2011 IEEE International Workshop on Machine Learning for Signal Processing
September 18-21, 2011, Beijing, China

# LEARNING DISTANCES TO IMPROVE PHONEME CLASSIFICATION

*Ryan Curtin, Nikolaos Vasiloglou, David V. Anderson*

Georgia Institute of Technology
School of Electrical and Computer Engineering
777 Atlantic Drive, Atlanta, GA, 30332-0250

## ABSTRACT

In this work we aim to learn a Mahalanobis distance to improve the performance of phoneme classification using the standard 39-dimensional MFCC features. To learn and to evaluate the performance of our distance, we use the simple $k$-nearest-neighbors ($k$-NN) classifier. Although this classifier exhibits low performance relative to state-of-the-art phoneme classifiers, it can be used to determine a distance metric that is applicable to many other better-performing machine learning methods. We devise a novel optimization method that minimizes the error function of the $k$-NN classifier with respect to the covariance matrix of the Mahalanobis distance, based on finite-difference stochastic approximation (FDSA) gradient estimates combined with a random perturbation term to avoid local minima. We apply our method to the problem of phoneme classification with the $k$-NN classifier and show that our learned distance provides performance improvement of up to $8.19\%$ over the standard $k$-NN classifier, and additionally outperforms other state-of-the-art distance learning methods by approximately $4$ percentage points. We also find that the computational complexity of our method, while not optimal, is better than other distance learning methods. The performance improvements for individual phoneme classes are given. The distances learned are applicable to other scale-variant machine learning methods, such as support vector machines, multidimensional scaling, and maximum variance unfolding, as well as others.

***Index Terms***— distance learning, Mahalanobis distance, phoneme classification, k-nearest-neighbors

## 1. INTRODUCTION

Distance learning is a popular topic in recent literature [1, 2, 3, 4, 5, 6]. The motivation for learning a distance comes from the intuitive observation that the performance of any machine learning method that uses a distance will be affected by a scaling of the dimensions of the dataset it is run on. From this observation, we can propose that there exists a particular distance that we could use to improve performance.

In this paper, we choose to restrict the problem to phoneme classification and the method to the $k$-nearest-neighbors ($k$-NN) classifier. $k$-NN is an attractive technique because it is distribution-free and entirely data-driven.

The $k$-nearest-neighbors classifier can be easily adapted to use a different distance metric; the distance between points is no longer the Euclidean distance but instead any arbitrary distance function. The performance of the classifier is heavily dependent on the distance chosen. In this paper, we restrict the distance functions to be diagonally weighted Euclidean distances (that is, the Mahalanobis distance with a diagonal covariance matrix).

We propose to optimize the classification error of the $k$-NN classifier using a simple, novel optimization method based on existing techniques and involving random perturbations to avoid local minima. Then, we apply this method to perform phoneme classification on MFCC features from the TIMIT speech dataset [7] and show that the learned metric improves classification scores by up to 8 percentage points.

Because the distance learned using $k$-NN will also be applicable to a vast array of other machine learning methods that depend on distance functions, the choice of such a simple classifier is justified. We expect that the significant performance gains exhibited in $k$-NN will be reflected in other methods; for instance, support vector machines, kernel PCA, multidimensional scaling, or even dimensionality reduction techniques such as maximum variance unfolding (MVU), which depends highly on the distance between points. The performance of any scale-variant method should be improved using the distance learned by our method; we have only mentioned a few possibilities above.

In addition to the generalizability to any scale-variant method, it is also intuitive to see that we can generalize to other machine learning tasks. The technique we have chosen, the $k$-NN classifier, gives classifications as results. The learned distance, however, could also be easily applied to a method that gives likelihoods as results (such as the naive Bayes classifier); the use of likelihoods as opposed to discrete classification is a common approach in signal processing.

## 2. NEAREST NEIGHBORS CLASSIFICATION

The $k$-nearest-neighbors ($k$-NN) classifier is a nonparametric machine learning technique that is commonly used as a

baseline for comparison with more advanced techniques (for example, in [8]). Its simplicity and intuitiveness help make it a popular and attractive technique.

To formulate our problem, we assume that we have a dataset $\mathbf{X} \in \Re^{n \times d}$ which contains $n$ points of dimensionality $d$. We will refer to an individual point $i$ as $x_i$. We assume that this dataset contains some number of classes, and the true class label of a point $x_i$ is denoted as $y_i$.

