MULTI-LAYER RATIO SEMI-DEFINITE CLASSIFIERS

Jonathan Malkin and Jeff Bilmes

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

ABSTRACT

We develop a novel extension to the Ratio Semi-definite Classifier, a discriminative model formulated as a ratio of semi-definite polynomials. By adding a hidden layer to the model, we can efficiently train the model, while achieving higher accuracy than the original version. Results on artificial 2-D data as well as two separate phone classification corpora show that our multi-layer model still avoids the overconfidence bias found in models based on ratios of exponentials, while remaining competitive with state-of-the-art techniques such as multi-layer perceptrons.

Index Terms— Pattern recognition, Speech recognition

1. INTRODUCTION

Exponential-based probability models for classification, such as Gaussian mixture models or multi-layer perceptrons (MLPs) [2], whether right or wrong, are often quite confident in their decisions even in regions of low training data concentration. For MLPs, this is the case even though theoretically, given sufficient training data and the "right" MLP, the model will converge to the true posterior distribution $p(y|x)$ — this is because in practice the model is almost never right and the training data is almost never sufficient. Moreover, additional scaling factors, such as $p(y|x)^\alpha$ with $0 < \alpha < 1$, followed by renormalization does not change the underlying exponential form of such models and can be seen as only a palliative.

Low entropy posteriors may be acceptable in some situations. But there are a number of instances in which they may be less desirable. Take, for instance, ranking classifiers, where we want not just a probability of a correct class but a numerically ordered list. Meaningful rankings become difficult when only vanishing probability exists to divide amongst non-top-ranked categories.

Another situation to consider is speech recognition (ASR). When decomposing an ASR system, the acoustic model typically uses an exponential-based probability model while the language model often uses an n-gram, which is much less likely to have a low entropy bias. Often one needs to find a parameter to trade-off between the two scores to compensate for the different dynamic ranges. By using an acoustic model lacking an overconfidence bias, we may be able to include alternative options in a beam-constrained search while eliminating or reducing the need for a trade-off parameter.

Finally, there is the application motivating our work, the Vocal Joystick (VJ) [1]. Designed as an assistive device for individuals with motor impairments, the goal is to allow voice-based *continuous* control of mouse movement, video games, drawing [5], or even a small robotic arm [6]. The VJ currently uses vowel quality to control 2-D movement direction while loudness determines speed. For mouse pointer control, vowel quality alone determines the movement direction. The system must run in real-time, as vocal tract micro-adjustments must immediately be reflected on-screen.

The VJ has had much success using an MLP-based vowel classifier where posterior probabilities act as mixing weights to estimate vowel quality [13]. This classifier, however, tends to be overconfident (low entropy) especially when input vowels are located between standard vowel categories — this tends to produce motion only in one of the cardinal or ordinal directions thus making it difficult to produce smooth curves. Our ultimate goal is to find models that can smoothly transition between classes, without a sudden jump at classification boundaries. Specifically, we want models that avoid highly confident decisions in areas with sparse data or conflicting labels. High confidence decision should be restricted only to areas with densely packed data from a single class, and even in those high confidence regions, we prefer posteriors with entropy high enough to allow meaningful rankings. We want, moreover, to do this using a method that does not sacrifice accuracy, computational tractability, and real-time response on modern microprocessors.

Last year, we presented the Ratio Semi-definite Classifier (RSC) [11], a discriminative multi-class classifier based on a ratio of semi-definite polynomials that does not rely on a fast-growing functional form such as an exponential. Derived from the model presented in [3], one interesting property of the RSC is that, unlike classifiers based on ratios of exponentials, it avoids a bias towards low entropy posterior distributions. This model provides a compelling alternative, but motion produced by an RSC is often *too* soft and accuracy suffers as well, relative to an MLP. In past work, we also evaluated a modified adaptive Kalman filter to provide movement in arbitrary directions [10], but the time lag inherent to the approach makes it hard to use for real-time control. We have also studied a Gaussian process approach [12], but so far that model's computational demands have exceeded our real-time requirements.

