for the discriminative, inductive, multiclass, parametric, and semi-supervised learning of MLPs.

While our objective is general and applies to a variety of parametric classifiers (see Section 6), we utilize MLPs in this paper for a variety of reasons. MLPs are used ubiquitously in many real-world applications, are universal approximators [3], and have been said to be "the second best way of doing just about anything" [25]. While interest in MLPs has waned over the past decade, relatively recent results have shown that there is much less difference between MLPs and reproducing kernel Hilbert space (RKHS) classifiers than what was originally thought [8]. An MLP can be seen as a hyperplane classifier in the RKHS implicitly defined by the MLP's input-to-hidden layer mapping [22]. At the same time, MLPs are sufficiently flexible that they can, at least in theory, learn any mapping given sufficient training data and number of hidden units h [3] and thus the implicitly-defined RKHS can vary with a single discrete parameter h. Moreover, since the input-to-hidden mapping is learnt, even with a fixed h, an MLP spans an uncountable number of (not necessarily unique) RKHSs. It is sometimes said that an MLP can simultaneously learn a mapping from feature space to output space and learn the underlying kernel in a data-driven fashion. In addition, standard MLP training objectives also mirror that of RKHS classifiers (an error term, a regularizer, and a tradeoff coefficient relating the two when summed). What is perhaps most different between the two is the optimization procedure: A convex optimization procedure under a kernel classifier vs. a non-linear optimization procedure with an MLP.

Given the general utility of MLPs, it is unfortunate that there has not, before now, been a corresponding SSL algorithm for their training to exploit what is becoming a common scenario in machine learning: A relative plethora of easily obtained unlabeled data compared to the dearth of labeled data that is expensive and time-consuming to acquire. To the best of our knowledge, this work is the first attempt at training standard MLPs in a semi-supervised manner where the parameters are jointly optimized using both the labeled and unlabeled data and where the KL-divergence loss is augmented in a natural way. Although there have been other proposed methods for the semi-supervised training of MLPs, they have not incorporated or extended a KL divergence-based loss function. As is common, both labeled and unlabeled data are assumed to be embedded within a low-dimensional manifold expressed by a graph.

As our results show, this forms a potent combination leading to improved performance over the state-of-the-art on a number of both small and large real-world data sets. By using a parametric model, we can limit the size of our model which can be useful with extremely large training sets since there is no memorization of the training set. We can also bound the size of our solution — with a non-parametric approach, even when sparse, it can be difficult to guarantee that the solution will abide by certain hard complexity limits. Further, our approach is naturally inductive, as we learn not the labeling of a set of pre-specified unlabeled points, but rather a mapping from input to output that can be applied anywhere in the input space. We also stress that, owing to the MLP's multiclass abilities via a soft-max output non-linearity, our method is naturally multiclass. Lastly, since the optimization of a conditional likelihood underlies our objective, our approach is discriminative in nature.

2 Parametric SSL Objective for Discriminative Classifiers

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 $w_{ij} \in \mathbf{W}$ to denote the weight of the edge between vertices i and j. For information on how to construct the graph, see section 4.3.

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

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

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 tradeoff parameters, which include $\gamma, \kappa, \lambda \geq 0$, is discussed in section 5. In the following 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 distributions. 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 $\mathbf{t}_i$ quite significantly 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. Because of a tendency towards overconfidence in many statistical classifiers, and to discourage degenerate solutions where all unlabeled points are given the same label, we want an explicit bias towards higher entropy posteriors. This is especially true near potential decision boundaries, where it may be more useful to encourage the expression of uncertainty. It is also useful in separate connected (or pendent) graph components containing few or no labeled samples. Note that entropy regularization could be used in purely supervised training, but it might be more useful when unlabeled data is present for the reasons given above.

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. Unlike a non-parametric approach, where the model complexity itself can adjust in accordance with the amount and quality of the training data, in our case for a fixed number of hidden units, the model family is fixed during training.