To use this technique, we calculate the $k$ neighbors with minimum distance to our query point $x_i$ according to some distance function $d(x_i, x_j)$. Typically, the standard Euclidean distance is used. Then, we take a majority vote of the $k$ neighbors to predict the class of the query point $i$ ($\hat{y}_i$). We repeat this procedure for each query point, and if we have the true classes of these query points to compare our results with ($y_i$), we can define an error function using 0-1 loss:

$$E(\mathbf{X}) = \sum_{i=1}^{n} L(\hat{y}_i, y_i) \tag{1}$$

in which $L(\hat{y}_i, y_i)$ is the 0-1 loss function taking the value 1 only when $\hat{y}_i \neq y_i$. Therefore, our error measure is simply counting the total number of misclassified points.

## 3. DISTANCE LEARNING

As noted earlier, the Euclidean distance is the standard distance function used in $k$-NN. This is not a requirement, though; we can use any distance function $d(x_i, x_j)$. We will restrict our choice of distance functions specifically to the Mahalanobis distance, which is defined by

$$d(x_i, x_j)^2 = (x_i - x_j)^T M (x_i - x_j) \tag{2}$$

in which $M = A^T A$ is a positive semidefinite matrix. This is equivalent to weighting the dataset with the matrix $A$. For $A = I_n$, (2) simplifies to the standard Euclidean distance. It is clear that the choice of the matrix $A$ affects the error of our classifier, and we can conclude that there exists an optimal matrix $\hat{A}$ that minimizes the error function given in (1).

Therefore, our goal is to find $\hat{A}$ which minimizes the error function $E_A(\mathbf{X})$, in which $E_A(\mathbf{X})$ is $E(\mathbf{X})$ applied to $k$-NN using (2) as a distance measure. Due to the computational complexity of allowing any arbitrary matrix $A$, we will only consider the case where the matrix $A$ is restricted to be diagonal.

## 4. PERTURBATION-BASED OPTIMIZATION

The error function $E_A(\mathbf{X})$ is non-smooth, non-convex, and discontinuous, which presents a difficult problem for optimization. Figure 1 demonstrates this property. Using a small subset of the TIMIT database, we plot the error function $E_A(\mathbf{X})$ with respect to only one dimension of our weighting matrix $A$ (in Figure 1, we have chosen dimension 2).

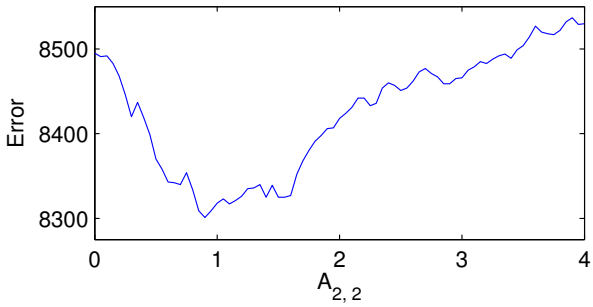


**Fig. 1**. Error function for different $A$ weights in dimension 2.

Other approaches to distance learning in the past reformulate the problem as a semidefinite program [1] or a smooth relaxation of the error function [4], but it is not impossible to optimize the error function directly.

The non-smooth characteristic of this error function means that the gradient does not exist; however, a gradient estimate can be assembled that can be used to find minima of the error function. Observing Figure 1, we can see that any gradient estimate is only valid locally. Further, we have no guarantee on the validity of a gradient estimate and therefore the problem lends itself to incorporating aspects of a perturbation-based approach (or an approach incorporating some elements of random search) in an attempt to avoid both local minima and bad steps based on inaccurate gradient estimates.

First, to approach the problem of the gradient estimate, we will use the finite-difference stochastic approximation (FDSA) approach, a common technique based on the work of Kiefer and Wolfowitz [9]. In this approach, we perturb $A$ in each dimension individually to assemble a gradient estimate $\hat{\nabla} E(\mathbf{X})$. The formula for one dimension of the gradient estimate is given below.

$$\hat{\nabla}_k E(\mathbf{X}) = \frac{E_{A+\alpha e_k}(\mathbf{X}) - E_{A-\alpha e_k}(\mathbf{X})}{2\alpha} \quad \forall 1 \leq k \leq d. \tag{3}$$

In this formula, $e_k \in \Re^{d \times d}$ represents the diagonal matrix of all zeros, except for the $k$th diagonal element, which is set to 1. Therefore, we are changing the weighting in the $k$th dimension and taking evaluations of the error function based on that. By doing this for all $k$ dimensions, we have a gradient estimate. The parameter $\alpha \in \Re$ controls the locality of the gradient estimate. For a function with many close stationary points, $\alpha$ should be set to a small value ($\leq 0.01$), and for a function with few stationary points, $\alpha$ may be set larger.

