A Comprehensive Exploration to the Machine Learning Techniques for Diabetes Identification

Sidong Wei[1], Xuejiao Zhao[2,3], Chunyan Miao[2,3]
[1]School of Mathematical Sciences, Shanghai Jiao Tong University, China
[2]Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY), NTU, Singapore
[3]School of Computer Science and Engineering, NTU, Singapore
Email:weisidong-g@sjtu.edu.cn {xjzhao, ASCYMiao}@ntu.edu.sg

***Abstract*—Diabetes mellitus, known as diabetes, is a group of metabolic disorders and has affected hundreds of millions of people. The detection of diabetes is of great importance, concerning its severe complications. There have been plenty of research studies about diabetes identification, many of which are based on the Pima Indian diabetes data set. It's a data set studying women in Pima Indian population started from 1965, where the onset rate for diabetes is comparatively high. Most of the research studies done before mainly focused on one or two particular complex technique to test the data, while a comprehensive research over many common techniques is missing.**

In this paper, we make a comprehensive exploration to the most popular techniques (e.g. DNN (Deep Neural Network), SVM (Support Vector Machine), etc.) used to identify diabetes and data preprocessing methods. Basically, we examine these techniques by the accuracy of cross-validation on the Pima Indian data set. We compare the accuracy of each classifier over several ways of data preprocessors and we modify the parameters to improve their accuracy. The best technique we find has 77.86% accuracy using 10-fold cross-validation. We also analyze the relevance between each feature with the classification result.

***Index Terms*—machine learning, deep neural network, classification, diabetes identification**

I. Introduction

Diabetes mellitus has a direct signal of high blood sugar, together with some symptoms including frequent urination, increased thirst, increased hunger and weight loss. Patient of diabetes usually need constant treatment, otherwise, it will possibly lead to many dangerous life-threatening complications. The diabetes is diagnosed with the 2-hour post-load plasma glucose being at least 200mg/dL [1], and the necessity of identifying diabetes timely calls in various studies about diabetes recognition.

Many previous research studies have been done about machine learning in diabetes identification. Research has been done focused on diabetes identification through GDA (Generalized Discriminant Analysis) and SVM (Support Vector Machine) [2] and they obtained some inspiring results. Another research was to do the same thing by GRNN (General Regression Neural Network) [3], which also had a very high accuracy.

Comparing to the previous work, we make a more comprehensive study containing a number of common techniques used to diabetes identification, intending to compare their performance and find the best one among them.

Through this experiment, we compare several common and data preprocessors for each of the classifiers we use, and find the best preprocessor respectively. Then we compare these classifiers after we modify the parameters of them to reach their approximate maximum accuracy, and we particularly analyse how to modify the parameters in DNN (Deep Neural Network). At last, we also analyze the relevance of each feature with the classification result, and this will help to modify the data set in future studies.

II. Related Work

The research about using machine learning technique to identify diabetes can be traced back to 1988 when J. W. Smith and his cooperators published a paper about using a so-called 'ADAP' algorithm to identify diabetes [4]. They use the Indian Pima data set of diabetes onset of women as their training and testing data, and the accuracy of their algorithm is about 76%. Though the result it made was not the best, it has inspired many researchers to apply machine learning technique to the identification of diseases like diabetes.

Many great results have been made using various algorithms. For example, Asha and her colleagues used a hybrid model of Genetic Algorithm and Back Propagation Network to identify diabetes [5]. They especially focused on adopting the algorithm on some particular input data and reached 84.7% on the identified inputs.

Kayaer's team used GRNN technique [3] to identify diabetes. They discussed how to build the network and had a similar result as Gail A. Carpenter and his group has made, which used a very complicated ARTMAP-IC network [6]. The technique Kayaer used was much simplified compared to Gail's, but it was still a complex one regard to the scale of the data set.

From all those researches we can see that they all explored diabetes identification through one particular method, and modified and improved it to its best or approximate best. The purpose of our research is to explore a bunch of common machine learning techniques for diabetes identification, and compare them comprehensively.

III. Methodology

The experiment we carry out has four steps in total, the first step is to find the best data preprocessor for each classifier we

choose, while the next step is to optimize the parameters of each classifier. The third step is to compare these technique of diabetes identification by their accuracy, and after that, we will consider the relevance of the features.

The procedure of the experiment is depicted in Fig. 1.

