Semi-Supervised Learning for Improved Expression of Uncertainty in Discriminative Classifiers

Jonathan Malkin and Jeff Bilmes

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

Abstract

Seeking classifier models that are not overconfident and that better represent the inherent uncertainty over a set of choices, we extend an objective for semi-supervised learning for neural networks to two models from the ratio semi-definite classifier (RSC) family. We show that the RSC family of classifiers produces smoother transitions between classes on a vowel classification task, and that the semi-supervised framework provides further benefits for smooth transitions. Finally, our testing methodology presents a novel way to evaluate the smoothness of classifier transitions (interpolating between vowels) by using samples from classes unseen during training time.

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

1. Introduction

Many classifiers can discriminate between classes quite well, even in multi-class scenarios. For some situations, though, many models discriminate a little too well, placing extremely high confidence on one class even in the presence of quite plausible alternatives, or when the input vector is in regions with very little training data concentration. Coarticulated speech is a good example of this, where the vocal tract articulary may not reach the theoretical target for one phone before starting the transition to the next sound. In such cases, a classifier that can smoothly transition between phone class probabilities, rather than having a very abrupt (i.e., suddenly changing) transition on either side of some unseen decision boundary, would be quite useful at avoiding overly confident but potentially wrong predictions. When classifiers are overly confident in their predictions, yielding probability distributions having a winner-take-all quality, they fail to accurately represent the true uncertainty in a decision making problem. Another example, and the one most directly motivating this work, is the Vocal Joystick [6], an assistive device for individuals with motor impairments that allows continuous control of the mouse pointer using non-verbal vocalizations. For the Vocal Joystick, our goal is to learn to smoothly change between different vowel sounds which used vocalizations. For the Vocal Joystick, our goal is to learn to smoothly change between different vowel sounds which used vocalizations. For the Vocal Joystick, our goal is to learn to smoothly change between different vowel sounds which used vocalizations.

In previous work [1], we presented a semi-supervised learning (SSL) objective and applied it to neural networks. That objective, however, is quite general and can be applied to any parametric differentiable classifier to create a semi-supervised version of that model. This paper extends that objective to the Ratio Semi-definite Classifier (RSC) [2] and its multi-layer extension (ML-RSC) [3]. The RSC family of classifiers was developed to address the overconfidence problem found in many classifiers based on exponential functions. In addition to showing simple classification accuracy results, we provide an improved analysis that demonstrates that the SSL objective leads to smoother transitions between classes on a vowel classification task.

2. A SSL Training Framework

During training for supervised classification, every sample point has both a feature vector and a class label. For (SSL) tasks, we have a mix of labeled and unlabeled points. Let \( D^l = \{ (x_i, y_i) \}_{i=1}^l \) represent labeled points and \( D^u = \{ x_i \}_{i=l+1}^n \) unlabeled points, where \( n = l + \ell \) giving \( n \) points in total. We denote the output vector of posterior probabilities from a classifier with parameters \( \theta \) given point \( x_i \) as \( p_\theta(x_i) \). And the \( k \)th entry of \( p_\theta(x_i) \) is denoted \( p_\theta(k|x_i) \). Finally, let \( t_i \), \( 1 \leq i \leq l \) denote a probabilistic label vector for training sample \( i \). If the input \( x_i \) has a single output \( y_i \), \( t_i \) is a vector with zeros except for a one in position \( y_i \), known as a hard label. Note that in this paper, each \( x_i \) corresponds to a fixed-sized window of speech data, and \( p_\theta(k|x_i) \) is a parametric classifier based on this fixed window.

The SSL approach in this paper assumes that the data lie on a manifold embedded in features space. We construct a weighted, undirected graph \( G = (V, E) \) where each vertex \( V = \{ 1, \ldots, n \} \) corresponds to one data point \( x \) and \( E = V \times V \) are the set of edges between vertices. The weight of the edge between vertices \( i \) and \( j \) is given as \( \omega_{ij} \in W \).

