ECE 544 Final Report

Yuhui Lai[1]

***Abstract*— In this paper, I will make comparison between three algorithms, namely linear discriminant analysis, neural network and binary decision tree in terms of accuracy and computation complexity using a two class datasets which is associated with the BUPA liver disorders. This paper begins with a brief introduction of the dataset used, through the theories of all three different algorithms, ends with quantitative analysis for the experiment results. In the discussion section, I will explain the reasoning of choosing these three distinctive learning algorithms for this particular dataset. Also, the efficiency and advantages in using each algorithm for this data set will be presented. Overall, for nonlinear non-sparable dataset, the binary decision tree gives us the best accuracy for both training and testing dataset. Neural network provides us similar performance compared to binary decision tree algorithm regarding to testing dataset. In terms of time complexity, binary decision tree consumes the least computation power among the three. The time complexity for neural network is highly dependent on the structure of the neural net.**

I. Introduction

In this part, I will first provide you a brief introductory of the training data set and testing data set. Then, a brief discussion of each three algorithms will be presented.

A. two class datasets

In order to compare the performance of three distinctive learning algorithms, I used dataset from BUPA liver disorders from BUPA Medical Research Ltd. The number of instances or samples is 345, and the number of feature dimension also known as attributes is 7. The first 5 variables are all blood tests which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption. Each line in the bupa.data file constitutes the record of a single male individual. It appears that drinks>5 is some sort of a selector on this database. The following is the Attribute information:

- 1. mcv mean corpuscular volume
- 2. alkphos alkaline phosphotase
- 3. sgpt alamine aminotransferase
- 4. sgot aspartate aminotransferase
- 5. gammagt gamma-glutamyl transpeptidase
- 6. drinks number of half-pint equivalents of alcoholic beverages drunk per day
- 7. selector field used to split data into two sets

The training data consists of the first 300 samples for the bupa datasets, and the rest 45 samples will be taken as testing dataset. There are totally 6 dimensions of feature in this case and the class label will be the 7th dimension. A figure elaborates the first and second dimensions for training dataset is shown in figure 1.

A figure demonstrates the first and second dimensions for testing dataset is shown in figure 2.

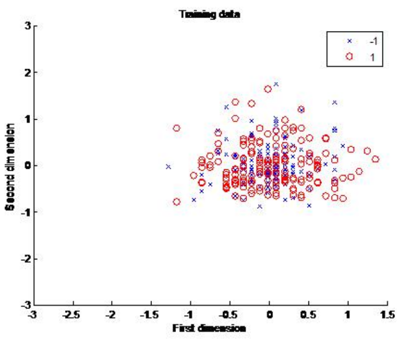


Fig. 1. First and second dimension plot of training dataset

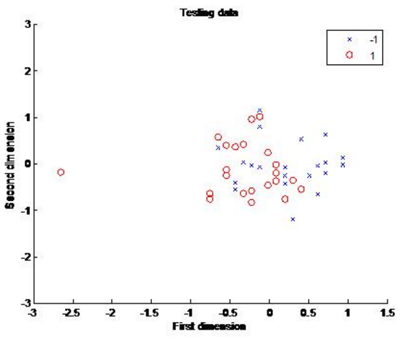


Fig. 2. First and second dimension plot of testing dataset

After visualizing all 6 dimensions, I conclude that the two class datasets are neither linear nor separable in all cases which makes our selection of algorithms of this particular dataset interesting. I choose LDA, neural network and binary decision tree to model the training dataset and utilize the training model to classify the testing dataset. In theory, LDA should probably fail to classify the testing tokens given the nature of the training data is nonlinear and non-separable at all. The performance of neural network and binary decision tree will be much better than LDA given they are able to handle nonlinear and non-separable training data. An overview of all three different algorithms will be presented in the following section.

