Final Project: Learning Machine Classifiers

MAE 277 Learning Control Systems

Alex Nguyen

Department of Mechanical and Aerospace Engineering
University of California Irvine

Instructor: Thanasis Sideris
June 14, 2021

Contents

1 Introduction

This project's goal was to compare the performance of a variety of Learning Machines applied on readily available benchmark data (UC Irvine Machine Learning Repository [1]) with one another. The data set chosen for training, validation, and testing was the wine data set where two MATLAB toolboxes (i.e., Statistics and Machine Learning and Deep Learning Toolboxes) were utilized for classification and analysis. The data was a result of a chemical analysis of wines grown in the same region in Italy, but derived from three different cultivars. There are 13 different wine attributes associated with the three wine classes. Thus, our input data set is multivariate with $x \in \mathbb{R}^{13}$ and our output data set is multi-class with $y \in \mathbb{R}^3$. So, our problem boils down to a multi-classification problem (3 or More Classes) rather than a binary classification (2-Classes) problem like we have regularly seen in MAE 277. For more information regarding the wine data set, see the "UC Irvine ML Repository" website for details.

Figure 1: Wine data set, using chemical analysis to determine the origin of wines.

2 Machine Learning Algorithms

Although this project required no training or testing data to be generated, there was the question of how to best classify the various wine attributes into their respective classes. My project will compare the performance of four different Learning Machines classifiers: Feed-Forward Neural Network, K-Nearest Neighbors, Multi-Class Support Vector Machines, and K-Means clustering with Principal Component Analysis. The following subsections give a brief high-level overview of these different Learning Machines and their implementation with the help of MATLAB's available toolboxes.

2.1 Feed-Forward Neural Network (FFNN)

The FFNN algorithm is the simplest type of artificial neural network devised. The information moves forward from the input layer nodes, through the hidden layer nodes, and to the output layer nodes. In a layer, each neuron receives inputs from the neurons of the previous layer and form the outputs by applying an activation function $\theta(\cdot)$ to a linear combination of the input neurons. The Learning Algorithm is the method of Steepest Descent (SD) to minimize R_{emp} over the Neural Network (NN) weights and biases. This Learning Machine was implemented with the help of MATLAB's "Deep Learning Toolbox" to perform classification on the wine data set. Mathworks has an online example for a FFNN using pattern recognition on the same data set, which aided in the classification and analysis computations for this section.

Our problem setting differed from Mathwork's, as the FFNN parameters were changed. Here, we utilized a two-layer FFNN (i.e., one hidden layer) as this problem can be classified sufficiently with a single hidden layer. The FFNN had eight neurons in the hidden layer, implemented gradient descent with momentum as the training function, and had the mean-square error (MSE) as the performance function. The number of hidden layer neurons were chosen applying a rule of thumb stating “the hidden layer neuron # = mean of the number of inputs and number of outputs.” The MATLAB pattern recognition function `patternnet` preps the network to be trained with the samples being automatically divided up into training, validation, and testing data sets. The training/validation set is used to teach and tune the FFNN whereas the test set is used to provided an independent measure of FFNN's accuracy. The MATLAB NN training tool `nntraintool` trained our network and provided all necessary performance data to compare our FFNN with other classifiers.

2.2 K-Nearest Neighbors (K-NN)

The K-NN algorithm is a non-parametric classification algorithm used for both classification and regression where the inputs consists of the k closest training examples in the data set. The common distance metric used between the k "neighbors" is Euclidean distance. The K-NN approach is based on variable volume regions containing a fixed amount of k points to our test point x and assigning the class with the most samples among the k samples. Thus, K-NN requires no learning but its computational effort scales to $\mathbb{O}(Nd + N\log(k))$ in exchange. This Learning Machine was implemented with the help of MATLAB's "Statistics and Machine Learning Toolbox" to perform the classification of the wine data set. The Mathworks documentation of K-NN supplied enough information to apply the toolbox's functions to our problem.