All the models used in this work share the same objective function, enabling semi-supervised learning by adding additional terms to the standard cross entropy objective for discriminative training. As a result, setting weights on the new terms to zero reduces to supervised training. Our objective function \( J(\theta) \) to be minimized is:

\[
J(\theta) = \sum_{i=1}^l D(t_i \parallel p_\theta(x_i)) + \gamma \sum_{i,j=1}^n \omega_{ij} D(p_\theta(x_i) \parallel p_\theta(x_j)) + \alpha \sum_{i=1}^l D(p_\theta(x_i) \parallel u) + \lambda \| \theta \|.
\]

(1)

Here, \( D(p \parallel q) \) is the Kullback-Leibler (KL) divergence between probability distributions \( p \) and \( q \). \( u \) is the uniform distribution, and \( \| \theta \| \) is a parameter regularizer. The objective also has several hyperparameters to tune: \( \gamma, \alpha, \lambda \geq 0 \). In Equation 1, the first term favors distributions close to the target distribution. Setting \( \gamma = 0 \) and using hard labels yields standard conditional maximum likelihood training. The second term is a
graph regularizer, which favors smooth solutions over the graph — it prefers posterior distributions that change slowly between similar points in feature space. The third term is an entropy regularizer, preferring posterior distributions with higher entropy.

The last term is a standard parameter regularizer (e.g. $\ell_2$). More details are given in [1].

3. Models and their Optimization

Of the three models with which we have used our objective function, the multi-layer perceptron or neural network is by far the most well-known. For the MLP, we have model parameters $\theta_{\text{MLP}} = \{W^{\text{ih}}, W^{\text{ho}}\}$, a pair of matrices corresponding respectively to the hidden-to-output weights and the input-to-hidden weights of the MLP. Note that we use the symbol $w$ to refer to MLP weights, contrasted with $\omega$ to refer to graph edge weights.

The ratio semi-definite classifier defines a posterior distribution as a ratio of semi-definite polynomials:

$$p_{\text{RSC}}(y|x) = \frac{(x - d_i)^T B_k B_k^T (x - d_i)}{\sum_k (x - d_i)^T B_k B_k^T (x - d_i)}$$

(2)

where $B_k \geq 0$, for all $k$ and our model parameters in this case are $\theta_{\text{RSC}} = \{(B_k, d_k)^T\}$ for $K$ classes. The matrix substitution ensures the RSC produces a valid probability distribution and allows an unconstrained optimization problem. The model avoids any fast growing functional forms such as exponentials, leading to some interesting properties. Also note that, despite the similarities to the argument of the exponential in a Gaussian distribution, the $d$ vectors are akin to anti-means in the RSC. More details of the model may be found in [2].

The RSC tends to have a high entropy bias with more than a few classes, and cannot model very complex decision boundaries. The third model used is the multi-layer RSC and was designed to help alleviate these issues in multi-class classification. The ML-RSC model extends the original RSC in the same way that an MLP extends the original perceptron, by adding a non-linear hidden layer. Thus, we replace the inputs $x$ to the RSC with $z = \sigma(x)$ where $\sigma(\cdot)$ is a non-linear function such as a sigmoid and the model parameters are $\theta_{\text{MLRSC}} = \{W, (B_k, d_k)^T\}$. By performing this substitution, the ML-RSC learns a kernel-like transformation in a data-driven matter, achieving much greater modeling power in the process. Additional information on this model appear in [3].

When differentiating Equation 1 with respect to the model parameters, none of the models used have a closed form optimal solution. In all cases, then, we learn models via stochastic gradient descent.