II. Linear Discriminant Analysis

There are many possible techniques to do classification for training dataset. Linear Discriminant Analysis (LDA) is a commonly used measure for data classification and dimensionality reduction. Linear Discriminant Analysis can easily handle the case that the within-class frequencies is different. LDA maximizes the ratio of between-class covariance matrix to the within-class variance for my particular dataset in order to best separate data from different class. In other words, LDA projects the training dataset onto a lower-dimensional vector space known as affin subspace such that the ratio of the between-class distance to the within-class distance is maximized. The computation of optimal projection (transformation) is achieved by eigen decomposition on the scatter matrices of the training dataset.

Given a training dataset $A \in IR^{N\times n}$, LDA maps each feature dimension which corresponding to a vector a_i where $1 \leq i \leq n$ in the N-dimensional space to a vector b_i in the L-dimensional space. The purpose of LDA is to find the transformation matrix H that the class structure of the original high-dimensional space which is N in this case is preserved in the low-dimensional l subspace.

In theory, if each class is linearly separable from the other classes, the performance of the clustering should be high. In LDA, we have to find the within class and between class scatter matrices for each class as follow

$$S_w = \sum_{i=1}^{k} \sum_{x \in \prod_i} (x - m_i)(x - m_i)^T \tag{1}$$

$$S_b = \sum_{i=1}^{k} n_i (m_i - m)(m_i - m)^T \tag{2}$$

$$m_i = \frac{1}{n_i} \sum_{x \in \prod_i} x \tag{3}$$

$$m = \sum_{i=1}^{k} \sum_{x \in \prod_i} x \tag{4}$$

where m_i is the mean of the ith class, and m is the global mean. The trace(S_w) measures the closeness of the vectors within the class and trace(S_b) measures the separation between the classes. After the projection onto low-dimensional subspace, and given the transformation matrix H, the within class and between class matrices become

$$S_b^L = H^T S_b H \tag{5}$$

$$S_w^L = H^T S_w H \tag{6}$$

In this case, H should be achieved such that trace(S_w) will be minimized and trace(S_b) will be maximized. This is

$$max(H) trace((S_w^L)^{-1} S_b^L) \tag{7}$$

$$min(H) trace((S_b^L)^{-1} S_w^L) \tag{8}$$

In order to minimize the within class scatter matrices and minimize the between class matrices, we have to solve the following eigenvalue problem:

$$S_b x = \varepsilon S_w x \tag{9}$$

$$x^{-1} S_w^{-1} S_b x = \varepsilon \tag{10}$$

The solution of above equation can be obtained through eigen decomposition to the matrix $S_w^{-1} S_b$ for nonsingular S_w Sw or nonsingular S_b. There are at most $k-1$ eigenvectors corresponding to nonzero eigenvalues where k denotes the number of classes, since the rank of the matrix S_b is bounded from above by $k-1$. Therefore, the reduction in dimension by LDA is at most $k-1$.

In terms of time complexity, LDA has $O(mnt + t^3)$ computation complexity and requires $O(mn + mt + nt)$ memory, where m is the number of samples, n is the number of features and $t = min(m,n)$.

III. Neural Network

The purpose of neural network is trying to achieving human-like performance in the fields of speech and image recognition. Training a neural net composes of many nonlinear computational elements operating in parallel and arranged in patterns reminiscent of biological neural nets. Computational elements or nodes are connected via weights that are typically iterated using gradient descent during the learning process to enhance performance.

In theory, given enough nodes, a 3-layer network with sigmoid or linear activation functions can approximate any functions with bounded support sufficiently accurately. In our experiment, we will be measuring the accuracy with increasing hidden nodes and hidden layers. The general architecture of neural net is shown in figure 3.

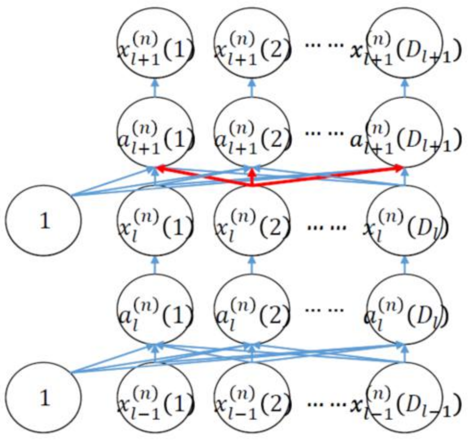


Fig. 3. General structure of Neural net

The goal of training a neural net is to achieve the minimum loss function, which is defined as follow

$$L(y^{(1:N)}, t^{(1:N)}) = \sum_{n=1}^{N} l(y^{(n)} - t^{(n)}) \tag{11}$$

where N is the number of hidden nodes for the neural net and y^n is the output of neural net which gives the label for input x^n. t^n represents the correct label for input x^n.