Drawing inspiration from the addition of a hidden layer to the original perceptron to create an MLP, we propose in this work to augment the original RSC with a hidden layer, forming what we call a multi-layer RSC (ML-RSC). This model is continuous and differentiable, and we show that its training is simple and easy via stochastic gradient descent. We compare ML-RSC, MLP, and RSC on both 2-D artificial data (for analysis purposes) and real speech data (both the vocal-joystick vowel corpus and also TIMIT). We find that, as expected, the accuracy of the ML-RSC significantly improves relative to an RSC but without sacrificing the beneficial high-entropy properties as does the MLP.

This material is based on work supported by the National Science Foundation under grant IIS-0326382.

2. PROPOSED MODEL

As presented in [11], the basic RSC is:

$$p(y|x) = \frac{(x - d_y)^T A_y (x - d_y)}{\sum_k (x - d_k)^T A_k (x - d_k)} \quad (1)$$

where x are the input features and the parameters to be learned are $\Theta = \{A_k, d_k\}_{k=1}^K$ where K is the number of classes. In order to be a valid probability distribution, we must have $A_k \succeq 0 \ \forall k$. Parameterizing each matrix A_k as $A_k = B_k B_k^T$ can guarantee semi-definiteness at the loss of convexity, as seen in [14]. The d_k vectors are interesting in that $p(k|x = d_k) = 0$ — as such, they are more like *anti-means* than the typical mean vectors we find in models such as Gaussians (or mixtures thereof). To avoid confusion, we refer to these d_k vectors as shift vectors.

Given training data $\mathcal{D} = \{(x_i, t_i)\}_{i=1}^N$ where x_i are feature vectors and t_i are target label distributions, we can put our problem in a conditional maximum likelihood framework. Learning RSCs with hard (integral) targets is solved via an optimization problem:

$$\min_\Theta - \sum_i \log \frac{(x_i - d_{y_i})^T B_{y_i} B_{y_i}^T (x_i - d_{y_i})}{\sum_k (x_i - d_k)^T B_k B_k^T (x_i - d_k)}. \quad (2)$$

Here, the reason for parametrization of A becomes clear: the original version of this problem is an instance of semi-definite programming [15] which, while polynomial in complexity, can be in practice quite computationally expensive for problem sizes at our scale. Although we have sacrificed convexity, optimization methods such as stochastic gradient descent have proven quite successful for optimizing non-convex models such as MLPs [2]. We have shown this same approach works well for RSCs [11].

2.1. Multi-layer RSC

Although the RSC quite successfully avoids a low-entropy bias while producing good results, there are some data sets on which it achieves relatively poor accuracy, as shown in Section 3. Additionally, we have observed that as the number of classes grows, its entropy starts to show a high-entropy bias quite opposite to the problem we find with the MLP.

One solution is to extend the RSC in a way similar to how the MLP extends the perceptron: we can add a hidden layer to the RSC allowing us to learn a non-linear mapping from our input features to hidden units. We can then treat the hidden unit outputs as input features to an RSC. In the event that a small hidden layer with dimensionality smaller than the original input features can be used, this approach also allows us to speed both training and evaluation over the original RSC since the ratio-semidefinite portion of the computation dominates. We note that although the resulting non-linear projection is non-convex, neither is our originally proposed RSC, yet the ML-RSC still yields an efficient effective training procedure and has good performance (see below).

Define $z = \sigma(Wx + b)$ where $\sigma(x) = \frac{1}{1+e^{-x}}$ is a sigmoid. The posterior distribution of a ML-RSC is then:

$$p(y|x) = \frac{(z - d_y)^T B_y B_y^T (z - d_y)}{\sum_k (z - d_k)^T B_k B_k^T (z - d_k)}. \quad (3)$$

The training objective function of Equation 2 also changes in a straightforward manner.