2.1 Relationship to other work

The idea of using a graph regularizer while training supervised algorithms was first proposed in [2] and referred to as *Manifold Regularization* (MR). The Harmonic Mixtures algorithm [33] and also [30] falls 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 $f : \mathcal{X} \to \mathcal{Y}$. Given $\mathcal{D}^\ell$ and $\mathcal{D}^u$, MR defines the optimal mapping as:

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

Here, $\mathcal{L}$ is a loss function, $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a Mercer kernel, and $\mathcal{H}_K$ the associated RKHS with norm $\parallel . \parallel_K$. 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. It is important to note that the summation in the expansion of f^* extends over both labeled and unlabeled points. 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 $n \times n$.

Clearly, MR as proposed in [2] is similar to our proposed objective in that it is a sum of three terms, one of which penalizes against not respecting the labels, another term regularizes w.r.t. the graph, and a final term which acts as a regularizer over f. The work in [30] also has clear similarities to our proposed model. And all three methods are inductive. Our approach, however, differs from the other two in several important respects.

Starting with a comparison to MR, we use an MLP to learn f, so our approach is inherently parametric in nature. In contrast, MR is inherently non-parametric, potentially requiring that both $\mathcal{D}^\ell$ and $\mathcal{D}^u$ be stored for testing (classifying a new input). While any samples for which $\alpha_i = 0$ are not necessary, in practice a large number of α_i are non-zero (see Section 5). On a related note, a transductive learner can be used to produce an inductive one by taking a weighted average of the outputs of a new point's nearest neighbors, but the learner is still inherently non-parametric. Such a learner can also be used to produce a parametric one by labeling the unlabeled training samples, and then training a fully-supervised parametric classifier. But this later approach does *not* jointly optimize the labels on the unlabeled data and the parameters of the model.

Next, the graph regularization term in the case of MR is based on squared-loss which is theoretically optimal under only a Gaussian assumption over differences in classifier outputs. When measuring the similarity between posteriors, KL-divergence is a more natural measure as it is based on relative rather than absolute error. Further KL-divergence

is asymptotically consistent w.r.t. the underlying probability measures. Of course, a squared loss is appropriate when graph-regularizing the weights as in [30].

A third difference is that, as stated above in the case of MR, a convex $\mathcal{L}$ implies convexity of the overall objective. In our case, $J(\theta)$ is *not* convex because of the non-convexity of the supervised MLP loss function. Yet, as shown in section 5, our model trains quite efficiently.

Our approach is also *naturally* multiclass whereas LapRLS and LapSVM often rely on potentially suboptimal methods such as one-vs-rest classification to produce a multiclass classifier. However, as shown in [11], it is possible to use multiclass SVMs with MR leading to a true multiclass approach, albeit with increased computational complexity.

Finally, while MR allows a user to choose the appropriate kernel k, in practice, the choice is not always straightforward. In many cases, it might be desirable to allow the training algorithm to learn the optimal kernel in a data-driven fashion. As mentioned above, in an MLP, the mapping from the input to the hidden nodes can be considered to be implicitly specifying a kernel learned from training data. In our case, this kernel is not only based on the labeled data but also exploits the information conveyed by the unlabeled samples. As a result, our approach is also related to the algorithm proposed in [32]. Of course, the MLP approach still requires choosing the number of hidden units, something that, like choosing the appropriate kernel, might require trial and error. As mentioned above, however, employing a final ℓ_2 regularizer can mitigate this problem to some extent.

Compared to Weston *et al*. [30], the biggest difference is our use of KL divergence in our objective. Without a graph, using a cross entropy loss function is typically faster to train than squared loss and produces more accurate posterior probabilities [16]. And while [30] proposes the addition of a graph regularizer on any layer in a deep network architecture (as suggested above), their experiments apply it only to the output layer.

Perhaps more subtle but equally important, Weston's work uses a fundamentally different approach to training than does our work. In their paper, gradients are calculated for each (random) training sample as per standard stochastic gradient descent training. Then, for each graph regularizer, gradients are calculated for one random pair of neighbors and one random unlabled example. The portion of points used in training will therefore vary based on the ratio of labeled to unlabled samples in the training set. By contrast, we use all information from all points on every pass through the training data, as shown in our derivatives in Section 3.