The calculation for training a neural net is following:

Bottom layer

$$a_1^{(n)} = b_1 + W_1 x_0^{(n)} \tag{12}$$
$$x_1^{(n)} = g_1(a_1^{(n)}) \tag{13}$$

Middle layer

$$a_{l+1}^{(n)} = b_{l+1} + W_{l+1} x_0^{(n)} \tag{14}$$
$$x_{l+1}^{(n)} = g_{l+1}(a_{l+1}^{(n)}) \tag{15}$$

Top layer

$$a_L^{(n)} = b_l + w_l x_{l-1}^{(n)} \tag{16}$$
$$y^{(n)} = g_L(a_L^{(n)}) \tag{17}$$

The goal is find the optimal set of W_l, b_l which minimize the loss function. In order to obtained the optimal set of W_l, b_l, we have to adjust the weights for each hidden layer accordingly. The error back propagation algorithm will help us to achieve that goal. Basic idea of error back propagation is to compute the gradient for each hidden node at each hidden layer. The gradient descent will be performed after we calculated gradient. There are four steps in implementing error back propagation algorithm:

Step 1:Forward propagation

$$a_{l+1}^n = b_{l+1} + W_{l+1} x_0^n \tag{18}$$
$$x_{l+1}^n = g_{l+1}(a_{l+1}^n) \tag{19}$$
$$a_L^n = b_l + w_l x_{l-1}^n \tag{20}$$
$$y^n = g_L(a_L^n) \tag{21}$$
$$L(y^{1:N}, t^{1:N}) \tag{22}$$

Step 2:Error back propagation

$$\delta_L^n = g_L^{'}(a_L^n) \cdot \nabla_{y^n} l^n \tag{23}$$
$$\delta_l = g_l^{'}(a_l^n) \times [W_{l+1}^T \delta_{l+1}] \tag{24}$$

Step 3: Gradient calculation

$$\nabla_{W_l} L = \sum \delta_l^n x_{l-1}^{(n)T} \tag{25}$$
$$\nabla_{b_l} L = \sum \delta_l^n \tag{26}$$

Step 4: Gradient descent

$$W_l \leftarrow W_l + \eta \nabla_{W_l} L \tag{27}$$
$$b_l \leftarrow b_l + \eta \nabla_{b_l} L \tag{28}$$

The time complexity of each error back propagation is $(weights \times tokens)$.

IV. Binary Decision Tree

In the field of classification, binary decision tree is a commonly used technique to solve difficult decision problems where there are many classes and many available features related in a complex manner. We seek to create a model with the objective of attaining a high average prediction power which is expressed by means of average prediction entropy of the binary tree leaf distributions. An example of binary tree of level is shown in 4.

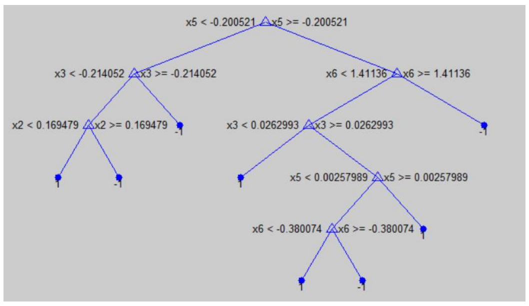


Fig. 4. Level 3 binary decision tree for training data

The entropy for a distribution for a given set X at a leaf l as

$$H_l = -\sum_{s_i \in X} P_l(s_i) log_2 P_l(s_i) \tag{29}$$

The average binary tree prediction entropy is

$$\overline{H} = \sum_l P_l H_l \tag{30}$$

P_l is the prior probability of visiting the leaf l, and $P_l(s)$ is the probability of observing a symbol s at that leaf. The goal is to minimize the average entropy given the training dataset. The optimum tree structure and the corresponding node questions is solved by applying a greedy search algorithm at each node, combined with a recursive procedure for creating tree nodes. The steps of building the tree architect is as follows:

Step 1: Initialize the current node c as the root of the tree.