The final gradient estimate matrix $\hat{\nabla} E(\mathbf{X}) \in \Re^{d \times d}$ is assembled as a diagonal matrix with the $k$th diagonal elements set to $\hat{\nabla}_k E(\mathbf{X})$.

As a side observation, it is worth noting that any other method of finding a gradient estimate, such as simultaneous

perturbation stochastic approximation (SPSA) [10] or others, could be used; in our work, we found the best performance using an FDSA approach.

As noted earlier, we cannot use just this gradient estimate as a direction (which is what the standard FDSA method does) or we risk becoming stuck in local minima. Instead, we combine our gradient estimate with random perturbations in an attempt to avoid this issue.

Given that we are on iteration $t$ of our optimization and that $A_t$ represents the current weighting matrix, we use an inner iterative process (with the inner iteration number specified by $j$) to find a matrix $\tilde{A}_t^j$ which gives an improvement in $E_{\tilde{A}_t^j}(\mathbf{X})$. $\tilde{A}_t^j$ is chosen as follows:

$$\tilde{A}_t^j = A_t + \beta_n \left( \eta_j \left( -\hat{\nabla} E(\mathbf{X}) \right) + (1 - \eta_j) \Delta_j \right). \quad (4)$$

In this formula, $\Delta_j \in \Re^{d \times d}$ represents a random diagonal perturbation matrix. $0 \leq \beta_n \leq 1$ is the learning rate of the algorithm, which should decrease to 0 as $n \to \infty$. $\eta_j$ is a sequence starting from $\eta_0 = 1$ and decreasing to 0 as $j$ increases.

Qualitatively, we are seeking an improvement along the direction of our estimated gradient $\hat{\nabla} E(\mathbf{X})$; but because we know the gradient estimate is noisy, we are not guaranteed any improvement. As a result, we iteratively add larger and larger perturbations and depend less and less on our gradient estimate to find an improvement. For $\eta_j = 0$ (when $j$ is large enough), this method is equivalent to random search.

Once we have found $\tilde{A}_t^j$ (for some $j$) satisfying

$$E_{\tilde{A}_t^j}(\mathbf{X}) < E_A(\mathbf{X}) \quad (5)$$

we can update $A$ using the following simple rule:

$$A_{t+1} = \tilde{A}_t^j. \quad (6)$$

The algorithm terminates when we have searched unsuccessfully for a suitable $\tilde{A}_t^j$ for long enough ($j > \tau$ for some large $\tau$). We do not terminate when $||\hat{\nabla} E(\mathbf{X})|| = 0$ because we may be stuck in a local minimum, and we may find some $\tilde{A}_t^j$ that manages to escape it. In practice, we typically chose $\tau = \mathcal{O}(100\text{k})$.

A psuedocode implementation of the complete optimization algorithm is given below in Algorithm 1. This implementation uses a linear decrease for $\eta_j$ controlled by the parameter $\delta$, though this is not the only possible decrease scheme for $\eta$.

