DIABETES CLASSIFICATION AND PREDICTION USING ARTIFICIAL NEURAL NETWORK

Kshitij Tripathi
The Maharaja Sayajirao University of Baroda, Vadodara, India

ABSTRACT

The classification of data is an important field of data mining comes under supervised learning. In this approach classifier is trained on the pre-categorized data thereafter tested on unseen part called test data to evaluate it. The other related field clustering comes under unsupervised learning is used for categorizing data into different clusters or assigning labels to them which are previously unknown. In this article the classification of data is done and we are using artificial neural networks (ANN) for pre-processing i.e. removing noisy instances through novel clustering technique and then classifying pre-processed data through ANN. Both are exhaustive approaches. The data set used in this article is PIMA Indian diabetes data set available on UCI repository.

Key words: Classification, Artificial Neural Network, Machine Learning, Cross Validation, Clustering, K-fold TVT.

Cite this Article: Kshitij Tripathi, Diabetes Classification and Prediction Using Artificial Neural Network, International Journal of Computer Engineering and Technology 10(3), 2019, pp. 154-165.

http://www.iaeme.com/IJCET/issues.asp?JType=IJCET&VType=10&IType=3

1. INTRODUCTION

The diabetes is the most dreaded disease in the present time and is gradually increasing all over the world at a high-speed. This disease decreases the energy level of patient and turns out to be a cause of many other diseases like cardiac problem is one of them. Not only old but young people and children have also come under the diagnosis of this either less or more. According to International Diabetes Federation report, from 2012 to 2015, approximately 1.5 to 5.0 million deaths each year resulted from diabetes. In the current article we select the Pima Indian diabetes dataset (PIMA) [7], for diagnosing patients, containing multiple attributes (symptoms) contributing to the classification of patients into two categories. The ANN's [11] are known as universal approximator as they have proved themselves for solving a wide variety of problems which are function approximation (for mathematical functions or formulas), prediction (machine learning), curve–fitting, classification (machine learning) and clustering (unsupervised learning) are few of them. Here we are using ANN for classifying diabetes based on PIMA data set.
2. RELATED WORK

There is a lot of studies already performed and available in past literature on PIMA by different researchers who employed various techniques of data mining like Support Vector Machines (SVM), K-nearest neighbour, Decision Trees, Bayesian classifier, Artificial neural networks and various variants of regression. The main task is to correctly classify the diabetic and non-diabetic patient on the basis of 8 attributes or symptoms which can be numerical measured or recorded. Patil et al. (2010) uses HPM (Hybrid predictive model) and C4.5 for the same. Choubey et al. (2016) presented a technique based on GA (Genetic algorithm) as attribute selection and Naïve bayes (NB) for the same. Alexis Marcino et al. (2011) uses artificial meta-plasticity neural network for classification. Han wu et al. (2018) employs k-means clustering and logistic regression to pre-process and classify the data. Ahmed et al. (2011) uses MLP (multilayer perceptron) for the same.

3. DATASET

PIMA Indian diabetes dataset is available on UCI repository and is one of the most popular among medical domain for the researchers in the field of data mining. There are 768 instances or samples of females who are at least 21 years old. Following are the 9 attributes (Numerical values):

1. Number of times pregnant
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)
9. Class variable (0 or 1)

4. DATA PRE-PROCESSING

The quality of data-set is also one of the significant factors in achieving better classification accuracy. If we remove some instances or maybe some attributes then the results of classification are significantly improved. Further, there is some noise in the data set which when treated properly also helps in obtaining the better results. Some of these are:

(a) Irrelevant features like those whose values as 0 or 1 are sufficient to analyse instead of other numerical values (like zero (0) is sufficient for absent and 1 is sufficient for the present (instead of 2, 3 or 4 or more).

(b) Missing values like zeroes (0), which are due to various reasons like non-availability or not to be truthful.

(c) Some attributes which when removed from data-set effects the result of classification significantly. The reason is subset of features is large enough to classify data accurately.

