ACOUSTIC CLASSIFICATION USING SEMI-SUPERVISED DEEP NEURAL NETWORKS AND STOCHASTIC ENTROPY-REGULARIZATION OVER NEAREST-NEIGHBOR GRAPHS

Sunil Thulasidasan*†, Jeffrey Bilmes†

*Los Alamos National Laboratory
†Department of Electrical Engineering, University of Washington

ABSTRACT
We describe a graph-based semi-supervised learning method for acoustic data that uses a Deep Neural Network (DNN) combined with a stochastic graph-based entropic regularizer to favor smooth solutions over a graph induced by the data. We consider graph embeddings constructed from the input features and also from dimensionality-reduced encodings obtained from the bottleneck layer of a separate deep auto-encoder. We use a computationally efficient, stochastic graph-regularization technique that uses mini-batches that are consistent with the graph structure but that also provide enough data diversity for the convergence of stochastic gradient descent methods to good solutions. For this work, we focus on results of frame-level phone classification accuracy on the TIMIT speech corpus but our method is general and scalable to much larger data sets. Results indicate that our method significantly improves classification accuracy compared to the fully-supervised case when the fraction of labeled data is low, and it is competitive with other methods in the fully labeled case.

Index Terms— semi-supervised learning, graph-based learning, deep learning, speech recognition

1. INTRODUCTION
Semi-supervised learning (SSL) methods use both labeled and unlabeled data to improve learning performance [1] and are especially useful in situations where labeled data is scarce. Since unlabeled data can usually be collected in a fully automated, scalable way, SSL methods aim to leverage unlabeled data to improve prediction performance by exploiting the similarity between labeled and unlabeled data. A natural way to capture this relationship is via graphs where the nodes represent both labeled and unlabeled points and the weights of the edges reflect the similarity between the nodes [2]. The main idea behind graph-based SSL methods is that given a similarity metric, the objective function in graph-based SSL methods encourages similar (i.e., nearby) nodes to have the same label by imposing a graph-neighbor regularization; this is effective because it prefers the labels to be consistent with the graph structure (and the underlying manifold represented thereby). Graph-based SSL algorithms have been successfully applied to tasks such as phone and word classification in automatic speech recognition (ASR) [3, 4, 5, 6], part-of-speech tagging [7], statistical machine translation [8], sentiment analysis in social media [9], text categorization [10] and many others.

In this work, we describe algorithmic improvements for efficient and scalable graph regularization that can be applied to any parametric graph-based SSL framework. We use a fully parametric learner – a deep neural network – with an entropy regularizer over the graph induced by the data, a method that was first described in [3] in the context of a multi-layered perceptron (MLP) with one hidden layer. By sampling the data using graph partitioning, but at the same time preserving the statistical properties of the data distribution, and by stochastically regularizing over the graph, we are able to significantly outperform the original results even on an MLP, and make further improvements using a DNN. For the results reported in this paper, we limit our data-set to fixed length speech frames, only reporting frame-level phone classification accuracy on the TIMIT [12] speech corpus without using HMM-based decoding and n-gram language models. Our aim in this work is not to beat state-of-the-art ASR systems (which all use language models [13, 14, 15] and typically will have higher accuracy than the results presented here) but to demonstrate the efficacy of a computationally efficient technique that can potentially be used to improve ASR and other machine learning systems in a semi-supervised setting.

2. PARAMETRIC OBJECTIVE FOR GRAPH-BASED SSL CLASSIFIERS
Graph-based SSL techniques assume that data are embedded in some low-dimensional manifold in a higher dimensional ambient space, and that nearby nodes will likely have the same labels (the manifold and smoothness assumptions, respectively); the objective function in these methods thus impose a penalty when the output on nearby nodes differ. The general form of the loss function in graph-based SSL has the following form

\[ \sum_{i=1}^{L} l(y_i, f(x_i)) + \lambda \sum_{(i,j) \in E(G)} \omega_{i,j} g(f(x_i), f(x_j)) \]  