**Algorithm 1** Psuedocode implementation of the described optimization algorithm.

**Require:** $\alpha \geq 0, \beta \leq 1, \eta \leq 1, \delta \leq 1, \tau > 0, d > 0$
  $A_0 \leftarrow I_{d \times d}$
  $t \leftarrow 0$
  **loop**
    $\hat{\nabla} E(\mathbf{X}) \leftarrow \mathbf{0}_{d \times d}$
    **for** $k = 1 \to d$ **do**
      $E_+ \leftarrow E_{A_t + \alpha e_k}(\mathbf{X})$
      $E_- \leftarrow E_{A_t - \alpha e_k}(\mathbf{X})$
      $\left\{ \hat{\nabla} E(\mathbf{X}) \right\}_{k,k} \leftarrow \frac{E_+ - E_-}{2\alpha}$
    **end for**
    $j \leftarrow 0$
    $\eta_0 \leftarrow 1$
    **repeat**
      $\Delta_j \leftarrow \{\text{any random perturbation vector}\}$
      $\tilde{A}_t^j \leftarrow A_t + \beta_n \left( \eta_j \left( -\hat{\nabla} E(\mathbf{X}) \right) + (1 - \eta_j) \Delta_j \right)$
      $\eta_{j+1} = \max(\eta_j - \delta, 0)$
      $j \leftarrow j + 1$
      **if** $j > \tau$ **then**
        **return** $A_t$
      **end if**
    **until** $E_{\tilde{A}_t^j}(\mathbf{X}) \leq E_{A_t}(\mathbf{X})$
    $A_{t+1} \leftarrow \tilde{A}_t^{j-1}$
    $t \leftarrow t + 1$
  **end loop**

## 5. IMPLEMENTATION

The dataset chosen in this paper is the TIMIT dataset [7]. The speech features used were MFCCs (Mel Frequency Cepstral Coefficients) with delta and delta-delta values, the standard speech processing feature. Variance normalization was not performed. The Hidden Markov Model Toolkit (HTK) [11] was used to calculate the MFCCs. This gives a total dataset size of 1.7 million 39-dimensional points.

For the $k$-nearest-neighbors calculation, the dual-tree algorithm [12, 13] was used, which improves upon the $\mathcal{O}(n^2)$ naive approach to a more usable $\mathcal{O}(n)$. The chosen value of $k$ was 7; for the TIMIT dataset, any value between 5 and 9 produced approximately equal results for the $k$-NN classifier.

The update rule (4) was implemented with $\eta_j$ decreasing linearly from $\eta_0 = 1$ to $\eta_{200} = 0$. For $j \geq 200$, the perturbation-based search was equivalent to simple random search. In practice, however, the occurence of $j \geq 200$ was infrequent (this will be discussed more later). $\beta_n$ similarly decreased linearly from $\beta_0 = 0.7$ to a minimum value of $0.05$. The value of $\alpha$, the locality parameter, was found to perform best when chosen between $0.15$ and $0.1$.

The algorithm described above was run on a small subset of the TIMIT training dataset (12k points). We found that classification performance when optimizing on a small subset was equivalent to classification performance when optimizing on the whole training dataset (1.2M points).

When a final matrix $A$ was obtained, the TIMIT test set (500k points) was evaluated with $k$-NN and the learned distance using the training set (1.2M points) as labeled reference points.

## 6. RESULTS

Although the algorithm was run many times, the matrix $A$ tended to converge to similar classification accuracy improvement levels, even though the individual weightings produced by the algorithm tended to differ. We will focus on three of the highest-scoring learned distances, which we denote as $A_1$, $A_2$, and $A_3$. Table 1 shows the resulting phoneme classification accuracy for these matrices, as well as the baseline phoneme classification accuracy ($I_{39}$).

In addition, other state-of-the-art methods for distance learning were used in the same problem setting for comparison. The resultant performance improvements for those methods are also given. Weinberger's implementation of distance learning for large margin nearest neighbor classification (the 'LMNN2' software package) [1] was used with the nearest neighbors parameter $k = 3$ and $k = 7$ until convergence. Neighborhood Components Analysis [4] was also used for comparison, and the results of using the FDSA-based gradient descent algorithm [9] are additionally shown. For these comparisons, $k$-NN classification was run with the distance weightings produced by each on the 500k point TIMIT test set.

| | Accuracy (%) | Improvement (%) |
|---|---|---|
| Baseline ($I_{39}$) | 41.684% | |
| $A_1$ | 49.863% | 8.19% |
| $A_2$ | 49.845% | 8.16% |
| $A_3$ | 49.778% | 8.09% |
| LMNN2, $k = 3$ | 44.690% | 3.00% |
| LMNN2, $k = 7$ | 45.730% | 4.05% |
| NCA | 45.766% | 4.08% |
| FDSA | 45.938% | 4.25% |

**Table 1**. Classification accuracy for learned metrics.

Figure 2 shows the diagonals of the matrices $A_1$, $A_2$, and $A_3$ plotted against the standard identity matrix $I_{39}$. A higher weight indicates a higher importance of that dimension in classification. Each of the three plotted matrices ($A_1$, $A_2$, and $A_3$) produced at least $8$ percentage points of improvement in classification over the standard $k$-nearest-neighbors result ($I_{39}$). The average classification improvement over all runs of our optimization was $7.68\%$.