Our problem utilized the K-NN classifier with the wine's input attributes as the samples for classification. The number of k fixed points was chosen based on a rule of thumb recommending it be around the square root of the number of inputs divided by two (i.e., $k = \lceil \frac{1}{2}\sqrt{N_{\text{train}}} \rceil$). Now, we are ready to "train" our K-NN classifier with `fitcknn` function. After which, we can pass our testing data to the prediction function `predict` to evaluate our classifier's performance. Also, other MATLAB functions for evaluating performance during training and cross-validation which are discussed in the results section below.

2.3 Multi-Class Support Vector Machines (SVMs)

The SVM algorithm is a supervised learning model used for both classification and regression analysis. Typically, SVMs are used for binary classification but an error-correcting output codes (ECOC) classifer model can be used for multi-class learning which breaks down the multi-classification problem into multiple binary classification problems. This multi-class to binary decomposition is known as a One-to-One approach. This Learning Machine was implemented with the help of MATLAB's "Statistics and Machine Learning Toolbox" to perform the classification of the wine data set. As before, the Mathworks documentation provided sufficient information to apply the toolbox functions to our problem.

Our problem utilized a multi-class ECOC SVM model to train our training data with MATLAB's `fitcecoc` function. After standardizing the input data, this fit the multi-class wine data set by using the One-to-One approach uses $\frac{m(m-1)}{2}$ binary SVMs for classification. The data points will be classified into m data points. After which, we use the prediction function `predict` on our testing data to evaluate our classification performance. As before, other MATLAB functions are used to evaluate performance during training and cross-validation which are discussed in the results section below.

2.4 K-Means Clustering with Principal Component Analysis (PCA)

The PCA algorithm is commonly used for dimensionality reduction by projecting each data point onto the first few principal components. The goal is to obtain a lower-dimensional data set while preserving the much of the data's variance as possible. The principal components are the (orthonormal) eigenvectors of $\mathcal{U}$ matrix corresponding to the p largest eigenvalues of $\mathcal{S}$ where our normalized data matrix $\mathcal{Z}\mathcal{U}\mathcal{S}\mathcal{V}^\top$ by a SVD. The K-means clustering algorithm is a method of vector quantization to partition the input data set's N observations into K

clusters defined by their respective centroids. K must be chosen at the start, but we already know the number of clusters we wish to find from the lower-dimensional wine data set. The k-means clustering minimizes squared Euclidean distances. The K-means clustering and PCA algorithms were implemented with the help of MATLAB's "Statistics and Machine Learning Toolbox" to perform the classification of the wine data set. Again, the provided Mathworks documentation was sufficient to apply the toolbox functions to our problem.

