Kernel Discriminant Functions for Classification

Bernard Lampe, *Member, IEEE*

***Abstract*—In this study we investigate the effectiveness of augmenting support vector machine (SVM) and Fisher linear discriminant analysis (FLDA) classifiers using known kernel methods. Strictly speaking Fishers is a data reduction technique, and we extend it to build a classifier. The extension was done in two ways using Euclidean and Mahalanobis distance metrics. We test the three well known kernels of polynomial, tangent hyperbolic sigmoid and radial basis function (RBF). We judge the effectiveness of these kernels and classifiers on a five class problem based on a biased coin which implies binomial class distributions of differing means and variances.**

***Index Terms*—Multi-class Classification, Kernel Fisher Linear Discriminant Analysis, Kernel Support Vector Machine, KFLDA, KSVM**

I. Introduction

FISHERS linear discriminant analysis (FLDA) and support vector machines (SVM) aim to find an optimal linear hyperplane which will provide the best discriminating power between labeled class instances. These supervised training methods perform very well when the classes can be linearly separated. However, many real-world problems have non-linear or inseparable data when considering data in the original feature space. To overcome these non-linearities we could try to use more complicated parametric models to approximate the class conditional probabilities and then construct a Bayes classifier. This method gives us little idea on what model to choose or how to extend an analytical form which will remain computationally tractable. We could also attempt to estimate the class conditionals using non-parametric methods. However, this approach requires an exponential number of data samples from each class as the dimensionality of the feature space is increased. A more effective approach is to make the data linearly separable by mapping the data to a higher dimension feature space F via non-linear mapping $\Phi(\vec{x^n})$. If F is of a very high dimensionality or infinitely dimensional, the mapping will be impossible to compute. To overcome this limitation a kernel trick is used to compute dot products $\Phi(\vec{x^i}) \cdot \Phi(\vec{x^j})$ between vectors in the feature space F without explicitly computing the vectors themselves. This is accomplished via Mercer kernels [8] [9]. A discussion of Mercers theorem, FLDA and SVM is beyond the scope of this report and there are many very thorough references [1] [4] [8].

II. Training Data and Test Sample Generation

In this study the training data was simulated by flipping a hypothetical coin $d = 100$ times. The training observations were labeled with the target value $t = i$ which indexes the class C_i. We consider the five class problem and the data was generated by flipping five different, biased coins. The biases are in the set $\Theta = \{0.1, 0.2, 0.3, 0.4, 0.5\}$. The data pool consists of 3000 samples each of 100 flips such that $\Delta_i = \{\vec{x}_i^j\}_{j=1}^{200i}$ and $\Delta = \cup_{i=1}^{5}\Delta_i$. The n-th training sample of Δ is a tuple of the sample vector and target value $(\vec{x}^n, t^n)$. For all training runs, the data was partitioned into two equal sized groups. Half of the training samples were used to train the classifier. The other half were used as new test observations.

III. Kernel Function Formulations

There are many Mercer kernels which can be used to perform the kernel trick. To speed computation we compute the Gram matrix which is essentially applying the kernel trick to all pairs of training observations in Δ. We implemented the polynomial kernel as in equation 1, the tangent hyperbolic sigmoid kernel as in equation 2 and the radial basis function (RBF) kernel as in equation 3. Each KFLDA and KSVM method we implemented in this study applied the kernel to the data and not the feature vectors.

$$K(\vec{x_i}, \vec{x_j}) = (\gamma \vec{x_i}^T \vec{x_j} + c)^d \tag{1}$$

$$K(\vec{x_i}, \vec{x_j}) = \tanh(\gamma \vec{x_i}^T \vec{x_j} + c) \tag{2}$$

$$K(\vec{x_i}, \vec{x_j}) = \exp\left(-\frac{\|\vec{x_i} - \vec{x_j}\|}{2\sigma^2}\right) \tag{3}$$