Given our objective, the ML-RSC updates are quite simple. Due to the use of stochastic gradient descent, we need the gradient for only one point at a time. We therefore drop the summation over all points i for simplicity, and consider the cross entropy objective: $E' = -\sum_k t_k \ln p(y_k|x)$.

Define $\alpha_k(z) = (z - d_k)^T B_k B_k^T (z - d_k)$ and $\beta(z) = \sum_k \alpha_k(z)$ so that $\log p(y|x) = \log \frac{\alpha_y(\sigma(x))}{\beta(\sigma(x))}$. Differentiating with respect to B_k and d_k, respectively, yields

$$\frac{\partial E'}{\partial B_k} = 2 \sum_j t_j \left(\frac{\alpha_j - \beta \cdot \delta(k = j)}{\alpha_j \beta} \right) (z - d_k)(z - d_k)^T B_k$$

$$\frac{\partial E'}{\partial d_k} = -2 \sum_j t_j \left(\frac{\alpha_j - \beta \cdot \delta(k = j)}{\alpha_j \beta} \right) B_k B_k^T (z - d_k).$$

Both of these are identical to the gradients for the regular RSC if we treat the hidden unit outputs z as our input features. For the weight updates, we follow a process similar to standard back propagation:

$$\frac{\partial E'}{\partial W} = 2 \sum_k \left(\frac{\alpha_k - t_k \beta}{\alpha_k \beta} \right) \left(B_k B_k^T (z - d_k) \circ z \circ (\mathbf{1} - z) \right) x^T$$

where $\circ$ is the Hadamard (elementwise) product which in this case is performed after any matrix-vector multiplications, and $\mathbf{1}$ is a vector of ones. By appending an extra dimension with a constant 1 onto input vector x, this handles the input-to-hidden layer biases as well.

2.2. Regularization and Penalty

We add regularization [2, 9] terms to the training objective via the Frobenius norm of the matrices B_k, the L2 norm of the shifts d_k, and the Frobenius norm of the weight matrix W. We use regularization coefficients λ_B, λ_d and λ_W, respectively. As detailed in [11], if $d_k = d \ \forall k$, the RSC is not continuous where $x = d$ (or $z = d$ in the ML-RSC). We thus introduced a penalty $\frac{1}{|C|}$ where $C = \sum_{k=1}^K (d_k - \mu)(d_k - \mu)^T$, with $\mu = \frac{1}{K} \sum_k d_k$. We expect this penalty may be even more important here since the hidden unit outputs z will change during optimization and can possibly saturate, unlike the input features x. In practice, a very small weight λ_s on this term has proven sufficient. For full details, see [11].

With the regularization terms and penalty, the final objective is:

$$E = E' + \lambda_B \sum_k \|B_k\|_F^2 + \lambda_d \sum_k \|d_k\|_2^2 + \lambda_w \|W\|_F^2 + \lambda_s \frac{1}{|C|}.$$

The final derivatives are thus:

$$\frac{\partial E}{\partial B_k} = \frac{\partial E'}{\partial B_k} + \lambda_B B_k \quad (4)$$

$$\frac{\partial E}{\partial d_k} = \frac{\partial E'}{\partial d_k} + \lambda_d d_k - \lambda_s \frac{2}{K|C|} C^{-1} (d_k - \mu) \quad (5)$$

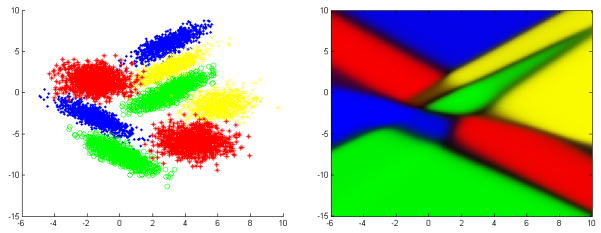
$$\frac{\partial E}{\partial W} = \frac{\partial E'}{\partial W} + \lambda_W W. \quad (6)$$

3. ANALYSIS ON ARTIFICIAL 2-D DATA

To better understand the ML-RSC compared to the RSC and MLP, we first looked at the performance of each model on artificial 2-D data. We used 4-class Gaussian mixture data with 2 components per mixture.