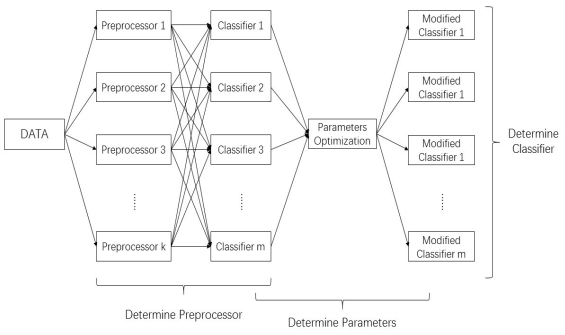

Fig. 1. Experiment Procedure

A. Data Preprocessors

There are two essential data preprocessors called PCA (Principal Component Analysis) [7] and LDA (Linear Discriminant Analysis) [8]. Both of them aim at decreasing the dimension of the feature set,but the approaches they applied are somehow different. In practical classification problems, there are usually hundreds of features to look into, most of which have little relevance to the classification, and that is where the PCA and LDA methods will work. On the country, if the number of features in a data set is already limited, these two methods may not help much.

Another problem we may face is missing data, which refers to the lack of information of some samples' some features. So one way of preprocessing data is to use the mean of all existing data to fill the missing ones, and this method is called 'Imputation'.

Besides, other preprocessors may also help to improve the performance of a classifier. One of them is called 'Scaling'. By scaling it means that we compute the mean and variance of each feature, then we adjust each data by minus the mean and then divided by the variance. So that the data after scaling will have a mean of 0 and a variance of 1. It is represented by mathematical symbols by:

$$x' = \frac{x - \frac{1}{n}\sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} x^2 - (\sum_{i=1}^{n} x)^2} \tag{1}$$

Where x_i in (1) represents the $i-th$ sample's feature x, and n is the number of samples.

Another preprocessor is called 'Normalizing', which refers to making each sample a vector with $k - components$, where k is the number of features and the value in the vector is its attributes. Then we divide the vector by its norm to make each vector to have unit norm. In this case, we use the $L^2 - norm$:

$$x(i)' = \frac{x(i)}{\sqrt{\sum_{i=1}^{k} x(i)^2}} \tag{2}$$

$x(i)$ means the $i-th$ component of vector x.

For Fig. 3 and TABLE I, noted that for Decision Tree and Naive Bayes perform exactly the same on the original data set and on the scaled data set. This result comes from the theoretic basis of these two methods. Decision Tree aims mainly on finding some thresholds for each feature after training. Thus when we scale the data, the threshold gets scaled too, and the result will remain the same.

As for Naive Bayes, the judgment calls from the conditional probability. That also means when you scale a feature for all data at the same ratio, the result won't change.

B. Classifiers

DNN is one of most popular and effective model in machine learning. In this experiment, all the layers we use in this model are dense layers, which means every pair of neurons in adjacent layers are connected [9]. The fundamental idea for DNN is to simulating the method that human learns new things through the neural network in human's brain. So the structure of the DNN is very similar to a simplified human's brain, and it's depicted in Fig.2.

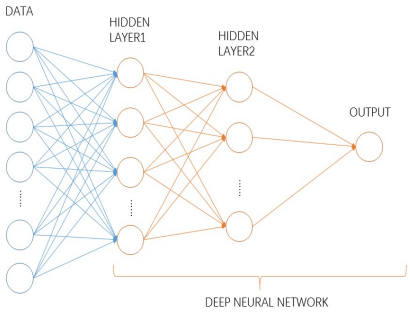

Fig. 2. Structure of DNN

When the data are sent into the neural network, they can be regarded as signals. For each layer the signal reaches, it is transformed through a function called activation function, and each node in the current layer will receive a different signal because they will use different linear combinations of the signal as input. When the signal arrives at the output layer, the algorithm will use the output to compute the error through loss function and use back propagation to adjust the activation parameter [10]. Actually, the back propagation can be regarded as another signal which is transmitted in the contrary direction, and this procedure includes quantities of complex mathematical computation. Finally, the error got sent back to the first layer, then the activation function is modified according to the error so that the error will get smaller, then the iteration starts again, and that's the way DNN works.

We also used some common techniques for classifying. Decision Tree is trying to identify each method through a sequence of comparison of some features with a specific order [11]. In doing that, it uses greedy decision tree learning to find some thresholds for a feature (also known as decision stump),

which can split the data with best classification accuracy. Then for each branch of the stump, it will do the same thing on another feature, and the algorithm will iterate the procedure until some branches have had pure classification or the features have been used up.

What SVM (Support Vector Machine) is trying to do is first to map the data set to a high dimensional space in which the data can be separated clearly. Then it try to find the best hyperplane to separate the data, and the criterion on 'best' is determined by the maximal of space between the two nearest data point on each side of the hyperplane [12].

Logistic Regression is a simplified case of DNN in some way, which has none hidden layer [13]. The data is processed with a particular activation function, the sigmoid function, and the output will be compared with 0.5 to classify the data. The classification result then will alter the coefficient of the linear combination of the input data, and the iteration goes on.