IV. KFLDA Classification Distance Metrics

To extend the kernel Fishers linear discriminant analysis (KFLDA) from a data reduction method to a classifier, we start by computing the KFLDA feature vectors as described in the references by Mika, Max, Baudat or Park [9] [11] [2] [10]. We have five classes therefore the subspace has four feature vectors and dimensions. We compute the projected means, $\vec{\mu_i}$ and covariances $\vec{\Sigma_i}$ for each class in the subspace. After we have the feature vectors, projected class means and projected covariances, we can dismiss the original training data. Finally to classify new samples, we project new sample into the subspace made up of these four feature vectors and compute the distances to each class using the means and covariances. We derive the distance metric by starting with the optimal Bayes classifier as in equation 4. For computational tractability we then assume the class conditionals are effectively modeled by a Gaussian. This assumption is not unfounded due to the binomial nature of the random variable. If we combine equations 4 and 5 then we arrive at equation 6. We refer to this distance throughout the remaining portion of this paper as the Mahalanobis distance though strictly speaking, this distance is closer related to the Bhattacharyya distance [3, p. 166].

$$C^* = \arg\max_{C_i}\{log(p(\vec{x}|C_i)) + log(p(C_i))\} \tag{4}$$

$$p(\vec{x}|C_i) = \frac{1}{(2\pi)^{\frac{d}{2}}} |\Sigma_i| \exp\left\{-\frac{1}{2}(\vec{x} - \vec{\mu_i}^T \Sigma_i^{-1}(\vec{x} - \vec{\mu_i})\right\} \quad (5)$$

$$C^* = \arg\max_{C_i}\{ - \frac{1}{2}log(|\Sigma_i|) - \frac{1}{2}(\vec{x} - \vec{\mu_i})^T\Sigma_i^{-1}(\vec{x} - \vec{\mu_i}) \quad (6)$$
$$+ \, log(p(C_i))\}$$

To compute the Mahalanobis distance from a new observation $\vec{x}$ to each class C_i using the formula in 6, we need to estimate the class priors, means and covariances using the maximum likelihood estimators in equations 8, 9 and 10. In each of the equations below, W is the $3000x4$ matrix consisting of feature vectors computed via KFDLA and W_n^T is a $4x1$ vector from a single row in W.

$$N_i = \sum_{n=1}^{N} r_i^n, \text{ where } r_i^n = \begin{cases} 1, & \text{if } \vec{x}^n \in C_i \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

$$p(C_i) \approx \hat{p}(C_i) = \frac{N_i}{N} \quad (8)$$

$$\vec{\mu_i} = W^T\vec{\mu_i}^{\phi} \approx \frac{1}{N_i}\sum_{n=1}^{N}\sum_{k=1}^{N} r_i^n W_n^T K(\vec{x_n}, \vec{x_k}) \quad (9)$$

$$\Sigma_i \approx \frac{1}{N_i - 1}\sum_{n=1}^{N}\sum_{k=1}^{N} r_i^n(W_n^T K(\vec{x}_n, \vec{x_k}) - \vec{\mu_i})(W_n^T K(\vec{x_j}, \vec{x_k}) - \vec{\mu_i})^{\prime} \quad (10)$$

Finally, if we assume that the covariances are equal between each class (i.e. the "shape" of the class distributions are equal) and we assume the variances are the same between the dimensions (i.e. the class distributions are spherical) and the priors are the same between classes, then we arrive at the Euclidean distance metric as in equation 12. In this study, we used both Euclidean 12 and Mahalanobis 6 distance metrics to evaluate the KFLDA classifiers under differing parametrized kernels and data set sizes.

$$\Sigma_i = \Sigma_j, \sigma_i = \sigma_j, p(C_i) = p(C_j), \forall i \neq j \quad (11)$$

$$C^* = \arg\min_{C_i}\{\|\vec{x} - \vec{\mu_i}\|_2\} \quad (12)$$

V. KFLDA and SVM Implementations Used

There are many different formulations of the KFDLA problem. Each use the Gram matrix and Mercer kernels to compute the feature vectors. We implemented four different ways to compute the projection matrix W. The first and most widely known is by Mika [9]. The remaining three where by Max [11], Baudat [2] and Park [10]. We refer to each KFLDA algorithm by the leading author on the publication. The Mika and Max implementations are similar and each other and both require a suitable regularization constant to be found. The Park and Baudat implementation are similar and both rely on the singular value decomposition (SVD). Park and Baudat do not require a regularization constant. We also investigated the popular LibSVM [5] which does one-to-one classification and compared that to true Multi-Class SVM techniques [6]. We discovered one other multi-class SVM implementation called MSVM-Pack [7] but the run times of this implementation were prohibitive on our problem size.