(d) Some values which are much higher or much lower comparatively than other values of same attribute also alter the results as these are called outliers.

It is seen in case of PIMA data-set that old researchers do not use any pre-processing technique for classification on the PIMA dataset. But in the recent past most researchers use data pre-processing techniques before applying any classification algorithm for analysis to
improve the efficiency of classification. The reason is given dataset is imbalanced due to the above mention reasons.

4.1. Steps performed for cleaning and pre-processing of data.
Step1: Zeros present in the data set with a mean of the corresponding attributes.
Step2: Unsupervised normalise filter is used to prepare the data.
Step3: Finally we have removed the instances which are incorrectly classified through novel clustering algorithm based on ANN.

After performing step 3 we remove 218 instances and finally settled on 550 instances. The proposed clustering algorithm is discussed in section 7. All ANN related work like clustering and classification is done in Matlab. Data pre-processing (step 1 and 2 above) before clustering is done in Weka [12]. The experiments are performed without removing any attribute or feature.

5. THE ARTIFICIAL NEURAL NETWORKS
The artificial neural networks (figure 1) [9][11] are inspired from human brain in which neuron is its basic unit. Biological neurons are input-output processing elements and similar is the case with artificial neuron. Another similarity is that both biological neurons and artificial neurons are interconnected structures. The artificial neural networks are layered architectures which consist of multiple layers as seen in figure. In each ANN there are at-least three layers called as input layer, hidden layer and output layer. But it is seen that ANNs having more than one hidden layer is giving better results than a single hidden layer network in large size datasets having greater number of attributes.

![The basic architecture of ANN](image)

**Figure 1** The basic architecture of ANN

5.1. Working of ANN
The artificial neural networks are trained on input-output vector pairs. The most widely used algorithm for training ANN is the back-propagation algorithm [5] which dominates the entire ANN domain. The working of the back-propagation algorithm is as follows:
Step1. First, each edge layer is assigned the weights according to a randomized technique or function.
Step2. Inputs are applied to the input layer. Step3. The weighted sum of the inputs is determined for each neuron.
   
   \[ w_1x_1+w_2x_2+\ldots+w_nx_n \]  
   where \( w_i \) = weights of corresponding edges \( x_i \) = corresponding inputs.
Step4. The weighted sum acts as an input for the transfer function.
Step 5. The output of each neuron is calculated up to the output layer by applying the transfer function.

Step 6. The error is determined at the output layer which is the difference between computed output and given output. The weights are then readjusted to minimize the error according to the back-propagation training algorithm.

The step 2 to step 6 above have been iterated with all instances of given data set containing input-output vector pairs and each iteration is called an epoch. Training is stopped when the network gives the optimized performance. The stopping criteria for training are:

- The maximum number of epochs is reached.
- The minimum gradient is reached.
- Best validation performance is achieved.
- The goal is reached (mean square error is minimized or zero).

The main derivations [11] applied in the back-propagation algorithm is:

\[ y_{ik} = \sum_j z_j w_{jk} + b \]

Where \( y_{ik} \) is the net input to \( k^{th} \) output neuron and \( z_j \) is the input to \( k^{th} \) neuron and \( w_{jk} \) is the weight associated with it.

\[ y_k = f(y_{ik}) \]

\( y_k \) is the output of \( k^{th} \) output neuron, \( f \) is the transfer function applied to the weighted sum of inputs and weights assigned to edges and \( b \) is the bias associated with the particular neuron.