The Harmonic Mixtures algorithm [33] 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. In our case, the objective is inherently discriminative.

Entropy regularization [12] is similar to our work in that it attempts to maximize conditional likelihood but has a few crucial differences: First, it attempts to minimize entropy of the unlabeled samples. The motivation for this is that decision boundaries should avoid high density regions and unlabeled points are likely to lie in high density regions. In our case, we encourage higher entropy. And second, entropy regularization does *not* make use of a graph regularizer. Our results in section 5 demonstrate that using a graph regularizer and encouraging higher entropy leads to improved performance.

We lastly note that our proposed objective can be applied to models other than MLPs. Also, while in this work we use the KL-divergence between distributions, any valid measure of similarity between probability distributions can be used. In fact, the first and/or second terms 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

As evidenced in part by the term in our objective (Equation 1) encouraging higher entropy, we prefer models that will not have a bias towards overconfidence in areas of low training data density (labeled or unlabeled). Additionally, we prefer models that reflect the underlying ambiguity in the data, especially near decision boundaries. Higher confidence may be acceptable in areas with a high density of labeled points, but is less desirable outside of those regions. While not a major goal of this work, our hope is that the new objective will provide these properties while retaining high accuracy.

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 above 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 $H^c(a,b) = -\sum_i a_i \ln b_i$. We consider each point $\mathbf{x}_i, i \in 1,..,n$ individually and in random order while using stochastic gradient descent. For a problem with K classes, doing so gives[1]:

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

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 each of its graph neighbors as well. The use of "error" in this case, however, is perhaps 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 output of the hidden layer 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:

$$\frac{\partial H^c(\mathbf{p}_i, \mathbf{p}_j)}{\partial w^{ho}_{km}} = (p_{jk} - p_{ik}) z_{jm} - (p_{ik} \log p_{jk} + p_{ik} H^c(\mathbf{p}_i, \mathbf{p}_j)) z_{im} \tag{4}$$
$$\frac{\partial H^c(\mathbf{p}_i, \mathbf{p}_j)}{\partial w^{ih}_{m\ell}} = z_{jm}(1 - z_{jm}) x_{j\ell} \sum_k w^{ho}_{km} (p_{jk} - p_{ik}) - z_{im}(1 - z_{im}) x_{i\ell} \sum_k w^{ho}_{km} (p_{ik} \log p_{jk} + p_{ik} H^c(\mathbf{p}_i, \mathbf{p}_j)).$$

Note that the first term in Equation 4 is like the standard back-propagation 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 conducted a number of experiments using several data sets. First, we used three corpora from the UCI Machine Learning Repository [1], comparing our method against both a standard MLP and LapRLS. Sample counts for all corpora appear in Table 1.

To demonstrate the scalability of our proposed method, we also looked at several larger corpora. One was a portion of the Vocal Joystick (VJ) vowel corpus [15], a database of speakers uttering vowel sounds[2]. The other large corpus was TIMIT [10], a well-known corpus for phone classification and a task for which MLP classifiers achieve state-of-the-art performance for frame-by-frame independent classification. Both of these corpora will be described in more detail below.

LapRLS was unable to handle these two data sets due to their large size, so we only compare results to a standard MLP. And while our SSL-MLP training time was not short (several hours on VJ, a few days on TIMIT, using a single

[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.

[2] The corpus is available online for free at
http://ssli.ee.washington.edu/vj/corpus.html

core machine), it is certainly feasible, especially considering that the end model can be evaluated very quickly using fast matrix libraries. Moreover, as mentioned above, the parametric case has bounded computation at test time, unlike a non-parametric case where the test-time computational cost is controlled only indirectly by the accuracy-regularization coefficient.