It is discernible that the first set of 13 dimensions, in each trial, were generally found to be less important (hence, lower weights) than the second or third set of 13 dimensions. The weightings for dimensions 14 through 39 (which represent the delta and delta-deltas of the MFCCs) were much more variable. From this we can postulate that our algorithm found the delta and delta-delta MFCCs to contain more discriminatory information than the 13 MFCCs themselves.

Table 2 shows the classification results given by each of the three selected matrices for each class of phonemes. The results for the unweighted case ($A = I_{39}$), are given for comparison. The phoneme class results are the combined results of classification of the individual phonemes that make up the particular class. The phonemes are split into the same classes given by the TIMIT documentation [7].

| Phoneme Class | $A_1$ | $A_2$ | $A_3$ | $I_{39}$ |
|---|---|---|---|---|
| Stops | 38.29% | 37.88% | 37.56% | 21.82% |
| Affricatives | 28.69% | 27.87% | 28.12% | 14.07% |
| Fricatives | 65.28% | 65.13% | 64.98% | 59.32% |
| Nasals | 45.11% | 44.94% | 45.45% | 33.85% |
| Glides | 48.32% | 47.89% | 48.07% | 36.95% |
| Vowels | 38.76% | 39.01% | 38.90% | 31.28% |

**Table 2**. Classification results for different phoneme classes.

We see a maximum improvement of $16.47\%$ for stops (the highest class-specific improvement shown), $14.62\%$ for affricatives, $5.96\%$ for fricatives, $11.60\%$ for nasals, $11.37\%$ for glides, and $7.73\%$ for vowels.

## 7. DISCUSSION

It is clear from the results that our method has produced a significant increase in accuracy for nearest-neighbors classification. Most runs of the algorithm produced classification gains between 7 and $8$ percent. As shown before, however, the most optimal solution we found resulted in a classification improvement of $8.19$ percent, which is a non-negligible improvement. In addition, this is a significant improvement over the results of other state-of-the-art distance learning algorithms.

Observing the phoneme class results in Table 2, we can see that the most significant gains were achieved with stops. A stop can be characterized intuitively not by its frequencies but instead by the changes over time, unlike a vowel, which is mainly constant over time. Therefore, it is reasonable to conclude that the improved classification performance on stops is related to the fact that our algorithm generally weighted dimensions 14 through 39 highly. Interestingly, $A_2$, which performed best on vowels, shows higher weightings for the first 13 coefficients. This supports the intuitive idea that more discriminatory information for a steady-state phoneme (such as a vowel) is contained in the first 13 MFCCs.

In Figure 2 we observe drastically different weightings in the delta and delta-delta coefficients for each matrix $A_1$, $A_2$, and $A_3$. This result would seem to indicate that each matrix would perform better for different phonemes. Indeed, Table 2 confirms this, showing that $A_2$ performs best for vowels, $A_3$ performs best for glides, and $A_1$ performs best for all other classes.

This apparent specialization of matrices demonstrates that our algorithm has converged to different local minima, though the performance of each matrix is comparable. The existence