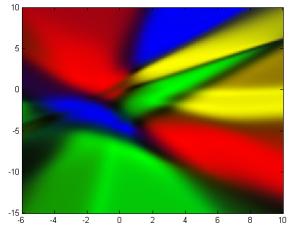
For each classifier, we did a manual parameter search and looked at decision boundaries and entropy over a region enclosing the training data. The data appears in Figure 1(a) and combined decision region and entropy plots appear in Figures 1(b), 1(c), and 1(d). The

color at each point is a linear combination of colors weighted by probabilities, and entropy at each point determines brightness. A fully saturated color represents lowest entropy, and black represents maximal entropy. The best MLP used a total of 7 hidden units, while the best ML-RSC used 14.

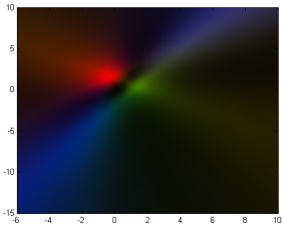


(a) 4-class training data

(b) MLP



(c) ML-RSC



(d) RSC

Fig. 1. Color plots showing (a) training data used, (b) MLP (7 hidden units), (c) ML-RSC (14 hidden units), and (d) original RSC, all trained on the same data. In all cases, brightness corresponds to confidence, and the colors are a linear combination of predicted labels weighted by the probability of each label.

As we can see, the original RSC clearly has much trouble with this data; its error on the training data is 30%. The MLP had a training set error of 1.775% and the ML-RSC's error was 1.725%, an insignificant difference (proper held-out test sets showing generalization performance are used below with the real data).

The entropy results are also interesting. The RSC is lacking in confidence everywhere, in addition to its high error rate. As expected, the MLP is much more confident than the ML-RSC in general, losing confidence only right around decision boundaries. The ML-RSC is more confident in areas with data, but its confidence starts to fall quickly in areas lacking data. The ML-RSC's entropy is lower than that of the MLP 27% of the time in this plot area, and its mean normalized entropy over the area is 0.30, versus 0.15 for the MLP. The RSC, by contrast, had a mean normalized entropy of 0.74. Based on this preliminary result, we have a very promising improvement over the original RSC. The ML-RSC retains the high accuracy of the MLP, while overcoming the low entropy bias yet retaining the ability to be confident given sufficient training data.

4. EXPERIMENTAL ENVIRONMENT

We have tested our model on two data sets. In both cases, we used MFCCs with first-order deltas giving 26-d feature vectors. Frames were 25ms long with a 10ms shift.

The first data set is the Vocal Joystick Vowel Corpus [7]. This is a set of vowels collected specifically for the VJ project. We created a training set from 21 recording sessions (2 speakers appear twice, although there is only partial overlap in their sounds), a development set of 4 speakers, and a test set of 10 speakers. All speakers come from the earlier data collection efforts described in [7] and capture the wide variability in human vowel production. We considered utterances containing only a single vowel. We tested two conditions: for the 4 vowel case, there are approximately 275k training frames (1931 utterances), and 550k frames (3867 utterances) for the 8 vowel case. For both development and testing, we determined accuracy values by splitting the data 6 ways, calculating accuracy over 5 of the 6 sets, and taking the average result. Experiments have shown that the train/dev sets and the test set on this corpus are not well-matched, creating an interesting challenge for classifiers.

VJ Dev	4 class			8 class		
	Acc.	Entropy	N	Acc.	Entropy	N
MLP	98.5%	0.12/0.21	75k	74.6%	0.77/0.55	4.8k
RSC	98.2%	0.89/0.37	133k	74.9%	2.73/0.15	266k
ML-RSC	99.2%	1.60/0.21	7.2k	75.6%	1.62/0.46	99k

VJ Test	4 class			8 class		
	Acc.	Entropy	N	Acc.	Entropy	N
MLP	93.1%	0.18/0.28	75k	68.4%	0.86/0.58	4.8k
RSC	91.5%	1.01/0.39	133k	64.8%	2.74/0.15	266k
ML-RSC	93.1%	0.26/0.30	7.2k	66.1%	1.65/0.49	99k

Table 1. Top: Dev set results for the VJ Corpus. Only the best results for each model are shown. Bottom: Test set results using the best models based on dev set results. Entropy is given as mean/dev.