4.1 Data Corpora: UCI Data Sets

We used vehicle silhouette, Japanese vowels and pendigits from the UCI Machine Learning Repository corpora. For pendigits, we used the suggested train, development and test sets for writer-independent testing. On Vehicle Silhouettes (which originates from the Turing Institute, Glasgow Scotland), we used 65% of frames as training, 15% for development and 25% for testing. And for Japanese vowels, we used only the middle sample from each time series, giving a total corpus size of 640. We then created a development set by randomly selecting 20% of the frames in the test set. In each case we varied the number of labeled samples l in the training set.

4.2 Large Corpora: VJ Vowels and TIMIT

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 verse. The VJ corpus is ideal for SSL in that phoneticians often refer to a *vowel triangle* [17, 14], a space created primarily by the first two resonant frequencies of the vocal tract — which are very hard to estimate accurately — as being the primary determinants of vowel quality, the specific vowel being uttered. According to this view, vowels lie in a continuous space and, based on language and dialect biases, humans employ categorical perception to create decision boundaries in vowel space. As a result, the notion that this real-world data lies on a manifold [2, 24, 28] is supported by phonetic theory as well. We use the same training, development and test sets specified in [22, 21].

TIMIT is a standard corpus for phonetic classification consisting of phonetically balanced read English sentences. In contrast with the VJ corpus, TIMIT has many more classes; we used the standard 39-class variant [20].

4.3 Features and Graph Affinity Values

For UCI corpora, we used the provided features, applying only mean and variance normalization to the data. For the non-UCI corpora, features were 26-d Mel frequency cepstral coefficients [22]: 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 [21], yielding 182-d input vectors. For consistency, we used the same feature extraction for TIMIT, although we applied per-utterance mean and variance normalization.

Next we describe graph construction and hyperparameter selection for the proposed approach. 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 $\mathcal{K} = 10$ for the all the UCI corpora and $\mathcal{K} = 20$ for VJ and TIMIT. The value of σ was tuned 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 was obtained by a search over $\left[10^{-6}, 10^{1}\right]$ in multiplicative steps of 10^1 and over 0 and $\left[10^{-6}, 10^{0}\right]$ in multiplicative steps of 10^1 for κ (see equation 1). 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. The size of each hidden layer appears in Table 1.

4.4 Comparison model: LapRLS

Where feasible, we chose to compare the proposed approach against LapRLS as it has been shown to yield state-of-the-performance in SSL. LapRLS performs similarly to LapSVMs [7]. The graph here was constructed in a manner similar to that described in section 4.3 with the only difference that σ was set to the mean distance between adjacent nodes in the graph. The base kernel was also a RBF of width σ_s. The hyperparameters (see equation 2) were tuned were tuned per the recipe in [7] over the following sets $\gamma_A \in \{$1e–8, 1e–6, 1e–4, 1e–2, 1, 100$\}$, $r \in \{$1e–4, 1e–2, 1, 100$\}$, $\gamma_I = r\gamma_A$, $\sigma \in \{\frac{\sigma_0}{8}, \frac{\sigma_0}{4}, \frac{\sigma_0}{2}, \sigma_0, 2\sigma_0, 4\sigma_0\}$ on the development set. The optimal values of these hyperparameters were

Table 1: Sizes in frames of data sets used. SSL tasks were created by ignoring labels to create various ratios of ℓ to u such that $\ell + u = n$. TIMIT dev. set is a subset of train in this case. h indicates the number of hidden units in the MLP used with each model.

Corpus	n	Dev.	Test	h
Japan. Vowel	270	74	296	25
Veh. Sil.	508	128	210	25
Pendigits	3747	1874	3498	45
VJ	220k	41k	90k	50
TIMIT	1.4M	124k	515k	500

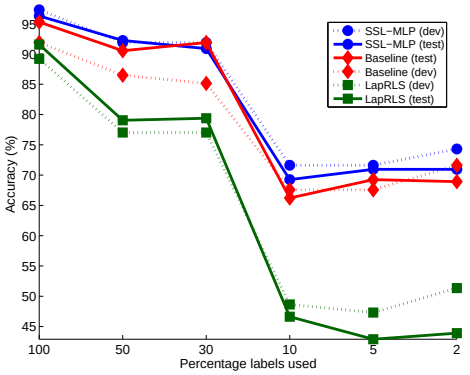


Figure 1: Japanese Vowels (9-class).

then used to generate results for the test set. Since all problems considered are multiclass, a *one-vs-rest* strategy was used.