(1)

where \( f : \mathcal{X} \rightarrow \mathcal{Y} \) is the classifier mapping from input to output space. The first term in Equation 1 the supervised loss function calculated on the labeled points. \( l(.) \) can be a squared loss, hinge-loss or some measure of divergence between predictions and ground truth. The second term is the graph regularizer, where \( \omega_{i,j} \) captures the similarity between points \( x_i \) and \( x_j \), and where \( E(G) \) are the edges (pairs of nodes) of a graph \( G \). The function \( g(\cdot, \cdot) \) captures the discrepancy between output \( f(x_i) \) and \( f(x_j) \), incurring a large penalty when similar nodes have different output. Additional regularizers (such as the standard \( \ell_1 \) or \( \ell_2 \)) can also be applied to prevent overfitting.

Concretely, let \( \{(x_i, y_i)\}_{i=1}^{L} \) be the labeled training data and \( \{x_i\}_{i=L+1}^{L+u} \) be the unlabeled training data, where \( n = L + u \) so that we have \( n \) points in total. We denote by \( \Delta_M \) the \( M \)-dimensional probability simplex (i.e., the set of all distributions over \( M \) class labels). Let \( p_\theta(x_i) \in \Delta_M \) represent the output vector of posterior probabilities dictated by \( \theta \), the parameters of the classifier and \( t_i \in \Delta_M \) for \( 1 \leq i \leq L \) denote a probabilistic label vector for the \( i \)-th training

---

*An earlier version of this work appeared in [11]
sample. We also assume that the samples \( \{x_i\} \) are used to produce a weighted undirected graph \( G = (V, E, W) \), where \( \omega_{i,j} \in W \) is the similarity (edge weight) between samples (vertices) \( x_i \) and \( x_j \) (i and j). We use the objective function defined in \([16, 3]\), namely:

\[
J(\theta) = \sum_{i=1}^{l} D(t_i || \phi(x_i)) + \gamma \sum_{i,j=1}^{n} \omega_{i,j} D(\phi(x_i) || \phi(x_j)) + \kappa \sum_{i=1}^{n} D(\phi(x_i) || u) + \lambda \|\theta\|_2^2 ,
\]

(2)

where \( u \in \Delta_M \) is the uniform distribution and \( J(\theta) \) is the loss calculated over all samples. The weights \( \omega_{i,j} \) themselves are sparse, so that \( \omega_{i,j} = 0 \) when \( (i, j) \notin E(G) \). We use KL-divergence (denoted by \( D(\cdot || \cdot) \)) as our loss function since our output is a probability distribution over classes. The first term in the above equation is the supervised loss over the training samples, and the second term is the penalty imposed by the graph regularizer over neighboring pairs of nodes that favors smooth solutions over the graph. The third term is an entropy regularizer and favors higher entropy distributions since MLPs and DNNs are often very confident in their predictions which can lead to degenerate solutions; favoring higher entropy solutions counters this and is especially useful near decision boundaries. An alternative to regularizing against the uniform distribution is to regularize against a prior \( \tilde{p}(x_i) \) as done in [4], where \( \tilde{p}(x_i) \) are the outputs from a first-pass classifier trained in a supervised manner. We can easily incorporate this into our framework, though the work described in this report regularizes against only the uniform distribution. The final term in Equation 2 is the standard \( \ell_2 \) regularizer to discourage overfitting.

3. RELATED WORK

There have been several graph-based learning algorithms that make use of some version of the objective function described in the previous section [4, 16, 17, 18]. Label propagation, described in [17] forces \( f \) to agree with labeled instances by minimizing squared loss between predictions of nearby points. Measure propagation, described in [16] uses essentially the same objective function as in Equation 2 but in a non-parametric setting. Prior-regularized measure propagation [4] substitutes the uniform distribution in Equation 2 with a prior \( \tilde{p}(x) \) that comes from a supervised classifier prior to the SSL process, and has shown to work well on speech data. One of the early works to use a graph regularizer in a deep learning context is described in [19], where squared loss is used instead of KL-divergence. The algorithms described in this paper are most related to [3] but significantly improve upon the graph construction and parallel programming methodologies and also apply it to deep models. Our methods will generally work on any objective function with a graph regularizer.