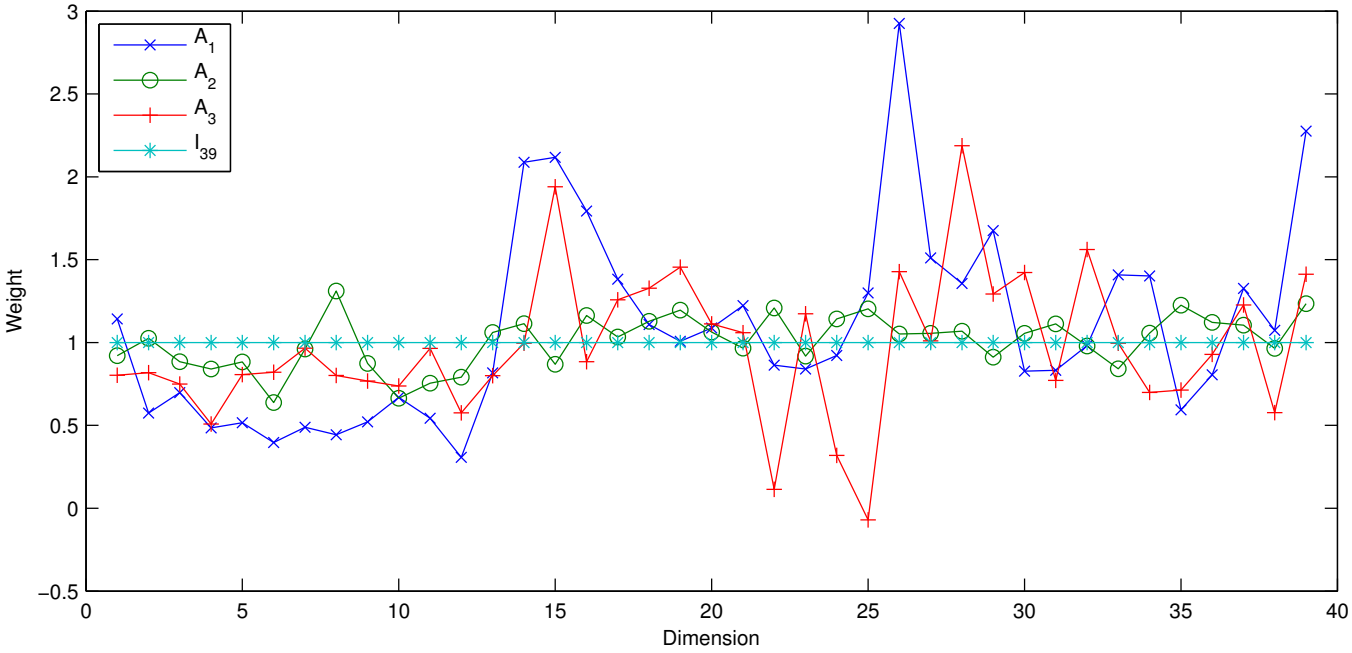


**Fig. 2**. Final weights for three different trials.

of local minima so far apart further demonstrates the non-convexity of the error function, and justifies our addition of a stochastic term to our optimization's direction. Comparison with the results obtained by the FDSA-based gradient descent algorithm justify this further; in Table 1 we see that the FDSA-only algorithm becomes trapped in a local minimum giving only $4.25\%$ improvement. The stochastic perturbation, though, is able to escape it, allowing the algorithm to find a far better local minimum.

## 8. COMPUTATIONAL COMPLEXITY

Computational complexity difficulties are a common issue for distance learning. For instance, Weinberger's formulation of distance learning as a semidefinite problem [1] unfortunately has significant complexity issues and suffers greatly as $k$ is increased. Neighborhood Components Analysis is known to have quadratic complexity, and in our trials we found NCA to perform significantly slower than both LMNN and our own method.

In spite of our efforts to keep the computational load minimized by using fast nearest-neighbors algorithms [12, 13], the evaluation of our error function $E_A(\mathbf{X})$ is still not trivial, requiring 1 or 2 seconds on relatively recent hardware for one hundredth of the TIMIT dataset (12k points). Like many random-search optimization methods, our algorithm can spend many iterations searching for a matrix $\tilde{A}_t^j$ (in Equation 4) before finding an improvement. In practice, we found that when $A$ becomes trapped in a local minimum, up to tens of thousands of iterations could be required before an improvement is found. This maps to computing time of an hour to a several hours before a step is made (or the algorithm terminates unsuccessfully).

We observed that many steps would be taken requiring only a few inner iterations for $\tilde{A}_t^j$ (meaning that the gradient estimate produced a good step), but only occasionally would a step require thousands of inner iterations. This is opposed to pure random search, which generally required hundreds to thousands of inner iterations for each step. To address the occasional lengthy random search for a suitable step, $\tau$ was reduced to 200, disallowing entirely random search but still allowing perturbations. This still resulted in significant accuracy improvement; the maximum improvement seen was $7.5\%$ over $I_{39}$. Leaving $\tau$ large gave the best results, although it does incur the potentially significant computational costs associated with random search.

## 9. EXTENSIONS AND CONCLUSIONS

In this work we have shown that we can improve the performance of the $k$-nearest-neighbor classifier for phoneme recognition by learning a new distance metric. We use a (to our knowledge) novel optimization method based on FDSA [9] which incorporates a random perturbation to avoid local minima. The method is shown to produce performance improvements of up to $8.19$ percentage points, with the highest gains produced in the phoneme class of stops ($14.25\%$).