The KL divergence can be decomposed into entropy and cross entropy terms as $D(a \parallel b) = H'(a,b) - H(a)$, where $H'(a,b) = -\sum a_i \ln b_i$ is the cross entropy of distributions $a$ and $b$. Using this, the objective becomes:

$$J = D(q_i \parallel p_i) + \gamma \sum_{j=1}^n \omega_{ij} H'(p_i, p_j)$$

$$- (\kappa + \gamma \sum_{j=1}^n \omega_{ij}) H(p_i) + \kappa \log K + \lambda \|\theta\|.$$  

(3)

With hard labels, we recover fully supervised conditional maximum likelihood training by setting $\gamma = \kappa = 0$. Because none of these models has a closed form solution, if instead $\gamma > 0$, the cross entropy-based term in Equation 3 contributes two terms for which we need to calculate derivatives. For the MLP and ML-RSC, we can essentially extend back propagation, meaning we propagate results from the two terms independently in this case. Derivatives for the MLP can be found in [1]. For the RSC and ML-RSC, see the appendices in [4].

4. Toy Data — 2 moons

As a visual demonstration of the utility of this model, we tested the model on a 2-D problem common in the machine learning literature known as “two-moons.” This is actually a transductive learning problem, where our test data are the unlabeled training data points. As shown in Figure 1, we created a semi-supervised learning task by randomly sampling labels from the fully-labeled data. In this case, we used a graph constructed using Euclidean distance and 10 nearest neighbors.

![Figure 1: Color plots showing (a) fully labeled two moon data, (b) data after sampling 5 labels from each class. Bottom row shows decision regions and confidence (brighter is more confident) with overlaid data for 3 hidden layer models for the (c) SSL-MLP and (d) SSL-MLRSC.](image)

While none of the supervised training models we tried, namely a MLP, ML-RSC, and support vector machine (SVM), could successfully classify all points on this toy problem, and the original RSC could not model the boundary even in the fully-supervised case, both the SSL-MLP and SSL-MLRSC were able to correctly label all points — using only 3 hidden units — as shown in Figures 1(c) and 1(d). Increasing the number of hidden units made little difference for the SSL-MLP, while the SSL-MLRSC showed a greater contrast in confidence between areas with points versus areas without as the number of hidden units increased.

5. Experimental Environment

Testing the smoothness of classifier transitions is not entirely straightforward. Of course, however smooth a model transitions between classes, a classifier is not very useful if it cannot classify accurately.

---

\footnote{For notational simplicity, we will slightly change to the notation $p_i = p_i(x_i)$ where the dependence on the parameters $\theta$ and on the $i^{th}$ input sample is implicit.}
classes during training. Yet we do not want to significantly hurt sounds correctly, despite having never seen instances of those
between vowel classes will be more likely to classify the ordinal would expect that models which more smoothly transition be-
of the phonetically-motivated relationship between vowels, we that as a measure of accuracy (see details below). Because
the vowels associated with ordinal (diagonal) directions. From
experiment with both the original set of four vowels as well as
We presented the best scoring model of each type from the first
eels could learn to interpolate between different speech classes.
unlabeled, the diphthongs used were from the test set speakers.
tively little diphthong data from the training set speakers, and because they were
thongs as unlabeled data. Note that, because there was rela-
from monophthong utterances while the other included diph-
the SSL models, we tested two versions: one used only frames
x
accuracy is not always consistent. For the MLP, there is an in-
Table 1: Smoothness analysis on the 4 vowel VJ corpus de-
GM-linear-RSC+diph 98.8 97.8 18.6
GM-MLP+diph 99.0 98.0 8.3
Baseline-MLP 99.1 98.3 8.5
SSL-MLRSC 99.1 98.1 8.4
SSL-MLRSC+diph 98.8 97.3 9.4
Baseline-RSC 98.1 91.9 15.7
SSL-RSC 98.1 79.9 24.9
SSL-RSC+diph 96.1 87.7 18.6

Table 1: Smoothness analysis on the 4 vowel VJ corpus de-
ability set using all labels (and optionally unlabeled diph-
thons). Results are expressed as accuracies, with the last two
points referring to accuracy based on the direction of move-
ment from the posterior probability vectors. Bold indicates sig-
nificance (p < 0.0001).

Accuracy in the cardinal directions.