4. GRAPH REGULARIZATION VIA GRAPH PARTITIONING

Like other graph-based SSL methods we induce a graph on the data by constructing a \( k \)-nearest neighbor (\( k \)-NN) graph where the edge weights are the Euclidean distance between the feature vectors. Since we are dealing with a non-convex objective function, and a moderately large data set (\( \approx 1 \) million training samples), we use stochastic gradient descent (SGD) to optimize our objective function. We also use mini-batches to improve the gradient quality, and further, use larger mini-batches (size set to 1024) for better computational efficiency. Traditional SGD methods require randomly shuffling of the data for good convergence before constructing the mini-batches; this, however, poses a problem for our objective function. To see this, consider the terms involving graph regularization from our objective function, calculated over each point:

\[
G_i = \gamma \sum_{j=1}^{n} \omega_{i,j} D(\phi(x_i) || \phi(x_j))
\]

For the graph regularization term to have any effect at all, the \( w_{ij} \)'s corresponding to the points in the mini-batch have to be non-zero. For a randomly shuffled data-set, given that the \( k \)-NN graph is very sparse (since each of the \( \approx 1 \) million points only has a little more than \( k = 10 \) neighbors), the chunk of the affinity matrix corresponding to the mini-batch will be extremely sparse, implying that graph regularization will fail to take place on most computations. A naive way to address this would be to loop over all the neighbors for each point in the mini-batch, but this would prevent us from doing efficient matrix-matrix multiplications and would severely degrade performance negating any benefits of using fast processors like GPUs.

Thus, for the graph regularizer to be effective in a computation-efficient way, our mini-batches need to reflect the structure of the graph. To do this, we partition our affinity graph into \( k \) balanced parts by minimizing edge-cut (i.e., given a partition \( k \)-partitioned affinity matrix has a dense block-diagonal structure; during mini-batch gradient descent, each mini-batch tends to correspond to points inside one of the partition blocks. The corresponding relatively dense sub-matrices of the affinity matrix are used for the graph regularization computation over the mini-batches.

4.1. SGD on Graph-Based Mini-Batches

Theoretically, SGD, gives us an unbiased estimate of the true gradient, but only if the data is sampled from the true distribution. If our entire data set approximates this true distribution reasonably well, then a randomly sampled mini-batch will also be faithful to this distribution. However, for a graph partitioned mini-batch this argument no longer holds since the data points that comprise a mini-batch are not randomly sampled, but on the contrary, reflect relatively homogeneous regions on some low dimensional manifold (since we are partitioning by minimizing edge-cut on a \( k \)-NN graph). Thus our gradient estimate is no longer unbiased, leading to poor convergence of SGD. On the other hand we have also seen that randomly shuffled batches will cause the graph regularizer to become ineffective due to poor within-batch neighbor connectivity, unless one accepts extremely long computational times (and communication costs in a parallel implementation).

Constructing a mini-batch that gives good SGD convergence, and good neighbor connectivity represents a trade-off between two somewhat mutually opposing properties: diversity (for SGD convergence, also found to be the case in [20, 21] ) and good neighbor-connectivity (for efficient graph regularization), which usually implies homogeneity. The full batch (i.e., the entire data set) however, has both these properties: perfect neighbor connectivity (since it contains all the points) as well as diversity\(^2\) that mimics the diversity within the complete training data (assuming a large enough, well sampled training set). Indeed, if we are allowed to increase the size of the mini-batches as we please, we could presumably capture a more diverse set of points as well as a significant fraction of their neighbors, but computational and memory constraints prevent us from doing so. Note that

\(^2\)We use Shannon entropy, calculated on the labels in a mini-batch, as a measure of diversity, but we anticipate better diversity measures exist.
the global structure of the affinity graph, owing to its sparsity, consists of a large number of small tightly connected clusters, with relatively few edges between the clusters. Thus a mini-batch that somehow captures this structure, but on a smaller scale, will be expected to have reasonably good connectivity as well as high entropy. This suggests a possible heuristic for the construction of improved mini-batches:

1. Given \( N \) data points, a batch size \( B \) (that represents our memory constraint) and \( M \) classes, partition the entire graph into \( \frac{NM}{B} \) mini-blocks, where each mini-block is approximately balanced at size \( B/M \).