VI. Optimizing Best Kernel Parameters

Presumably, the kernel parameters which give the best classification rate is positively correlated with the hyperplane in feature space F that separates the class data with the maximal margin. For the polynomial kernel, we set $c = 0$ and $gamma = 1$. Then we varied the polynomial degree d from 1 to 10. After observing the classification rates in figure 1, the rate at $d = 2$ is highest rate for a non-linear kernel. Another interesting feature of this graph is that after $d = 4$ or higher, the classification rate varies highly for some algorithms. We found that performing an inversion of a highly non-linear matrix had some numerical instability issues. We further computed these same tests using the Mahalanobis distance and found the same trends of classification rate and numerical instability. The graph of Mahalanobis distances is not included here.

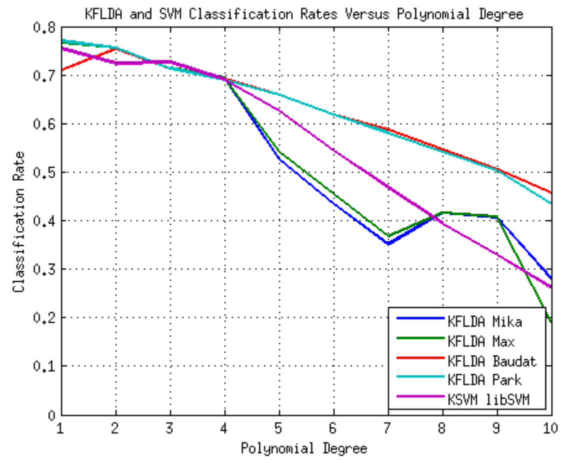

Fig. 1: Searching For Best Polynomial Degree from 1 to 10. Classification Rates Versus Polynomial Degree. KFLDA using Euclidean distance.

For the tangent hyperbolic sigmoid kernel we precede as in the polynomial case and set the parameter of $c = 0$ and then vary the amplitude $gamma$ from 0.01 to 1. These limits were found empirically to give non-constant results and after 1 the classification rate was constant. This is not unexpected given the shape of the sigmoid function. You can see from the graph in 2 that the lower number give better results. We chose $\gamma = 0.01$ when using this kernel. Also, notice the SVM and Baudat implementations skew sharply after the origin. For Baudat, this is due to the fact that we are doing a Eigenvalue decomposition on the Gram matrix which is not of full rank and we assume the same for the SVM.

For the RBF kernel we precede as before and varied the RBF variance σ from 1 to 10. After examining figure 3 we

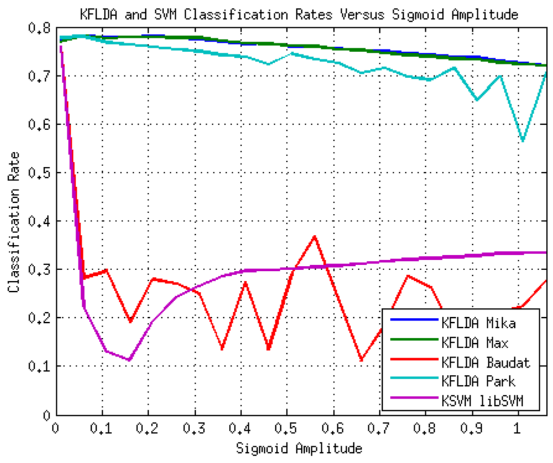

Fig. 2: Searching For Best Sigmoid Amplitude 0.01 to 1. Classification Rates Versus Amplitude. KFLDA using Euclidean distance.

see the best classification rate was chosen at $\sigma = 3$. Also, note the consistency of the result among the different algorithms. This is attributed to the fact that the Gram matrix will have full rank in most cases because the RBF of a vector with itself will be $e^0 = 1$.