In the Vocal Joystick, the movement direction is determined from probabilities \(p_i\) for each vowel \(i\). At a high level, each probability is considered a scaled vector in the direction of vowel \(i\), and projected onto each axis \(j\) \(\in\{x,y\}\). The vec-
tors for each axis are then added together to produce a direction \(\mathbf{d}_j = \sum p_i \cdot (v_i, u_j)\), where \(v_i\) is a unit vector pointing to-
towards vowel \(i\) and \(u_j\) is a unit vector along axis \(j\). With only 4
vowels, the direction vectors are simple:

\[
\begin{align*}
d_y &= p(a|x) - p(a|x) \\
d_x &= p(o|x) - p(i|x)
\end{align*}
\]

where each posterior distribution is conditioned on the associated input vector \(x\). The movement angle is then \(\hat{\phi} = \arctan\left(\frac{d_y}{d_x}\right)\), adjusted to use the entire \(2\pi\) output range as appropriate. The angle classification decision is argmin \(\phi\) \in {\$\hat{\phi}\$} where \(i\) is one of the 8 vowels and \(\phi_i\) is the direction associated with vowel \(i\) from Figure 2(a). See Figure 2(b) for a
graphical representation

Features were 26-d MFCCs (including single deltas) cal-
tuated over 25ms windows with a 10ms step between frames. Graphs were constructed over the training data using \(k\)-nearest neighbors (\(k\)-NN) based on Euclidean distance. In all cases, we used \(k = 20\). We applied a radial-basis function kernel with

\[
\omega_{ij} = e^{-\frac{\|x_i - x_j\|^2}{\sigma}}
\]

The value of \(\sigma\) was tuned over the set \(d_i/3, i \in \{1, 2, 3, 4\}\) where \(d_i\) is the average distance between each node and its \(i^{th}\) nearest neighbor. The other hyperparameters were also tuned via a grid search.

6. Results
As we present results, note that we tuned models using probability-based accuracy on the development set. Since we did not tune for direction-based accuracy, such compar-
isons are meaningful even on the development set. Comparing probability-based accuracies, we generally found that the SSL-MLP performed the best, although not significantly better than some flavor of the ML-RSC. This trend held quite consistently.

From Table 1, the impact of the graph on direction-based accuracy is not always consistent. For the MLP, there is an increase in cardinal direction accuracy, but a decrease in ordinal direction accuracy — except when the diphthongs are included
Table 2: Smoothness analysis on the 4 vowel VJ corpus test set using all labels (and optionally unlabeled diphthongs). Results are expressed as accuracies, with the last two columns referring to accuracy based on the direction of movement from the posterior probability vectors. Bold indicates significance ($p < 0.0001$).

<table>
<thead>
<tr>
<th>Test Set</th>
<th>Prob.- based</th>
<th>Direction-based</th>
<th>Cardinal</th>
<th>Ordinal</th>
</tr>
</thead>
<tbody>
<tr>
<td>All labels</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Baseline MLP</td>
<td>92.7</td>
<td>90.5</td>
<td>6.8</td>
<td></td>
</tr>
<tr>
<td>SSL-MLP</td>
<td><strong>94.2</strong></td>
<td><strong>92.3</strong></td>
<td>7.2</td>
<td></td>
</tr>
<tr>
<td>SSL-MLP+diph</td>
<td><strong>94.1</strong></td>
<td>91.8</td>
<td>8.7</td>
<td></td>
</tr>
<tr>
<td>Baseline ML-RSC</td>
<td><strong>94.1</strong></td>
<td>91.5</td>
<td>7.7</td>
<td></td>
</tr>
<tr>
<td>SSL-MLRSC</td>
<td>93.6</td>
<td>91.6</td>
<td>8.8</td>
<td></td>
</tr>
<tr>
<td>SSL-MLRSC+diph</td>
<td>93.2</td>
<td>90.8</td>
<td>10.2</td>
<td></td>
</tr>
<tr>
<td>Baseline RSC</td>
<td>91.1</td>
<td>82.6</td>
<td>17.8</td>
<td></td>
</tr>
<tr>
<td>SSL-RSC</td>
<td>92.1</td>
<td>77.0</td>
<td><strong>21.7</strong></td>
<td></td>
</tr>
<tr>
<td>SSL-RSC+diph</td>
<td>89.5</td>
<td>80.9</td>
<td>18.9</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Smoothness analysis on the 4 vowel VJ corpus development set using 1% of labels. Results are expressed as accuracies, with the last two columns referring to accuracy based on the direction of movement from the posterior probability vectors. Bold indicates significance ($p < 0.0001$).