The error function to be minimized is

\[ E = \frac{1}{2} \sum (t_k - y_k)^2 \]

Here, \( t_k \) is the target output and \( y_k \) is the computed output. The weight update equations are:

For hidden to output layer weights:

\[ w_{hi}^{k+1} = w_{hi}^k + \Delta w_{hi}^k \]

\[ = w_{hi}^k + \eta \left( - \frac{\partial E_k}{\partial w_{hi}^k} \right) \]

\[ = w_{hi}^k + \eta \delta_h^k \theta_i^k \]

For input to hidden layer weights:

\[ w_{ih}^{k+1} = w_{ih}^k + \Delta w_{ih}^k \]

\[ = w_{ih}^k + \eta \left( - \frac{\partial E_k}{\partial w_{ih}^k} \right) \]

\[ = w_{ih}^k + \eta \delta_h^k \theta_i^k \]

Note: \( \eta \) is the learning rate and \( \delta_h^k \) represents an error and signal slope product that is error scaled by the signal slope.

Table 1 describe the details of ANN used in experiments performed in this article.
6. CLASSIFICATION THROUGH K-FOLD TVT APPROACH

It is observed that ANN’s are sometimes over-fitted. Meaning is they reflect good results in training phase but not as good in the test phase. To avoid over-fitting of data, before testing, the algorithm is validated on the 'validation' set. The literal 'validation' is taken in two contexts. In the first context, it is used in the technique of K-fold cross-validation where the data is divided into ‘K’ folds most likely 10 or maybe 15. In this technique, each time during testing, each and every fold one by one can be treated as the test set and remaining folds are kept for training. The mean of all the ‘K’ results (as the case may be 10 or 15) are calculated for getting final results. This technique is called K-fold cross validation. The second context in which word ‘validation’ is used in which all data-set is divided into three parts called as training, validation [8] and test sets.

Table 1 K-fold TVT approach

| Folds       | Experiment 1 | Experiment 2 | Experiment 3 | ... | Experiment 14 |
|-------------|--------------|--------------|--------------|-----|---------------|
| Fold 1      | Training     | Training     | Training     | ... | Validation    |
| Fold 2      | Training     | Training     | Training     | ... | Training      |
| Fold 3      | Training     | Training     | Training     | ... | Training      |
| ...         | Training     | Training     | Training     | ... | Training      |
| Fold 13     | Training     | Validation   | Training     | ... | Training      |
| Fold 14     | Validation   | Training     | Training     | ... | Training      |
| Fold 15     | Test         | Test         | Test         | ... | Test          |

First, the learning is performed on training set till the optimized state is reached. After that, the algorithm is validated (tested) on validation set until best validation performance is achieved on the validation set. Finally, it is tested on the test set. Generally, the ratio of the three sets is 70:15:15. In this article the experiments are performed doing 15-fold cross-validation. The reason is PIMA has an unbalanced class ratio. So for serving maximum exposure to both classes, we are doing 15 fold cross-validation and each time 13 folds are used for training and 14th and 15th fold are used respectively for validation and testing in each experiment and each set is acting as test set. It is also revealed from K-fold TVT [9](table 1) approach that the solution of the problem of over-fitting in ANN is hidden in the data itself i.e. we are discovering most friendly validation set to the test set to avoid over-fitting. Further, it is also known that initialization of weights is an important factor which contributes to the accuracy of ANN at the end so we do exhaustive weight initialization.

Table 2 Configuration parameters of ANN

| Parameter                              | Value     |
|----------------------------------------|-----------|
| Total Number of layers                 | 5         |
| Number of hidden layers                | 3         |
| Number of neurons in each hidden layer | 10        |
| Training algorithm                     | Trainscg  |
| Transfer function in first hidden layer| Tansig    |
| Transfer function in second hidden layer| Tansig   |
| Transfer function in Third hidden layer| Tansig   |
| Transfer function in output layer      | Softmax   |
| Performance function                   | Mean square error |

That is why it is called exhaustive approach and is very efficient in obtaining optimized ANN as it is all a matter of weights with which ANN is configured and giving output on behalf of input.
7. PROPOSED CLUSTERING ALGORITHM

Classification and clustering (Fig 2) are two main tasks employed in the field of machine learning and the main objective is to build a learning model or algorithm which will adapt the nature or characteristics of instances or more specifically the complete dataset for establishing the relationship between the data instances which may be multivariate or multi-attribute. The obtained learned model must have a capability for performing the task of identifying the class of one or more unknown instances in case of classification and finding the optimal number of clusters present in the dataset as well as grouping the instances into these optimal numbers of clusters where the dataset has no predefined class labels. In both cases the main task is to train or build the model which will adapt the characteristics of given dataset. The steps of proposed clustering algorithm are as follows:

Step1: Find the appropriate (optimum) configuration of ANN for the given dataset like number of neurons, number of hidden layers, transfer function in given layers and the performance function etc.