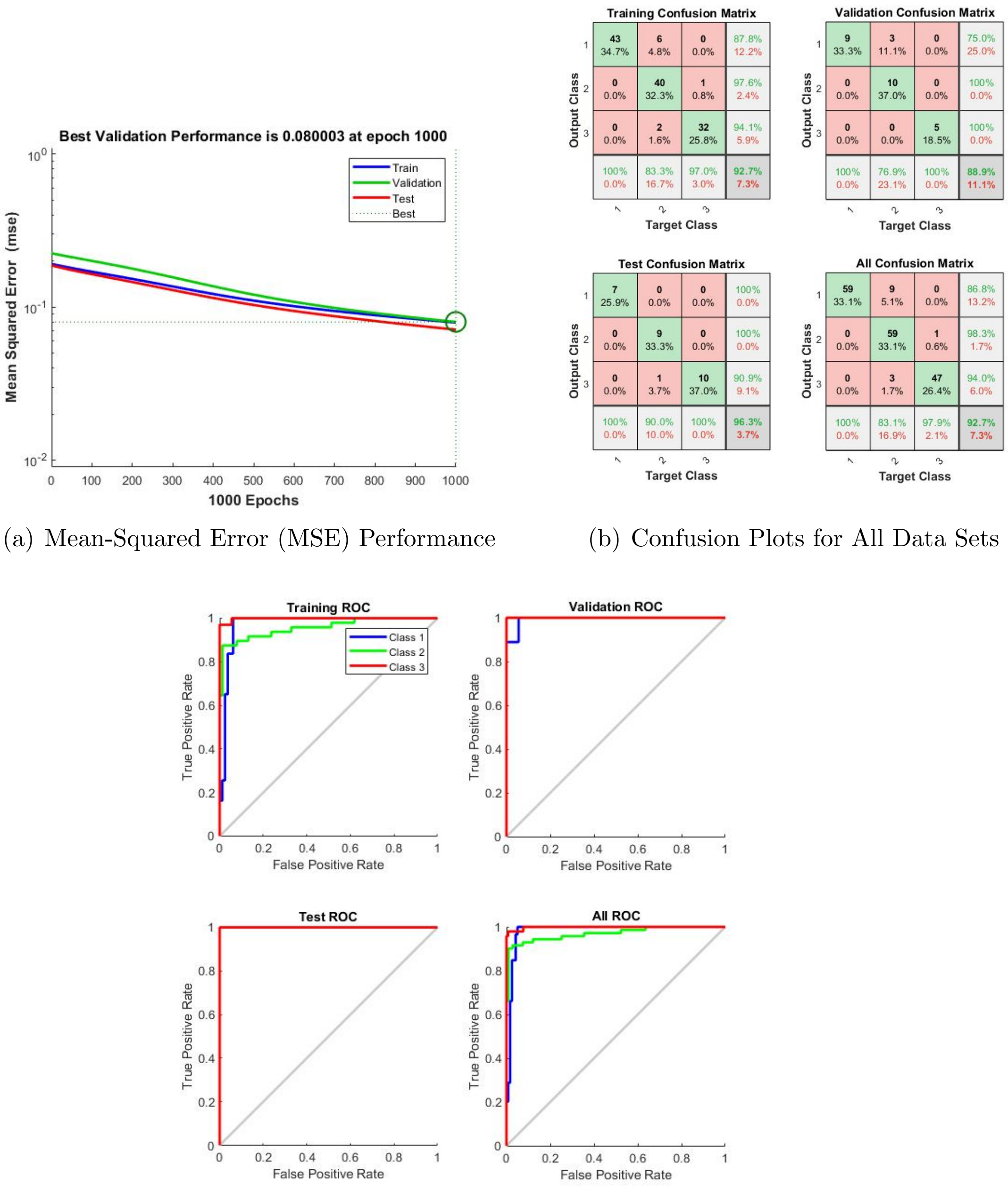
Our problem first called for a dimensionality reduction using PCA. This allowed us to easily visualize the input data as $x \in \mathbb{R}^2$ rather than $x \in \mathbb{R}^{13}$, as we chose to look at the first two principal components. After performing PCA, we are ready for classification of the wine data set using the K-means clustering algorithm. It is worth noting, in cluster analysis there is not usually a training/test data split so we perform the k-means clustering algorithm on our entire input data set using the `kmeans` function in MATLAB. Here, obviously $K = 3$ clusters to represent the three wine classes. This function returns the classified clusters with their respective centroids from our lower-dimensional input data.

3 Results

The simulation results aimed at comparing different learning machines with one another. In order to make this comparison fair, each classifier's simulation had its the random seed set to the same random number generator seed to reproduce the same performance metrics with each simulation run. Furthermore, test data classification results and resubstituion/cross-validation loss were presented as well. The resubsitution loss signifies the classifier's prediction inaccuracy on the training data set, whereas the cross-validation loss signifies the average loss of each cross-validation model when predicting on data not used for training.

3.1 FFNN Performance

The FFNN algorithm's performance results are shown in Figure 2. Figure 2(a) shows the mean-square error (MSE) for the training and testing data sets. The best validation performance was found to be 0.080003 at the 1000$^{\text{th}}$ epoch. Figure 2(b) shows the confusion plots for the training, validation, and testing data sets. We can see the overall confusion matrix for the combined data set had a classification of accuracy of 92.7 % correct and 7.3 % incorrect. The best classification came from the testing data while the worse came from the validation data. Figure 2(c) shows the receiver operating characteristics which is a metric used to check the quality of classifiers. The best classifiers require fewer false positives to be accepted to get a high true positive rate.