<table>
<thead>
<tr>
<th>Dev. Set</th>
<th>Prob.- based</th>
<th>Direction-based</th>
<th>Cardinal</th>
<th>Ordinal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1% labels</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Baseline MLP</td>
<td>97.7</td>
<td><strong>96.4</strong></td>
<td>7.2</td>
<td></td>
</tr>
<tr>
<td>SSL-MLP</td>
<td><strong>98.8</strong></td>
<td>91.1</td>
<td>17.0</td>
<td></td>
</tr>
<tr>
<td>Baseline ML-RSC</td>
<td>98.3</td>
<td>93.8</td>
<td>20.5</td>
<td></td>
</tr>
<tr>
<td>SSL-MLRSC</td>
<td><strong>98.7</strong></td>
<td>95.3</td>
<td>11.9</td>
<td></td>
</tr>
<tr>
<td>Baseline RSC</td>
<td>39.9</td>
<td>21.7</td>
<td>17.9</td>
<td></td>
</tr>
<tr>
<td>SSL-RSC</td>
<td>97.8</td>
<td>82.9</td>
<td><strong>27.6</strong></td>
<td></td>
</tr>
</tbody>
</table>

During training, in which case the cardinal direction accuracy increases. (Note that the diphthongs are from test set speakers, but these results are on the development set). For the ML-RSC, there is little change between the baseline ML-RSC and its semi-supervised version except when we include the diphthongs during training. In that case, classification and cardinal direction accuracy is slightly decreased, but we find an increase in ordinal direction accuracy. And for the RSC, there is a very large decrease in cardinal direction accuracy combined with a very large increase in ordinal direction accuracy with the graph (without diphthongs). Adding the diphthongs decreased the RSC’s classification accuracy quite a bit, but improved cardinal direction accuracy at the cost of decreased accuracy for the diagonals. The diagonals are, however, still improved when using the diphthongs when compared to the baseline RSC.

Looking at the test set, in Table 2, we largely find the same trends. In no case does the addition of the diphthongs provide an increase in classification accuracy, although they do not always hurt. The other differences worth noting are that, for both the MLP and ML-RSC, the graph without diphthongs provides an increase in ordinal direction accuracy on the test data.

Overall, we can see that, by considering not just classification accuracy but also the direction of movement, the addition of the graph regularizer and entropy term help provide smoother direction-based movement. This effect was not readily apparent from the classification accuracy results alone.

Moving to experiments using only 1% of the labeled data, the development set results appear in Table 3. We know that the addition of the graph never hurts classification accuracy in this case, but the effect on cardinal direction accuracy is less clear. For the SSL-RSC over the baseline RSC is not a surprise since the accuracy of the baseline is so low. But the difference between the MLP and ML-RSC when using so little labeled data is interesting. The baseline MLP seems to learn more confidently, in that the cardinal direction accuracy is much closer to the classification accuracy. In doing so, however, it allows for much less ambiguity between classes. The SSL-MLRSC, on the other hand, seems to take advantage of the graph to become more confident in some of its predictions to a greater extent than does the MLP.

7. Conclusions and Discussion

In this work, all our models were discriminatively trained, and the main analysis stemmed from testing on unseen classes. This may hide the fact that training with our semi-supervised objective consistently provided improvements in classification accuracy on this task. Although none of the models did particularly well on the ordinal direction frames, the use of our objective function still yielded improvements, often substantially — while tuning each model for a related but quite distinct task. The obvious next step, of course, is to select model hyperparameters via direction-based accuracy, revealing just how high the accuracies can reach as well as how that criterion will impact standard classification accuracy. Finally, we would like to thank Richard Wright for his help in developing the direction-based evaluation criterion.

8. References