2. Construct \( N/B \) “meta” batches of size \( B \) from the mini-blocks as follows:
   
   (a) For each batch \( b_i \), randomly choose \( M \) mini-blocks from the set of \( \frac{NM}{B} \) mini-partitions that were created in Step 1.

   (b) Group these \( M \) mini-blocks into one larger meta-batch. Since each mini-block is approximately size \( B/M \), our meta-batch will be approximately of size \( B \), satisfying our memory constraint.

At the end of this process we have meta-batches which are of the same size \( B \) as the earlier graph-based batches, but which are qualitatively different. Each meta-batch is now composed of many small homogeneous mini-blocks which, due to random sampling, are likely to be of a different class. We omit the proof here due to space constraints, but intuitively we expect that the resulting entropy from grouping together \( M \) such randomly chosen mini-blocks (of approximately equal size) to approach the entropy of the training set.

To see the effect of this process on the within-batch neighbor connectivity of the meta-batch, let \( N_i \) represent the set of neighbors of node \( i \) and \( C_i \subseteq N_i \) be the set of neighbors of a node \( i \) that are within the same batch. Let \( M_j \) be the set that represents mini-batch \( j \). We define the within-batch connectivity of \( M_j \) as

\[
e_j = \frac{\sum_{i \in M_j} |C_i|}{\sum_{i \in M_j} |N_i|}, \quad \forall i \in M_j, j = 1, 2, 3 \ldots k
\]

Let \( C_{\text{min}} \) and \( C_{\text{meta}} \) denote the random variables that represent the within-batch connectivity of a mini-block and meta-batch respectively. One can show that grouping \( K \) mini-blocks to form a meta-batch does not adversely impact the connectivity score, i.e., \( E[C_{\text{meta}}] \geq E[C_{\text{min}}] \). Further, using the Central Limit Theorem, we can show that the variance of \( C_{\text{meta}} \) is given by \( \sigma_{C_{\text{meta}}}^2 = \frac{1}{K} \sigma_{C_{\text{min}}}^2 \).

### 4.2. Stochastic Regularization over Graphs

Even though a meta-batch constructed using the procedure described in the previous section has much better neighbor-connectivity than a randomly shuffled batch, for a given node, a significant number of neighbors still lie outside the meta-batch. As we argued earlier, regularizing against all neighbors is computationally inefficient. To preserve efficiency while still regularizing against out-of-batch neighbors, at each step, we randomly pick one additional meta-batch and regularize against this neighbor as follows: consider the graph induced by the meta-batches, \( G_M = (V_M, E_M) \), where each \( M \) is a meta-batch, and edge \( e_{i,j} \in E_M \) exists between \( M_i \) and \( M_j \) if there exist some edge \( v_{i,t} \) between nodes \( v_i \) and \( v_t \) in the affinity graph \( G \), such that \( v_t \in M_i \). Thus, meta-batches are connected if their member nodes are connected in the original affinity graph. Let \( C_{i,j} \) denote the set consisting of all such unique pairs \( v_i, v_t \). Then we can define an edge-weight on each of the edges in \( E_M \) as \( |C_{i,j}| \).

For a given meta-batch \( M_j \), during each epoch, the probability of picking a neighboring meta-batch \( M_j \) is given by

\[
p_{i,j} = \frac{|C_{i,j}|}{\sum_j |C_{i,j}|}.
\]

Thus, a neighboring batch \( M_j \) of batch \( M_i \) is more likely to be picked during an epoch if there are a relatively large number of edges between the member nodes that comprise \( M_i \) and \( M_j \). Over a large number of epochs, graph regularization is likely to take place against all neighboring batches; this enables labels to propagate via a stochastic diffusion process within the connected components of the affinity graph.