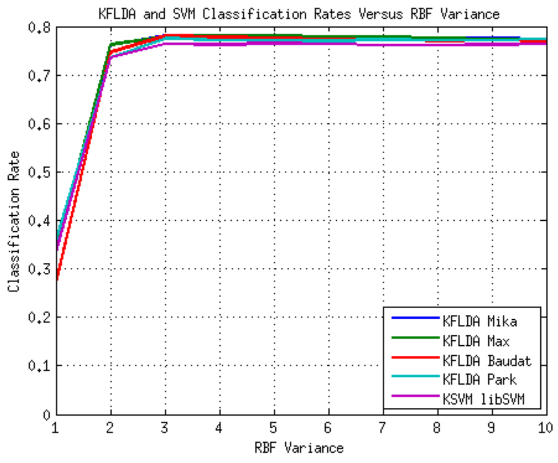

Fig. 3: Searching For Best RBF Variance from 1 to 10. Classification Rates Versus Variance. KFLDA using Euclidean distance.

VII. Algorithm Performance

We evaluated the classifiers over a range of data set sizes. These subsets were chosen by randomly selecting a percentage of training samples from each class. Then the classification rate was computed by classifying all of observations in Δ. You can see the performance of the classifiers in figures 4, 5 and 6. Among all the algorithms and data sizes it appears that KFLDA outperformed KSVM. Also, all the kernel methods outperformed the non-kernel FLDA and SVM when the amount of training data was low and using the RBF and polynomial kernels. The sigmoid kernel performed worst of all three for KSVM, but best overall for KFLDA. We computed the same curves using the Mahalanobis distance, and found the same trends. These graphs are not included here due to space.

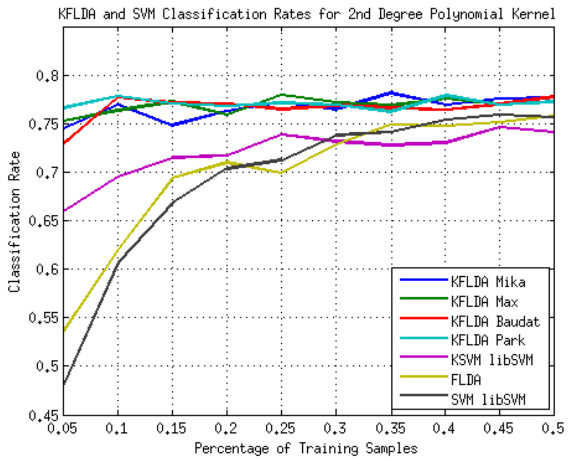

Fig. 4: Polynomial Kernel $degree = d = 2$. KFLDA using Euclidean distance. Classification Rates Versus Class Subset Size as Percentage of Each Class.

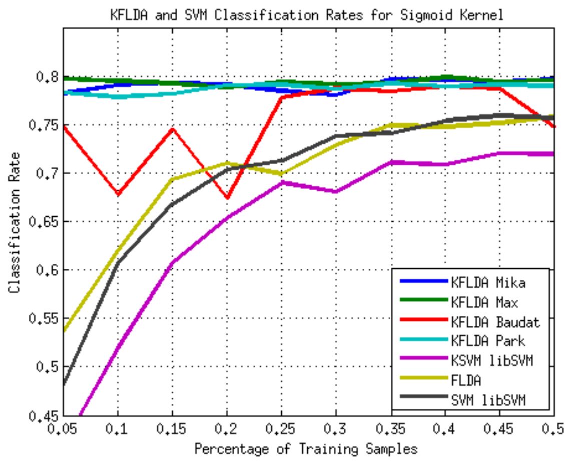

Fig. 5: Sigmoid kernel $Amplitude = \gamma = 0.01$. KFLDA using Euclidean distance. Classification Rates Versus Class Subset Size as Percentage of Each Class.