Step 2: For each predictor variable $X_i (i = 1 : N)$ find the subset S_i^c which minimizes the average conditional entropy of the symbol distribution Y at node c

Step 3: Determine which of the N questions derived in Step 2 leads to the lowest entropy. For example, if this is question k then k is:

$$k = argmin \overline{H_c}(Y | X_i \in S_i^c) \tag{31}$$

Step 4: The reduction in entropy at node c due to question k is

$$R_c(k) = H_c(Y) - \overline{H_c}(Y | X_k \in S_k^c) \tag{32}$$

where

$$H_c(Y) = -\sum_{s_j \in A} P(s_j | c) log_2 P(s_j | c) \tag{33}$$

If this reduction is significant, store question k, create two descendant nodes, $c1$ and $c2$, pass the data corresponding to the conditions $X_k \in S_k^c$ and repeat Steps 2-4 for each of the new nodes separately. The time complexity of binary decision algorithm is $O(m^2 \times n)$ where n is the number of samples and m is the number of feature dimensions

V. Results

The general results for all three different algorithms are shown below. In figure 5, the green straight line represents the between class linear classifier which separates group 1 and group -1.

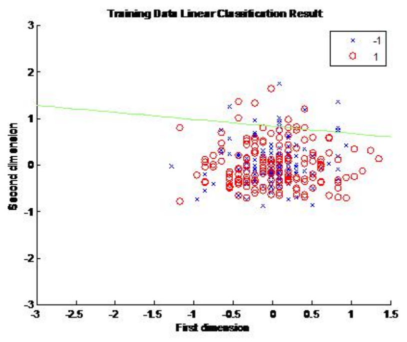


Fig. 5. LDA classifier for training data. Only first and second dimension are plotted in this case.

In figure 6, the results of misclassified token in training dataset are illustrated. Also, the linear classifier given the training dataset for testing dataset is demonstrated in figure 7

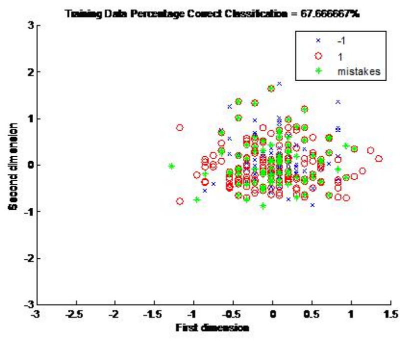


Fig. 6. LDA classifier for training data with only first and second dimension plotted in this case. The green dot represents the misclassified token for both class 1 and class -1

The result of neural net is shown in figure 8 and figure 9.

With regard to binary decision tree, the result is demonstrated in figure 10 and figure 11.

For neural network, the number of hidden nodes and number of hidden layers will affect the accuracy in classifying

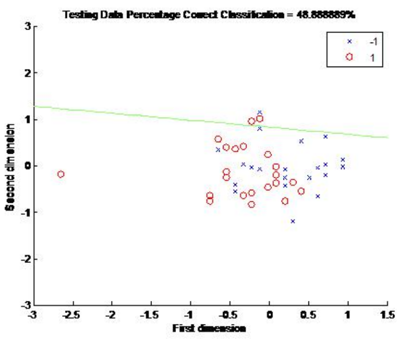


Fig. 7. LDA classifier for testing data with only first and second dimension plotted.

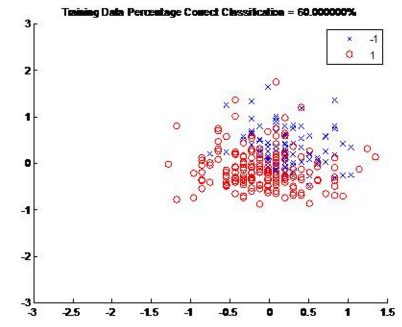


Fig. 8. NN classifier for training data with only first and second dimension plotted. 5 hidden nodes and 1 hidden layer is deployed.