5 Results

In all cases, we created a SSL problem by randomly dropping labels from samples in the training set. We used all samples in all cases; the only difference is the number of labels used. In other words, the 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.

5.1 UCI Corpora

Figures 1 through 3 show results of our SSL task on the three UCI corpora. On the smallest, Japanese vowels (Figure 1), we find that the SSL-MLP provides only a modest improvement over the baseline MLP, although both consistently *outperform* LapRLS. Vehicle silhouettes (Figure 2) proves a fairly difficult task with 18-d inputs yet few training samples, leading to a steady fall-off as fewer labels are used. Again, both MLPs outperform LapRLS, with the SSL-MLP again posting only a modest improvement over the baseline.

Finally, Figure 3 shows the largest of the small training sets, pendigits. Here, we find a much more gradual fall-off in accuracy initially, with all three methods performing similarly when many labels are used. As the number of labels drops, however, we start to see clear differences between the three classifiers, with the graph helping the SSL-MLP achieve a much smaller decrease in accuracy.

These results shows that there is a consistent improvement of the SSL-MLP over LapRLS. That coupled with the SSL-MLP's bounded computation at test time (e.g., for all datasets, in the LapRLS case, we found that $\alpha_i^* \neq 0, \forall\, 1 \leq i \leq n$, so the computation could continue to grow with the size of the data), the results show clear advantages to our approach. We see also a clear benefit of the manifold regularization over the baseline MLP. We next evaluate much larger data sets, for which it was not feasible to run LapRLS.

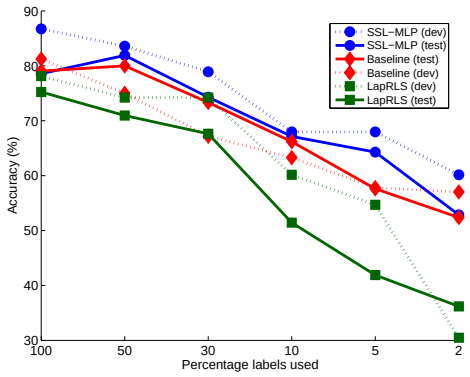


Figure 2: Vehicle Silhouettes data (4-class).

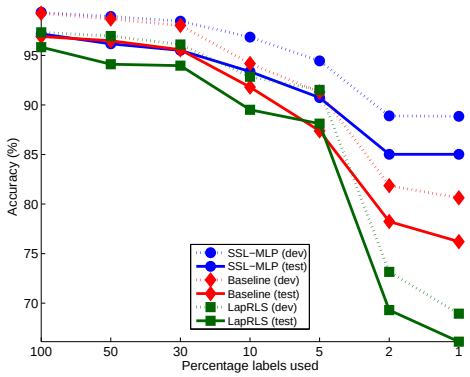


Figure 3: Pendigits data (9-class)

5.2 Vowel Classification

Results from the first of our large corpora appear in Figure 4. In all cases here, error rates were calculated via *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 moderate 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!

5.3 TIMIT Phone Classification

TIMIT results appear in Figure 5. 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 of 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 continuous differentiable objective function that expresses the cost of the current state of a classifier over both labeled and unlabeled data and have applied it to the multi-layered perceptron. The results on the

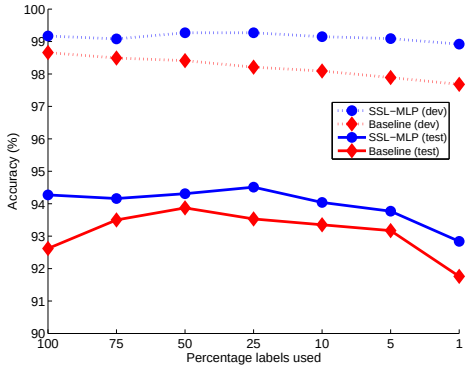


Figure 4: Vocal Joystick vowel data (4-class).