VIII. SVM Investigations, One Sample and Multi-Class

We also investigated some of the limits of SVM using only five support vectors, one training sample from each class. For each kernel, we choose five random support vectors and then trained and classified all of Δ 100 times. The results are shown in figure 7. We ran LibSVM with each kernel using the optimized parameters we found when optimize for the entire data set Δ and found the classification rate was around 20%. However, if we re-optimized the kernel parameter search using just five support vectors we arrive at the classification rates in columns 5, 6 and 7 which vary from 35% to 43%. This data

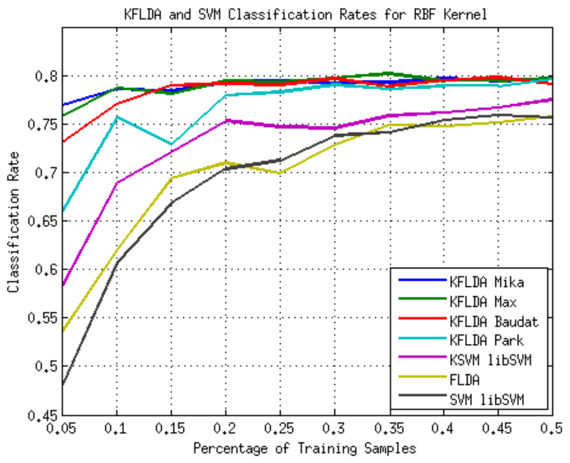

Fig. 6: RBF kernel $Variance = \sigma = 3$. KFLDA using Euclidean distance. Classification Rates Versus Class Subset Size as Percentage of Each Class.

strongly suggests that you need to retrain your classifier and optimize your kernel parameters for each training data set size.

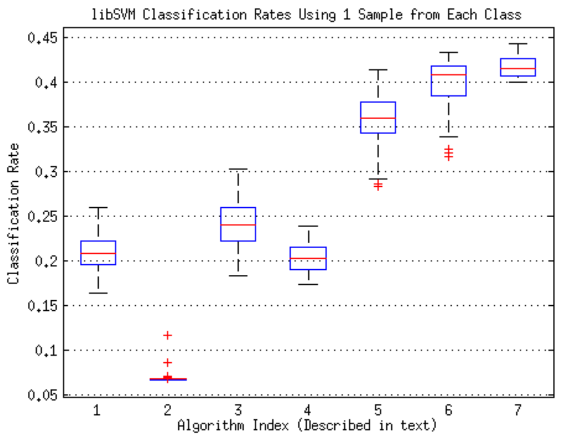

Fig. 7: LibSVM using 1 sample from each class. (1) No kernel, (2) Polynomial $\gamma = 1, c = 0, d = 2$, (3) Sigmoid $\gamma = 0.01, c = 0$, (4) RBF $\sigma = 3$, (5) Polynomial $\gamma = 1, c = 0, d = \sqrt{2}$, (6) Sigmoid $\gamma = 0.0331, c = 0$, (7) RBF $sigma = 0.15$

During our testing it became known that LibSVM does a one-to-one decomposition of the multi-class problem. We wanted to assess if a true multi-class approach could do better. Therefore, we found MCSVM by Koby Crammer [6] and MSVM-Pack by Lauer and Guermeur [7]. We ran MCSVM for all three kernels at the optimized parameters using all of Δ for training and testing as before. The classification rates are shown in figure 8. The figure shows that RBF does the best job, and sigmoid and polynomial do not perform well. We attempted to use the MSVM-Pack implementation, but found the run time to be prohibitive. From runs of very small data sets, we found anecdotal data that suggests that MSVM-Pack performs as well as MCSVM, but more work would be need to substantiate that claim fully.