While the new performance of the weighted $k$-NN classifier is by no means impressive (with an accuracy of $49.863\%$ for the best learned metric), our method will generalize to

any scale-variant machine learning method. It is clear that our method outperforms other state-of-the-art distance learning methods by approximately 4 percentage points, at least in the problem setting we have used.

The results here can be directly applied to existing scale-variant machine learning methods. In the introduction, we discussed several machine learning methods that would benefit from our work. The use of support vector machines for phoneme classification has produced good results in the past, and a distance metric learned by our algorithm is likely to improve SVM results. Preliminary results seem to indicate that our method provides some level of improvement for SVMs.

Maximum variance unfolding (MVU) [14] is another example of a scale-variant method that has been applied to speech for the task of dimensionality reduction [15]. It is likely that applying a weighting matrix learned with our algorithm would produce better results for MVU and may lead to an effective lower-dimensional representation of speech. work includes extending our results to other methods (specifically, but not exclusively, MVU) and evaluating the performance improvements.

Future work will investigate the extension of our results to other machine learning methods (specifically, but not exclusively, MVU) and evaluating the performance improvements. In addition, the performance of our method in problem settings in which competing distance learning algorithms were proposed will be evaluated; this will give a better evaluation of the generalizability of our method to non-speech applications.

## 10. REFERENCES


[1] Kilian Q. Weinberger and Lawrence K. Saul, "Distance Metric Learning for Large Margin Nearest Neighbor Classification," *Journal of Machine Learning Research*, vol. 10, pp. 207–244, 2009.

[2] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell, "Distance Metric Learning, With Application To Clustering With Side-Information," in *Advances in Neural Information Processing Systems 15*. 2002, pp. 505–512, MIT Press.

[3] Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon, "Information-theoretic metric learning," in *Proceedings of the 24th International Conference on Machine Learning (ICML 2007)*, New York, New York, USA, 2007, pp. 209–216, ACM Press.

[4] Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov, "Neighbourhood Components Analysis," in *Advances in Neural Information Processing Systems 17*. 2004, pp. 513–520, MIT Press.

[5] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Machine Learning*, vol. 46, no. 1, pp. 131–159, 2002.

[6] G. Chechik, V. Sharma, U. Shalit, and S. Bengio, "An online algorithm for large scale image similarity learning," in *Advacnes in Neural Information Processing Systems 22*. Citeseer, 2009.

[7] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren, "DARPA TIMIT acoustic phonetic continuous speech corpus CDROM," 1993.

[8] Yi-lin Lin and Gang Wei, "Speech emotion recognition based on HMM and SVM," in *2005 International Conference on Machine Learning and Cybernetics (ICMLC 2005)*. 2005, pp. 4898–4901, IEEE.

[9] J. Kiefer and J. Wolfowitz, "Stochastic Estimation of the Maximum of a Regression Function," *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 462–466, 1952.

[10] James C. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Transactions on Automatic Control*, vol. 37, no. 3, pp. 332–341, 1992.

[11] P.C. Woodland and S.J. Young, "The HTK Tied-State Continuous Speech Recogniser," in *Third European Conference on Speech Communication and Technology (EUROSPEECH '93)*, Berlin, Germany, 1993, pp. 2207–2210.

[12] Alexander G. Gray and Andrew W. Moore, "'N-Body' problems in Statistical Learning," in *Advances in Neural Information Processing Systems 14*. 2001, vol. 4, pp. 521–527, Citeseer.

[13] Parikshit Ram, Dongryeol Lee, William B. March, and Alexander G. Gray, "Linear-time algorithms for pairwise statistical problems," in *Advancies in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, Eds., 2009, pp. 1527–1535.

[14] Kilian Q. Weinberger, Fei Sha, and Lawrence K. Saul, "Learning a kernel matrix for nonlinear dimensionality reduction," in *International Conference on Machine Learning (ICML 2004)*. 2004, pp. 106–114, ACM Press.

[15] Nikolaos Vasiloglou, David V. Anderson, and Alexander G. Gray, "Learning the Intrinsic Dimensions of the TIMIT Speech Database with Maximum Variance Unfolding," in *Digital Signal Processing Workshop*, 2009, pp. 48–53.