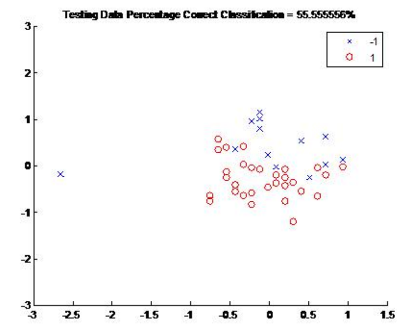


Fig. 9. NN classifier for testing data with only first and second dimension plotted. 5 hidden nodes and 1 hidden layer is deployed.

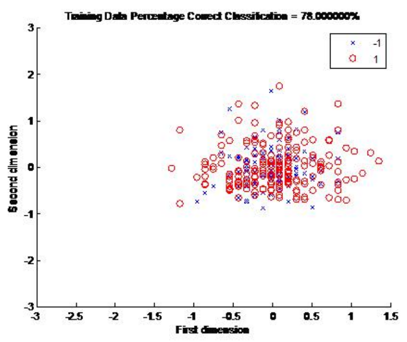


Fig. 10. BT classifier for training data with only first and second dimension plotted. A binary tree of level=3 is used.

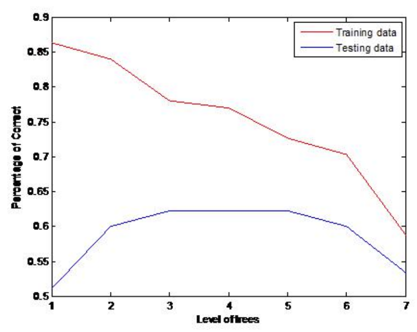


Fig. 11. BT classifier for training and testing data vs tree levels. The x axis represents the plumb in levels for trees. For example, a level=1 is defined as a binary tree level=8 being plumbed into level=7. On the other hand, a level=7 is equal to a binary tree level=8 plumbed into level=1.

the training and testing tokens, the tables with increasing hidden nodes and hidden layer for training and testing data are shown below:

In general, the accuracy in learning the training dataset for LDA is 67 percent, NN is 78 percent and BT is 84 percent given optimum level of tree. However, for the testing token, the best model trained by these three different algorithms only provides us around 60 percent accuracy for classification. The detailed analysis for each algorithm is in the following section.

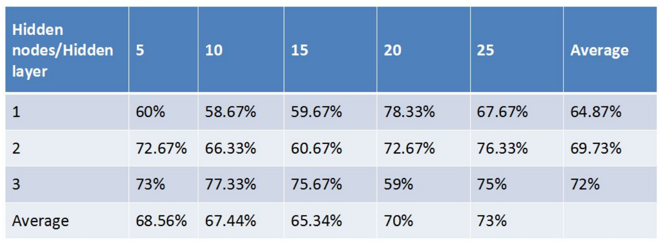
VI. Discussion and Conclusion

The overall comparison between these three algorithms is presented below:

where m is the number of samples and n is the number of feature dimensions.

For LDA, given the nature of the training and testing dataset is nonlinear non-separable, we expect the model trained by the training token will approximately give us around 50 percent accuracy for testing tokens. Even though

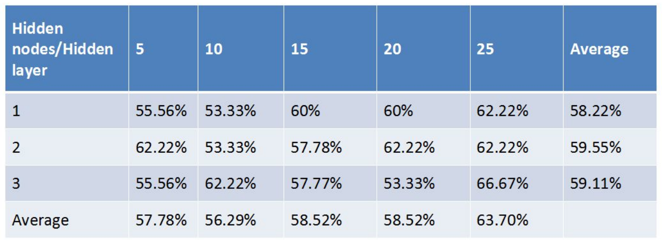
Training data



Hidden nodes/Hidden layer	5	10	15	20	25	Average
1	60%	58.67%	59.67%	78.33%	67.67%	64.87%
2	72.67%	66.33%	60.67%	72.67%	76.33%	69.73%
3	73%	77.33%	75.67%	59%	75%	72%
Average	68.56%	67.44%	65.34%	70%	73%	

Fig. 12. NN classifier for training data vs number of hidden nodes and number of hidden layers.