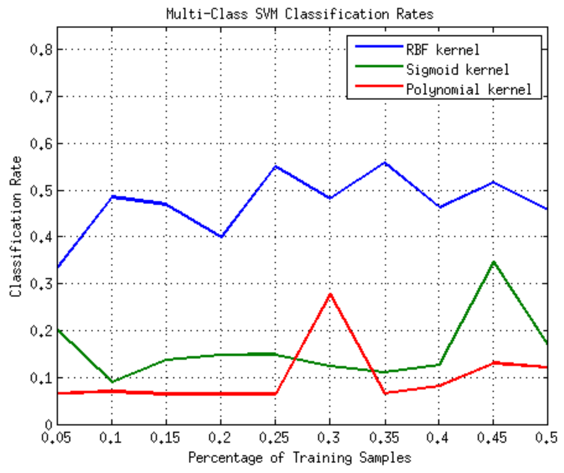

Fig. 8: Multi-Class SVM using Crammer [6] for all kernels.

IX. Classifications Rates per Class

The above analysis only considers the overall classification rate for each data subset, algorithm and kernel. We also investigated the individual classification rates to see if one class was dominating the others. Figures 9 and 10 show the individual class classification rates for KFLDA and SVM using the RBF and differing data sets sizes. Both graphs show that classes one and five are dominating the correct classification rate. Due to the linear nature of the data, classes one and five only share one hyperplane between itself and the remaining classes. While classes two, three and four share two separating hyperplanes with the remaining classes.

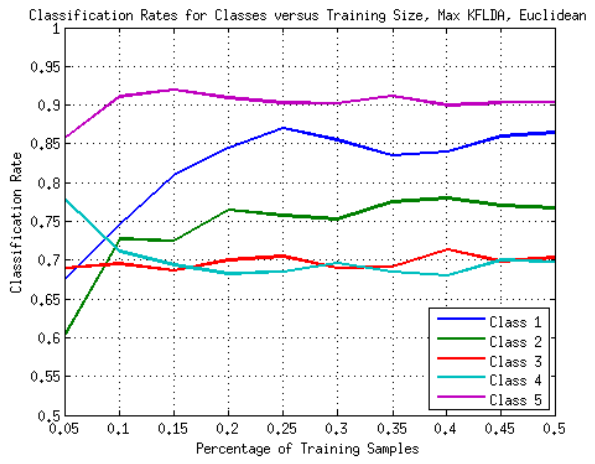

Fig. 9: Individual Classification Rates for each Class using RBF kernel $\sigma = 3$, KFLDA by Max [11] and Euclidean Distance Metric.

X. Algorithm Run Time Performance

We found the run times of the classifiers varied depending on the implementation. For example, LibSVM is written in

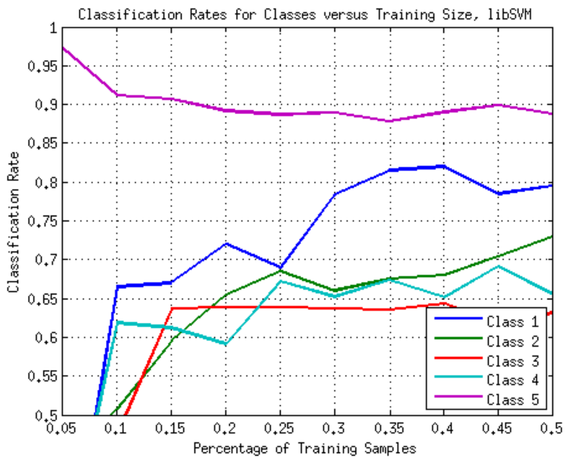

Fig. 10: Individual Classification Rates for each Class using RBF kernel $\sigma = 3$, libSVM [5].

C code while all the KFLDA implementations were done in Matlab. The run times versus data size are plotted in figure 11. You can see that FLDA and SVM had similar run times. This is interesting because FLDA was computed using Matlab code. This implies that the dominating run time component for the KFLDA implementations is the Gram matrix computation and Eigenvalue decompositions and SVD. Notice that the Park and Baudat had the worst run times but also do not require a regularization constant. Also, notice that Mika and Max has similar run times and similar classification performance. Finally, the run time of the Gram matrix computation was evaluated and found, as expected, the RBF was higher than the sigmoid and polynomial was the lowest.