Our second data set is TIMIT [4], a standard database often used for phone classification. We randomly selected 40 speakers from the training set for a 400 utterance development set. We used the standard test set and size-39 phone set described in [8].

In all cases, we grouped windows of 7 frames together, giving 182-d input vectors. This has proven to be an optimal number on the VJ corpus, and on TIMIT we used 7 instead of a more typical 9 to help reduce training time for the original RSC. The ML-RSC, by contrast, is limited more by the size of the hidden layer than the size of the input vectors.

We compared the ML-RSC to a 2-layer MLP and to an original RSC. For the Vocal Joystick corpus, we did a complete grid search over a substantial range of values to determine the best values for both the number of hidden nodes for the MLP and ML-RSC as well as regularization coefficients for all models. The exception was that for the RSC and ML-RSC we set $\lambda_s = 10^{-10}$ to be small so that it will have a significant effect only if the shifts are nearly equal. For TIMIT, all models used tied regularization coefficients to reduce search time. We noticed that the MLP often did best with regularization coefficients of 0 when tying the parameters, so we also tried setting $\lambda_W = 0$ for the ML-RSC while varying λ_B and λ_d (which remained tied).

5. RESULTS AND DISCUSSION

Development set results on the VJ Corpus appear in the top half of Table 1. We see that the original RSC's accuracy approaches that of the MLP on 4 classes, and passes it on 8. The RSC's entropy seems reasonable on 4 classes, but is quite high in the 8 class case. The ML-RSC, on the other hand, shows the best dev results under both conditions. Training MLPs with hidden layer size set to approximately equal the number of parameters of the best ML-RSCs gave results slightly worse than the best MLP results.

The bottom half of Table 1 shows test-set results. The two sets

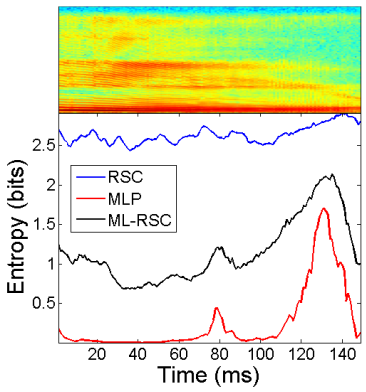


Fig. 2. Plot showing entropy over time for each classifier (8 class) on a diphthong /i-u/ with falling pitch, with spectrogram. The speaker changed vowel quality slowly until late in the utterance, hence the rise in entropy at the right edge.

TIMIT	Dev		Test		Num.
	Acc.	Entropy	Acc.	Entropy	Prm.
MLP	68.8%	1.43/0.97	68.2%	1.45/0.98	911k
RSC	49.8%	3.36/0.91	48.9%	3.38/0.91	1.3M
ML-RSC	59.9%	2.27/1.03	59.0%	2.29/1.03	333k

Table 2. Results for TIMIT, including parameter counts. Only the best dev results for each model are shown, and test results are based on that same set of parameters. Entropy is given as mean/std. Maximum possible entropy is ≈ 5.29.

are reasonably mismatched (see [7]) so accuracies have fallen. While the original RSC falls short of the baseline MLP results in both cases, the ML-RSC actually matches the MLP results for the 4 class data but not not quite on the 8 class data.

The ML-RSC shows a significant improvement over the RSC, but still with higher entropy values than the MLP, thus validating our approach on this data. We can moreover discriminatively adapt the model to each user [10] for further improved accuracy. This reinforces the positive impression from the 2-D toy data.

