Performance Comparison of Multilayer Perceptron and Kernel Linear Discriminant Analysis for Classification of 2D and 5D Data

Sivan Zlotnikov
Department of Electrical and Computer Engineering, Temple University
Sivan.Zlotnikov@temple.edu

Introduction: For this final project, the classification performance of neural network (NN) and non-NN machine learning algorithms are explored. For training and evaluation of the selected algorithms, two data sets are considered, a 2-dimentional and a 5-dimentional data set. For each given set, three *.txt* files are provided which contain data for training and evaluation. These files are *train, dev,* and *eval*, and the number of data points in each file are displayed in Table 1. For the NN approach, Multilayer Perceptron (MLP) was used, while, Kernel Linear Discriminant Analysis (KLDA) was implemented as the non-NN approach. Original code was written for both algorithms and implemented in MATLAB.

	Data Set		
Dimension	**Train**	**Dev Test**	**Eval**
2	10,000	2,000	2,000
5	100,000	10,000	10,000

Table 1. Data Set Dimensionality and Number Of Data Points

Kernel Linear Discriminant Analysis (KLDA): Various algorithms, such as Principle Component Analysis (PCA) and Linear Discriminant Analysis (LDA) were explored for the non-NN, however, using these linear classifiers, an error rate below 50% could not be achieve. In order to find a non-linear decision boundary, a kernel function was used as a pre-processing step in order to increase the dimensionality of the data. The two kernel functions that were used to improve the error rate were the radial basis function (RBF) and polynomial kernel functions shown in equations 1 and 2. Using a fine grid search approach, different values for σ and d were tested following classification with PCA and LDA, resulting in kernel PCA (KPCA) and kernel LDA (KLDA).

$$K\left(\boldsymbol{x}_i, \boldsymbol{x}_j^{\boldsymbol{T}}\right) = \exp\left(-\frac{\left\|\boldsymbol{x}_i - \boldsymbol{x}_j^{\boldsymbol{T}}\right\|^2}{2\sigma^2}\right) \quad (1)$$

$$K\left(\boldsymbol{x}_i, \boldsymbol{x}_j^{\boldsymbol{T}}\right) = \left(\boldsymbol{x}_i^{\boldsymbol{T}} \cdot \boldsymbol{x}_j\right)^d \quad (2)$$

For the 5D data, 100,000 training points are available, therefore, the size of the resulting kernel matrix is 100,000 x 100,000, however, sufficient memory was not available in order to obtain a kernel matrix of this size. To overcome memory limitations, 3,000 training points per class were chosen at random and used to construct the reduced training matrix $\boldsymbol{x}_i$, while $\boldsymbol{x}_j$ was the entire training set. This resulted in a 6,000 x 100,000 kernel matrix for training, increasing the dimensionality of the 5D data to 6000 dimensions. Similarly, the kernel matrices for classifying the *dev* and *eval* data were constructed, where $\boldsymbol{x}_i$ are the randomly selected training points, and $\boldsymbol{x}_j$ was either the entire *dev* or *eval* data matrix. For LDA, the Nearest-Centroid classifier was used while, for PCA, the Nearest-Subspace classifier was used. The best performance for this non-NN approach was achieve using RBF kernel followed by LDA, with $\sigma = -5.862$. For the 2D data, the entire was used for construction of the training kernel matrix, and similarly to the 5D data, the best performance was obtained using RBF kernel followed by LDA, with $\sigma = -4.421$. The achieved error rates are recorded in Table 3.

Multilayer Perceptron (MLP): Only one approach was test for NN, and that is the MLP network. In addition to determining the number of hidden layers and the number of nodes in each layer, the learning rate and the decay rate need to be determined. Using random values for the number of hidden layers and the number of nodes in each layer, a fine grid search over a range of values for the learning rate and the decay rate was used. The parameters that resulted in the lowest error rate for the training data where chosen as the optimal parameters. These parameters can be seen in Table 2, and the error rates obtained for the 2D and 5D data are recorded in Table 3. In addition to training the network on the raw data, it was attempted to reduce the dimensionality of the 5D data using PCA, as well as increasing the dimensionality using the kernel methods mentioned in the previous section. However, the optimal performance was obtained when training on the raw data.

Data	Hidden Layer					Learning Rate	Decay Rate
	1	2	3	4	5		
2D	82	55	144	43	7	1e-5	1e-4
5D	116	110	97	n/a	n/a	n/a	2e-5

Table 2. MLP Parameters

Results: The classification results obtained from KLDA and MLP can be seen in Table 3. The performance obtained by both classifiers is comparable. Both algorithms took a long time to run. For KLDA, several operations resulted in a long run time. First, computation of the kernel matrices for the *training*, *dev*, and *eval* data, took a long time to compute, in addition to a total of 4.89 gb of memory. Second, when training LDA, the subspace which maximizes the Fischer's criteria is solve by finding the eigenvectors of $S_W^{-1}S_B$, where S_W and S_B are the within and between class scatter matrices respectively. Computing these eigenvectors using SVD requires a long time, given the dimension of these matrices. With regards to MLP, this algorithm resulted in a long run time mostly because of the extensive search for optimal parameters.

	2D Data Set			5D Data Set		
Algorithm	Train	Dev Test	Eval	Train	Dev Test	Eval
KLDA	09.24%	09.45%	09.50%	38.04%	37.63%	38.13%
MLP	08.54%	09.25%	08.90%	39.99%	39.75%	40.45%

Table 3. Error Rate Comparison of KLDA and MLP for 2D and 5D Data

Conclusions: When comparing KLDA and MLP, based on the results obtained in this project, it cannot be determined if one algorithm is superior over the other. While KLDA gave slightly better results for the 5D data, and MLP gave slightly better results for the 2D data, both algorithms took equally long time to run. It is possible that is all data points were used to construct the kernel matrices, a lower error rate would have been achieved, but at the cost of computation time. It is also possible the MLP would have resulted in a better error rate if a better approach was used to optimize the systems parameters.