Step2: As the class labels are not available initially in case of clustering so we hypothetically convert classification algorithm into two class dataset. The first reason for this is that most classification algorithms like for example 'trainrp' in matlab expects before training that the dataset at-least contains two classes. Second reason for this is that every classification or may be clustering problem must have at-least two classes otherwise classification is not needed. So, before going into step3 of proposed clustering algorithm we convert and assign two class vectors for representing two classes like \( \begin{bmatrix} 1 \\ 0 \end{bmatrix} \) and \( \begin{bmatrix} 0 \\ 1 \end{bmatrix} \). For example if dataset contains 100 instances than assign class vector \( \begin{bmatrix} 1 \\ 0 \end{bmatrix} \) to instances from 1 to 99 and assign \( \begin{bmatrix} 0 \\ 1 \end{bmatrix} \) to last instance so that algorithm gets worked. It may also \( \begin{bmatrix} 1 \\ 0 \end{bmatrix} \) and \( \begin{bmatrix} 0 \\ 1 \end{bmatrix} \) as we proceed if required.

Step3: Run the proposed algorithm as an normal classification algorithm and save the results. (As the proposed approach is hierarchical so the algorithm runs more than one time for finding the optimal number of clusters). Employ performance function for evaluation and threshold based on the respective situation (as per dataset).

Step4: Finally we obtain the clusters based on the groups of indices according to threshold value.

Figure 2. Clustering

It has been seen that if two or more instances or say two indices (we can use it interchangeably) belong to same cluster than there is a great intimacy between their validation performance and test performance. In short we run the algorithm with above criteria as shown below.

Pseudocode of critical part of the algorithm
Kshitij Tripathi

(i). Training indices= [All indices except any two indices]
(ii). Validation indices= [Any one of the two indices left]
(iii). Test indices= [single left over indice]

Each time we train the network, initial configuration must be same like random number seed, number of hidden layers, number of neurons in hidden layer and training algorithm (we have opted 'trainrp' as training algorithm and the transfer function 'tansig' present in Matlab having one hidden layer and 'softmax' as transfer function on output layer).

**Proposed Clustering algorithm**

1. Assign index to each record of dataset D
   (for example D={i1,i2,i3,i4,...........in})
2. C=1
3. X=1:n
4. If C==1
   {  
     Set testindex=1 & S=D  
   }
   Else
   {  
     If D==All C’s
     Exit  
     S=S-{All C’s}
     Set testindex=first index of S
   }
5. For j=S2 to Sn
   valindex=j
   Trainingindices=S-{testindex,valindex}
   r=(abs(sum(abs(abs(tr.vperf)-abs(tr.tperf))))
   CL[C]=ε  
   if 0>r<1  
       CL[C]={CL[C]}+{valindex}
   End
   C=C+1
6. Repeat steps 3 to 5

From the graphs (Figure 3 (a-f)) obtained on the experiments performed it is revealed that those indices which belong to same cluster have very little or almost no difference between validation performance and test performance and those indices which belong to different clusters have considerable difference between validation performance and test performance. Now the final task is to discover the threshold function and its value which will reflect the boundaries between each cluster and also the total (optimal) number of clusters.
Following is the formula used for calculating the difference value (threshold according to table 3): \( (\text{abs}(\text{sum}(\text{abs}(\text{abs}(\text{tr.vperf})-\text{abs}(\text{tr.tperf})))) \)

where \( \text{abs} = \text{absolute value}, \text{sum} = \text{total}, \text{tr.vperf} = \text{validation performance}, \text{tr.tperf} = \text{test performance} \).