### 5. EXPERIMENTS

For all our experiments in this work we use the TIMIT speech corpus [12] and just report the frame-level phone classification accuracy. Features consist of 39-d vectors consisting of MFCC coefficients, and first and second derivatives. All data is normalized for zero mean and unit variance. We apply a sliding window of radius 4, resulting in a 351 dimensional feature vector. The output is a distribution over 49 classes, which is collapsed to 39 classes during scoring. We use the 362 speaker set for training and experiment with label ratios of 2%, 5%, 10%, 30%, 50% and 100% by randomly dropping labels from our training set. Hyper-parameters were tuned using parallel grid search on a validation set. We implemented all our models using the Theano toolkit [22]. For the results reported here we used the AdaGrad [23] variant of gradient descent and use a hold-out set for early stopping. For the k-NN graph construction, we set \( k = 10 \) for all the experiments and use the Scikit machine learning library [24] that constructs the graphs using a fast ball-tree search. After symmetrization, affinities are computed by applying a radial basis function (RBF) kernel, such that each entry \( w_{ij} \) in the affinity matrix \( W \), \( w_{ij} = e^{-\frac{d_{ij}^2}{2\sigma^2}} \), \( \sigma \) controls the width of the kernel and determines how quickly the influence of a neighbor node decays with distance. As in [3], we tune \( \sigma \) over the set \( \{d_i/3 \} \) where \( i \in \{1, 2, 3, 4, 5 \} \) and \( d_i \) is the average distance between a node and its \( i \)-th nearest neighbor. For graph partitioning, we use the METIS graph partitioning library [25] that uses a recursive multi-way partitioning to give approximately balanced blocks.

We initially tested the benefit of the meta-batches and stochastic graph regularization on a shallow neural network – a multi-layer perceptron (MLP) having one hidden layer of 2000 units and a softmax output layer. These results are shown in Figure 1. A graph-regularized MLP that uses mini-batches based on purely graph partitions, and without additional out-of-batch neighbor regularization performs the worst (red curve in Figure 1); this is not surprising considering the biased gradients when using relatively homogeneous graph-based mini-batches. Using meta-batches, both with and without stochastic out-of-batch regularization (the blue and green curves respectively), noticeably improves performance, the former beating the base MLP (a supervised learner) at all scenarios except the fully-labeled case. Next, we conducted experiments on a DNN with four hidden layers, each 2000 units wide, using Rectified Linear Units [26] as the non-linear activation function, and a softmax output layer. We also exploited the feature-learning ability of DNNs – which have been shown to be good at highly non-linear dimensionality reductions [27], and improve graph clustering [28] – to produce graph embeddings.
SSL framework reported in [4]. Compared to the latter work, for the shallow classifier (GraphMLP), we generally get better phone accuracy rates although at higher label ratios [4] is better. This is probably due to regularizing against a prior distribution output from a first-pass classifier, which provides better priors at higher label ratios. When moving to DNNs, however, we are able to improve performance over [4]: the graph-enabled DNN gives the best performance at all labeled ratios both due to the depth of the classifier as due to the added benefit of using a better embedding obtained from the deep auto-encoder. Any embedding can be used with the semi-supervised DNN in general, and one of the future directions of research is to compare the effect of different graph-based representations.

6. CONCLUDING REMARKS

We presented a method for graph-based semi-supervised learning with DNNs for classifying acoustic data. The training method samples data that enables regularization over the graph (which is a proxy for the manifold) but also preserves diversity in the training samples for convergence of SGD to good solutions. The stochastic graph regularization technique allows efficient out-of-batch regularization and though we only report frame-level classification accuracy on the TIMIT speech data set, this is a general, scalable technique that can be applied to much larger and different kinds of data sets, other types of parametric classifiers and can also be extended to online and distributed learning settings.

7. ACKNOWLEDGEMENTS

Thulasidasan was supported by the LDRD Program of the Los Alamos National Laboratory under U.S. Department of Energy Contract No. DE-AC52-06NA25396. Bilmes was supported by the National Science Foundation under Grant No. IIS-1162606, and by a Google, a Microsoft, a Facebook, and an Intel research award, and also in part by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

8. REFERENCES

Language Technologies Institute, School of Computer Science, 2005.