We also show a plot (Figure 2) of a user producing a slow diphthong, a continuously voiced transition from /i/ to /u/, with a falling pitch. With the 8 class model, the RSC entropy is always high — seemingly under-confident. The rise at the right end of the utterance exists, but is very small. The MLP, on the other hand, is quite confident everywhere except for near the decision boundary. The ML-RSC, by contrast, better captures the change in vowel quality as the speaker slowly begins to shift from /i/ to /u/ prior the more sudden change near the end.

On TIMIT, the RSC is again the least accurate (Table 2). The ML-RSC moreover shows a vast improvement over the RSC in terms of accuracy without worse entropy properties, but it is not as good accuracy-wise as the MLP. The number of parameters in the best case for the MLP (1500 hidden units) is larger than the best ML-RSC (which used 150 hidden units). An MLP with parameters comparable to the best ML-RSC gets 65.11% accuracy (100 hidden units) and 66.69% (200 hidden units). An ML-RSC with comparable parameters to the best MLP has not yet been tested. Moreover, the optimal number of hidden units for the ML-RSC along with the best regularization coefficients was not searched as thoroughly for TIMIT as it was for the VJ data (our primary application).

6. CONCLUSIONS AND FUTURE WORK

We have introduced a multi-layer extension to the ratio semi-definite classifier yielding vastly improved flexibility. This ML-RSC has been shown to produce accuracies competitive with state-of-the-art discriminative classifiers while avoiding both an overconfidence and an under-confidence bias. The ML-RSC shows a large accuracy improvement on TIMIT over the RSC, yet still lags behind the MLP accuracy at its current model size. The ML-RSC, however, appears to have the best accuracy/entropy trade-off of the three models.

In future work, we will investigate the ML-RSC as a ranking classifier, will present theoretical results regarding the ML-RSC's higher entropy properties, and introduce computationally cheaper training algorithms that will allow the ML-RSC to utilize as large a hidden-unit size as the MLP with comparable computational costs during training.

7. REFERENCES

[1] J. Bilmes et al. The Vocal Joystick: A voice-based human-computer interface for individuals with motor impairments. In *Human Language Technology Conf. and Conf. on Empirical Methods in Natural Language Processing*, 2005.

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

[3] K. Crammer and A. Globerson. Discriminative learning via semidefinite probabilistic models. In *Uncertainty in Artificial Intelligence*, Cambridge, MA, 2006.

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

[5] S. Harada, J. Wobbrock, and J. Landay. VoiceDraw: A hands-free voice-driven drawing application for people with motor impairments. In *ACM SIGACCESS Conf. on Computers and Accessibility*, Tempe, AZ, Oct. 2007.

[6] B. House, J. Malkin, and J. Bilmes. The VoiceBot: A voice controlled robot arm. In *CHI '09: Proc. ACM SIGCHI Conf. on Human Factors in Comp. Systems*, Boston, MA, Apr. 2009.

[7] K. Kilanski, J. Malkin, X. Li, R. Wright, and J. Bilmes. The Vocal Joystick data collection effort and vowel corpus. In *Interspeech*, Pittsburgh, PA, Sept. 2006.

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

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

[10] X. Li, J. Malkin, J. Bilmes, S. Harada, J. Landay, and J. Bilmes. An online adaptive filtering algorithm for the Vocal Joystick. In *Interspeech*, Pittsburgh, PA, Sept. 2006.

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

[12] J. Malkin, N. Lawrence, and J. Bilmes. The GP-LVM for Vocal Joystick control. Technical Report UWEETR-2006-0016, U. Washington Dept. of Electrical Engineering, 2006.

[13] J. Malkin, X. Li, and J. Bilmes. Energy and loudness for speed control in the Vocal Joystick. In *Automatic Speech Recognition and Understanding*, San Juan, PR, Dec. 2005.

[14] F. Sha and L. Saul. Large margin hidden Markov models or automatic speech recognition. In *Advances in Neural Information Processing Systems 19*, 2006.

[15] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.