On the basis of above criteria (Table 3) we have removed noisy instances which have given the predetermined class label but not actually fitted in their given class. Further the important part of this clustering algorithm is it gives 100% results of clustering accuracy on linearly separable datasets which may be multivariate in nature and can also determine the number of clusters. We start above algorithm after initializing indices of first instance as test set and iteratively changing validation instances till all remaining indices are validated. Now we obtain all the validation indices which are nearest to indices of test set and this set of indices is first cluster. Afterwards we again set any new indices as test instance which is compliment of first cluster. Again follow the steps discussed as before and we will discover all indices which are nearest to test indices and it is our second cluster and so on. It may happen for some datasets when threshold value varies from the value given in table 2. In PIMA dataset as the classes (class vector also) are already given so we just repeatedly taken one indices from each class (cluster) and apply above clustering algorithm and eliminate instances according to criteria given in table 3. That is one class (cluster) has assigned vector \( \begin{bmatrix} 1 \\ 0 \end{bmatrix} \) and other has assigned \( \begin{bmatrix} 0 \\ 1 \end{bmatrix} \).

### Table 3 Criteria of threshold

| State | Difference between validation indices performance and test indices performance |
|-------|--------------------------------------------------------------------------------|
| If nature of indices are similar (i.e. belong to same class) | 0 (Approx.) |
| If nature of indices are dissimilar (i.e. belong to different class) | >=1.00 |
8. RESULTS AND STATISTICS

Table 4 Confusion matrix obtained after pre-processing

| A   | B   | CLASSIFIED |
|-----|-----|------------|
| 507 | 0   | Test Negative |
| 21  | 22  | Test Positive |

Table 5 Confusion matrix obtained without pre-processing

| A   | B   | CLASSIFIED |
|-----|-----|------------|
| 461 | 47  | Test Negative |
| 74  | 186 | Test Positive |

Table 6 Statistics obtained after pre-processing

|                |             |          |
|----------------|-------------|----------|
| Precision      | 1           |          |
| Recall         | 0.511       |          |

Table 7 Statistics obtained without pre-processing

|                |             |          |
|----------------|-------------|----------|
| Precision      | 0.7982      |          |
| Recall         | 0.7153      |          |

General Format of confusion matrix

| Actual class | Class 1 | Not Class 1 |
|--------------|---------|-------------|
| Predicted class | True positive(TP) | False positive(FP) |
|                | False negative(FN) | True negative(TN) |

P=Number of real positive cases in the data.
N=Number of real negative cases in the data.

\[
\text{Accuracy} = \frac{TP + TN}{TOTAL}
\]

\[
\text{Misclassification rate} = \frac{FP + FN}{TOTAL}
\]

\[
\text{Recall} = \frac{TP}{TP + TN}
\]

\[
\text{Precision} = \frac{TP}{TP + FP}
\]
Diabetes Classification and Prediction Using Artificial Neural Network

Table 8 Results obtained without pre-processing and reduction

| Method          | Accuracy  |
|-----------------|-----------|
| Our Proposed model | **84.24%** |
| LogDisc         | 78.20%    |
| Discrim         | 77.50%    |
| SMO             | 77.00%    |
| SMART           | 76.80%    |
| J48             | 76.70%    |
| LMT             | 76.60%    |
| SGD             | 76.60%    |
| Random Forest   | 76.00%    |
| RBF             | 75.70%    |
| Backprop        | 75.20%    |
| NaiveBay        | 74.90%    |
| BayesNet        | 74.70%    |
| MLP             | 73.80%    |

Classification Accuracy = \( \frac{\text{Number of correctly classified instances}}{\text{Total instances}} \) \times 100

Classification Error = \( \frac{\text{Number of misclassified instances}}{\text{Total instances}} \) \times 100