Training Confusion Matrix

	1	2	3	
1	43 34.7%	6 4.8%	0 0.0%	87.8% 12.2%
2	0 0.0%	40 32.3%	1 0.8%	97.6% 2.4%
3	0 0.0%	2 1.6%	32 25.8%	94.1% 5.9%
	100% 0.0%	83.3% 16.7%	97.0% 3.0%	92.7% 7.3%

Validation Confusion Matrix

	1	2	3	
1	9 33.3%	3 11.1%	0 0.0%	75.0% 25.0%
2	0 0.0%	10 37.0%	0 0.0%	100% 0.0%
3	0 0.0%	0 0.0%	5 18.5%	100% 0.0%
	100% 0.0%	76.9% 23.1%	100% 0.0%	88.9% 11.1%

Test Confusion Matrix

	1	2	3	
1	7 25.9%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	9 33.3%	0 0.0%	100% 0.0%
3	0 0.0%	1 3.7%	10 37.0%	90.9% 9.1%
	100% 0.0%	90.0% 10.0%	100% 0.0%	96.3% 3.7%

All Confusion Matrix

	1	2	3	
1	59 33.1%	9 5.1%	0 0.0%	86.8% 13.2%
2	0 0.0%	59 33.1%	1 0.6%	98.3% 1.7%
3	0 0.0%	3 1.7%	47 26.4%	94.0% 6.0%
	100% 0.0%	83.1% 16.9%	97.9% 2.1%	92.7% 7.3%

(a) Mean-Squared Error (MSE) Performance

(b) Confusion Plots for All Data Sets

(c) Receiver Operating Characteristics

Figure 2: Performance results for a two-layer Feed-Forward Neural Network (FFNN) with 8 neurons

3.2 K-NN Performance

The K-NN's algorithm performance results are shown below, which are printed verbatim from the MATLAB command window. Here, we see a table showing the prediction results from a random subset of the testing data. Also, this study included the resubstitution loss to see the training inaccuracy classification as well as the error obtained by a 10-fold cross-validation model.

```
K-Nearest Neighbors (KNN)
Random Subset of Test Data:
    True Labels    Predicted Labels
    -----------    ----------------

         1                1
         2                2
         1                1
         2                2
         2                2
         2                2
         3                3
         1                1
         2                2
         1                1

Test Data:
 Correct Classification 96.2963 %
 Incorrect Classification 3.7037 %

Resubstitution Loss:
Classifier Predicts Incorrectly for 4.8387 % of the Training Data

Cross-Validated Loss:
Generalized Classification Error 5.6452 % of the Training Data
```