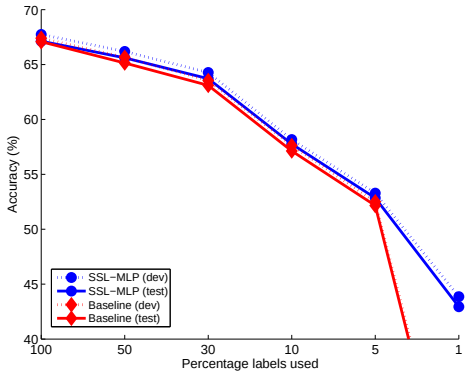


Figure 5: TIMIT data (39-class), everywhere statistically significant differences ($p < 0.0001$) except fully labeled case.

UCI sets demonstrate the utility of the SSL-MLP as the number of labels becomes small. Meanwhile, the VJ corpus shows how the graph can help provide performance boosts, even on fully labeled data. The results on TIMIT, while modest, are a clear demonstration that the model scales to very large data sets — and when using little labeled data, and even with widely varying class priors, the graph regularizer is able to help the model learn much 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.

There are obvious variations to our approach. For instance, we utilized herein only undirected graphs, although the objective applies equally well to directed graphs (where the inherent asymmetry of the KL-divergence, washed away via an undirected graph, might yield improved results or theory). Another approach would consider alternatives to the RBF to obtain graph weights, such as a cosine distance or an asymmetric distance (e.g., relative error).

MLPs are a reasonable initial parametric model to evaluate our objective function. Nevertheless, it should be noted that 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 [5], 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. We plan in the future to develop new training procedures for semi-supervised Bayesian networks and deep network architectures [23, 27, 30].

References

[1] A. Asuncion and D. Newman. UCI machine learning repository, 2007.

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

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

[4] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT: Proc. Conf. on Computational Learning Theory*, New York, NY, 1998.

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

[6] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[7] O. Chapelle, B. Scholkopf, and A. Zien, editors. *Semi-Supervised Learning*, chapter Analysis of Benchmarks. MIT Press, Cambridge, MA, 2006.

[8] R. Collobert and S. Bengio. Links between perceptrons, MLPs and SVMs. In *Intl. Conf. on Machine Learning*, 2004.

[9] A. Fujino, N. Ueda, and K. Saito. A hybrid generative/discriminative approach to semi-supervised classifier design. In *AAAI-05*, 2005.

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

[11] A. Goldberg, X. Zhu, and S. Wright. Dissimilarity in graph-based semi-supervised classification. In *AISTATS*, 2007.

[12] Y. Grandvalet and Y. Bengio. *Semi-Supervised Learning*, chapter Entropy Regularization. MIT Press, 2007.

[13] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, New York, NY, 2001.

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

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

[16] 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.

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

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

[19] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In G. B. Orr and K.-R. Müller, editors, *Neural Networks: Tricks of the trade*. Springer, 1998.

[20] 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.

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

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

[23] M. Ranzato, Y.-L. Boureau, and Y. LeCun. Sparse feature learning for deep belief networks. In *Advances in Neural Information Processing Systems 18*, 2007.

[24] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290, 2000.

[25] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach 2nd Edition*. Prentice Hall, 2003.

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

[27] I. Sutskever and G. E. Hinton. Deep, narrow sigmoid belief networks are universal approximators. *Neural Computation*, 20:2629–2636, 2008.

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

[29] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, NY, 1995.

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

[31] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

[32] X. Zhu, J. Kandola, Z. Ghahramani, and J. Lafferty. Nonparametric transforms of graph kernels for semi-supervised learning. In L. Saul, Y. Weiss, and L. Bottou, editors, *NIPS*, 2005.

[33] 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.