Testing data

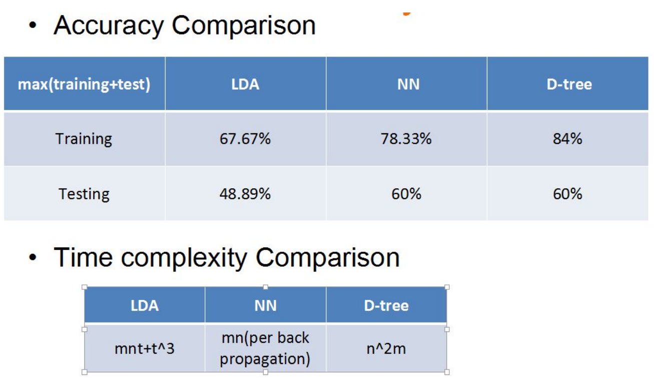


Hidden nodes/Hidden layer	5	10	15	20	25	Average
1	55.56%	53.33%	60%	60%	62.22%	58.22%
2	62.22%	53.33%	57.78%	62.22%	62.22%	59.55%
3	55.56%	62.22%	57.77%	53.33%	66.67%	59.11%
Average	57.78%	56.29%	58.52%	58.52%	63.70%	

Fig. 13. NN classifier for testing data vs number of hidden nodes and number of hidden layers.

we maximize the between class scatter matrices and minimize the within class scatter matrices, the classifier would still fail the correctly identify the testing data. Since classifier use to classify the testing data is similar to toss a coin and as a result we should get around 50 percent accuracy because the tokens from two classes are normally distributed under the same area.

Regrading to neural network, we notice that with increasing number of hidden nodes, the accuracy for correctly classify the training tokens increases. Also, adding extra hidden layers also enhance the accuracy in classification. However, there are a few exceptions in our case, for example, when the number of hidden layer is 3, and we increase the number of hidden nodes from 15 to 20, the percent of



- Accuracy Comparison

max(training+test)	LDA	NN	D-tree
Training	67.67%	78.33%	84%
Testing	48.89%	60%	60%

- Time complexity Comparison

LDA	NN	D-tree
mnt+t^3	mn(per back propagation)	n^2m

Fig. 14. A comparison between three classifiers for training and testing data. The max means that the sum of accuracy for training and testing is the best in all cases for that particular classifier.

accuracy dropped from 75 percent to 59 percent. I believe this is due to the over-fitting problem which enlarge the error rate on the contrary. Generally, with layer increased from 1 to 3, the average accuracy increased by 8 percent. With hidden nodes increased from 5 to 25, the percent accuracy increased by 4.5 percent.

Binary decision tree performs the best among all three algorithms. The deeper the tree, the better accuracy will be given for training dataset. However, it doesn't necessarily provides us the best accuracy for testing dataset. As we can see from figure 11, a tree with level 7 gives us 84 percent accuracy in classifying the training data, but it only has a 52 percent rate in correctly recognizing the testing data. In other words, over-fitting the training dataset with a deep binary tree may not give you the wanted result for testing token, since a shallower tree may classify the testing dataset in higher level. Plumbing is the key factor we should consider when dealing with binary decision tree.

VII. Reference

Navratil, Jiri, et al. "Phonetic speaker recognition using maximum-likelihood binary-decision tree models." Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on. Vol. 4. IEEE, 2003.

Ye, Jieping, Ravi Janardan, and Qi Li. "Two-dimensional linear discriminant analysis." Advances in neural information processing systems. 2004.

Cai, Deng, Xiaofei He, and Jiawei Han. "Training linear discriminant analysis in linear time." Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on. IEEE, 2008.