Table 9 Results obtained after pre-processing and reduction

| Method          | Reference       | #Samples | #Attributes | Description                          | Accuracy  |
|-----------------|-----------------|----------|-------------|--------------------------------------|-----------|
| Our proposed approach | This Paper      | 550      | 8           | With preprocessing and cross validation | **96.21%** |
| Logreg          | H.Wo et al      | 589      | 8           | With preprocessing and cross validation | 95.42%    |
| HPM             | B.M.Patil       | 433      | 8           | With preprocessing and cross validation | 92.38%    |
| AMMLP           | Alexis Marcono  | 308      | 8           | After Preprocessing without cross validation | 89.93%    |
| J48(Pruned)     | Aliza Ahmed     | --       | --          | --                                   | 89.93%    |
| J48(Unpruned)   | Aliza Ahmed     | --       | --          | --                                   | 89.30%    |
| Hybrid Model    | Humer Kahranali | --       | --          | --                                   | 84.50%    |
| MLP             | Aliza Ahmed     | --       | --          | --                                   | 81.90%    |
9. CONCLUSION

It is seen that K-fold TVT approach is giving good results on PIMA with and without preprocessing. Further, it is seen (table 8, figure 5) that the accuracy of results obtained without doing any pre-processing (i.e. with all 768 instances) is 84.24% which is very high then the results obtained by other researches[6] having same instances. The other statistics obtained are given in table 4 to table 8 also reflects good performance than other researches [6]. Also, accuracy obtained after pre-processing [6] (table 9, figure 6) is 96.21% which is higher than all other researches [6] which signifies that it is also a robust technique.

REFERENCES

[1] Alexis Marcano-Cedeño, Joaquín Torres, and Diego Andina, A Prediction Model to Diabetes Using Artificial Metaplasticity. IWINAC 2011, Part II, LNCS 6687, pp. 418–425, 2011.

[2] Aliza Ahmad and Aida MustaphaH, Comparison between Neural Networks against Decision Tree in Improving Prediction. Accuracy for Diabetes Mellitus. ICDIPC 2011, Part I, CCIS 188, pp. 537–545, 2011.

[3] Bankat M. Patil, Ramesh Chandra Joshi, Durga Toshniwal: Hybrid prediction model for Type-2 diabetic patients. Expert Syst. Appl. 37(12): 8102-8108(2010)

[4] Choubey, D.K., Paul, S., Kumar, S., Kumar, S., 2017. Classification of Pima indian diabetes dataset using naive bayes with genetic algorithm as an attribute selection, in:
Diabetes Classification and Prediction Using Artificial Neural Network

Communication and Computing Systems: Proceedings of the International Conference on Communication and Computing System (ICCCS 2016), pp. 451–455.

[5] D. E. Rumelhart, G.E. Hinton, R.J. Williams, "Learning internal representation by error propagation", Parallel distributed processing: Explorations in the microstructure of cognition, Vol.1, Bradford book, Cambridge, MA, 1986.

[6] Han Wu, Shengqi Yang, Zhangqin Huang, Jian He, Xiaoyi Wang," Type 2 diabetes mellitus prediction model based on data mining", Informatics in Medicine Unlocked V.10, 100–107, 2018.

[7] http://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes

[8] I.V. Tetko, D.J. Livingstone, A.I.Luik, "Neural network studies. 1. Comparison of overfitting and overtraining", Journal of chemical information and computer sciences V.35, 5, 826-833, 1995

[9] Kshitij Tripathi, Rajendra G. Vyas, Anil K. Gupta, “The Classification of Data: A Novel Artificial Neural Network (ANN) Approach through Exhaustive Validation and Weight Initialization”, International Journal of Computer Sciences and Engineering, V.6, 5, 241-254, 2018

[10] P.Jian, M. Kamber, "Data Mining: Concepts and Techniques", Morgan Kaufmann, 3rd Ed, 2012.

[11] Satish Kumar, "Neural Networks A Classroom Approach", Tata McGraw Hill, 2013.

[12] Weka at http: w.cs.waikato.ac.nz