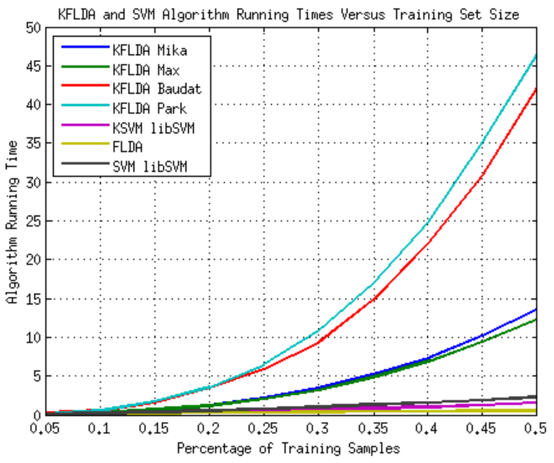

Fig. 11: Algorithm run times for KFLDA Algorithms, libSVM and FLDA versus Class Subset Size as Percentage of Each Class.

XI. Conclusion

In this study, we investigated the improvements that could be achieved using a kernel based approach to SVM and FLDA classifiers. We found that using a kernel allows a better classification rate with fewer samples, but does not improve overall classification rate when FLDA or SVM has sufficient samples. Also that optimizing kernel parameters for each training data set size is a good practice.

The Mika, Max, Baudat and Park KFLDA did best overall and the sigmoid kernel did the best using KFLDA, but the worst using KSVM. The Baudat and Park KFLDA do not require a regularization, but have high run time complexities. We were also surprised by how well that KSVM does using only one support vector per class.

Even though we did not include the Mahalanobis graphs in the paper, we discovered Mahalanobis distance did not do as well as Euclidean distance. However, the other distance metric did show the same trends overall. To investigate this further, we did a run using a data set using an equal number observations from each class (i.e. we made the priors equal). This shows the same trend that the Mahalanobis distance still did not do as well. We concluded that the assumption of a Gaussian class conditional probability model may be flawed when using a kernel. However, the Gaussian model is robust to non-Gaussian characteristics, so this may imply that the data is highly skewed. More work will need to be done to verify this finding.

References

[1] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, London, England, 2004.
[2] G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Comput.*, 12(10):2385–2404, October 2000.
[3] Chein-I Chang. *Hyperspectral Imaging, Techniques for Spectral Detecction and Classification*. Kluwer Academic/Plenum Publishers, New York, NY, USA, 2003.
[4] Chein-I Chang. *Hyperspectral Data Processing*. Jone Wiley & Sons, Inc., Hoboken, NJ, USA, 2013.
[5] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.
[6] Koby Crammer. Mcsvm c code for multiclass svm. 2003. Software available at http://www.cis.upenn.edu/~crammer.
[7] F. Lauer and Y. Guermeur. MSVMpack: a multi-class support vector machine package. *Journal of Machine Learning Research*, 12:2269–2272, 2011. http://www.loria.fr/~lauer/MSVMpack.
[8] J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 209(441-458):415–446, 1909.
[9] Sebastian Mika, Gunnar Rtsch, Jason Weston, Bernhard Schlkopf, and Klaus-Robert Mller. Fisher discriminant analysis with kernels, 1999.
[10] Cheong Hee Park and Haesun Park. Nonlinear discriminant analysis using kernel functions and the generalized singular value decomposition. *SIAM J. Matrix Analysis Applications*, 27(1):87–102, 2005.
[11] Max Welling. Fisher linear discriminant analysis, 2006.

Bernard Lampe (M'09) became an IEEE Member (M) in 2009 and received his bachelors of science degree from The University of Michigan in Ann Arbor, Michigan, USA in 2009. Mr. Lampe is also a member of the American Society for Computing Machines (ACM) since 2009.