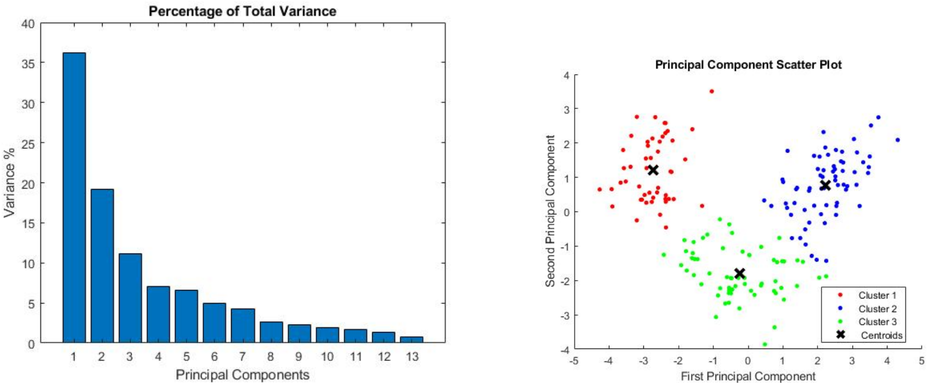
3.3 Multi-Class SVMs

The Multi-Class SVMs algorithm's performance are shown below which are printed verbatim from the MATLAB command window. Here, we see a table showing the prediction results from a random subset of the testing data. Also, this study included the resubstitution loss to see the training inaccuracy classification as well as the error obtained by a 10-fold cross-validated model.

```
Multi-Class Support Vector Machines (SVMs)
Random Subset of Test Data:
    True Labels    Predicted Labels
    -----------    ----------------

    {’Class 1’}      {’Class 1’}
    {’Class 2’}      {’Class 2’}
    {’Class 1’}      {’Class 2’}
    {’Class 3’}      {’Class 2’}
    {’Class 1’}      {’Class 1’}
    {’Class 3’}      {’Class 3’}
    {’Class 2’}      {’Class 2’}
    {’Class 1’}      {’Class 1’}
    {’Class 2’}      {’Class 2’}
    {’Class 1’}      {’Class 1’}

Test Data:
 Correct Classification 88.6792 %
 Incorrect Classification 11.3208 %

Resubstitution Loss:
Classifier Predicts Incorrectly for 0.5618 % of the Training Data

Cross-Validated:
Generalized Classification Error is 4.4944 % of the Training Data
```

3.4 K-Means Clustering with PCA

The K-means clustering with PCA algorithm's performance is shown in Figure 3. Figure 3(a) shows the percentage of total variance from each principal component in our input data set. This allows us to visualize our data's variance to help decide which principal components should form the basis of our lower-dimensional data set. Here, we see that the first two variance components account for a much of our input data's the variability so we decide $p = 2$. Figure 3(b) shows the classified clusters with their respective centroids marked as "x". The dimensionality reduction allowed us to visualize our data set in a 2D scatter plot with our two largest principal components as the lower-dimensional data. This is extremely useful because it is much easier to visualize a 2D data set than 3D+ data sets (e.g., 13D).



(a) Percentage of Total Variance for Each Principal Component

(b) K-Means Clustering with Centroids

Figure 3: Performance results for a K-means clustering with Principal Component Analysis (PCA)

Also, the K-means clustering with PCA classification results are shown below which are printed verbatim from the MATLAB command window. Here, we see a table showing the clustering results for a random subset of the wine data set. The classification accuracy is pretty good, especially considering we reduced the data's dimension by 11!

```
K-Means Clustering with PCA
13 Principle Components reduced to 2 Principle Components (p < d)

The First 2 Principal Components Account for 55.4063 % of the Variance

Random Subset of Cluster Data:
    True Labels    Predicted Labels
    -----------    ----------------
```

```
         3                3
         1                1
         3                3
         1                1
         3                3
         1                1
         2                1
         1                1
         1                1
         2                2

Cluster Data:
 Correct Classification 94.9438 %
 Incorrect Classification 5.0562 %
```

3.5 Performance Comparison

An overall performance comparison between the different Learning Machines using the wine data set is summarized below in Table 1. This table shows a comparison of classification on test/cluster data after the model was trained. We can see that the FFNN performed the best, but each algorithm classifies the test data fairly accurately and similarly. This is important as we can choose a Learning Machine which requires less computational effort to product approximately the same classification accuracy results as more computationally intensive algorithms.

Table 1: Learning Machines Testing Data Performance Comparison

Learning Machine	**Correct Classification** (%)	**Incorrect Classification** (%)
FFNN	96.2963	3.7037
K-NN	94.4444	5.5556
Multi-Class SVMs	88.6792	11.3208
K-Means Clustering with PCA	94.9438	5.0562

4 Conclusion

This study found that each Learning Machine classified the wine data set somewhat similarly and accurately. The ranking of classification accuracy went FFNN, K-Means Clustering with PCA, K-NN, and Multi-Class SVM. This is could be due to a smaller observation data set length, but this informs us that any of these classification algorithms can sufficiently classify the wine data set after training. For me, the most useful algorithm was the K-mean clustering using PCA for dimensionality reduction since it allowed us to accurately classify and visualize the high dimensional input data in a lower dimension based on the principal components. But nonetheless, Learning Machine allowed me to obtain a deeper understanding of the respective algorithms in practice. In the future, these classification algorithms could be implemented in my research for aiding in selection of ambient terrestrial Signals of Opportunity (SOPs) to improve localization and mapping estimation performance.

References

[1] D. Dua and C. Graff, “UCI machine learning repository,” 2017. [Online]. Available: http://archive.ics.uci.edu/ml