Moreover, we chose Naive Bayes as one method which is based on the well-known Bayes' rule [14]. The Bayes' rule can be written as:

$$P(y=c|x) = \frac{\prod_{i=1}^{k} p(x_i|y=c)p(y=c)}{\sum p(y=c)\prod_{i=1}^{k} p(x_i|y=c)} \tag{3}$$

Where x_i refers to the $i-th$ feature of sample x. Plus, c should be either 0 or 1, and as we know $p(y=1)+p(y=0)=1$, we don't need to care about the denominator. The classifier will use (3) to compute the possibility of y in the condition of given x and choose the one with higher possibility.

Those are the basic ideas of the classifiers we choose. Since cross-validation performs much better than splitting the data set into a training group and a testing group, we use cross-validation to examine the accuracy of each combination of classifier and preprocessors.

IV. Experiment

A. Data Set

In this paper, we also choose the data set of Pima Indian women about their diabetes recognition. It's a great data set to study due to the high onset rate in this area, which means the data here will reveal more typical features or symptoms of diabetes than other data set. This issue has been under long time study since 1965 by the National Institute of Diabetes and Digestive and Kidney Disease [4], with 8 particular features chosen to be studied. And the eight features are listed below:

- Number of Times Pregnant
- Plasma Glucose Concentration a 2 Hours in an Oral Glucose Tolerance Test
- Diastolic Blood Pressure (mm·Hg)
- Triceps Skin Fold Thickness (mm)
- 2-Hour Serum Insulin (mu U/ml)
- Body Mass Index (Weight in kg/(Height in m)^2)
- Diabetes Pedigree Function
- Age (years)

The Diabetes Pedigree Function (DPF) refers to the genetic factor about diabetes, which given in this equation:

$$DPF = \frac{\sum_i K_i(88-ADM_i)+20}{\sum_j K_j(ALC_j-14)+50} \tag{4}$$

In (4), 'i' ranges among the relatives who had developed diabetes while 'j' is the opposite. The coefficient K_x equals to the percentage of genes shared by $relative_x$, it can be 0.5, 0.25 and 0.125. Besides, 'ADM' represents the age of the patient when he was diagnosed with diabetes, and 'ALC' refers to the age when the last nondiabetic examination took place.

With the effort the previous researchers have done, the data set has already ruled out large numbers of irrelevant features. In other words, all these features are closely related to the onset of diabetes. So when we choose the preprocessors of this experiment, PCA and LDA have lost their advantages. As a result, we choose not to apply this two method on this data set.

B. Determine Preprocessor

After we have listed all the 5×6=30 combinations of preprocessors and classifiers, we compute each one with 10-fold cross-validation. Note that we use the same random seed for each test, such that the irrelevant variable of data splitting is controlled throughout the experiment. As we first need to determine which preprocessor is the best for each algorithm, the parameters of the algorithm do not really matter at this point. In this step, we just use the default parameters in every classifier. The output is shown in both Fig. 3 and TABLE I.

TABLE I
Accuracy of Techniques before Parameters Optimization

	Logistic Regression	DNN	SVC (SVM)	Decision Tree	Naive Bayes
Original	0.673189	0.661569	0.651059	0.711039	0.757861
Impute	0.690174	0.699248	0.651059	0.700581	0.747471
Scale	0.770899	0.751418	0.763072	0.711039	0.757861
Norm-alize	0.651059	0.691302	0.651059	0.611808	0.636637
Impute &Scale	0.768284	0.748667	0.753982	0.705793	0.747471
Impute &Norm	0.651059	0.703025	0.651059	0.678401	0.644429

Noticed that all the classifiers achieve the best accuracy after the data being preprocessed with 'Scale'. So we'll always use the scaled data in the rest of the experiment.

More precisely, Decision Tree and Naive Bayes perform exactly the same on the original data set and on the scaled data set. This result comes from the theoretic basis of these two methods. Decision Tree aims mainly on finding some thresholds for each feature after training. Thus when we scale the data, the threshold gets scaled too, and the result will remain the same.

As for Naive Bayes, the judgment calls from the conditional probability. That also means when you scale a feature for all data at the same ratio, the result won't change.

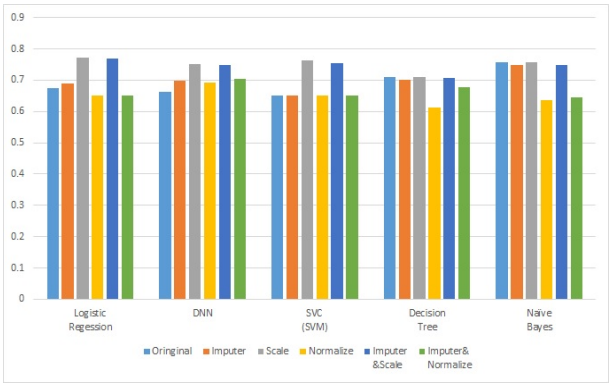

Fig. 3. Accuracy of Techniques before Parameters Optimization

C. Determine Parameters

The aim of this step is to optimize the parameters for each classifier, we'll omit most part of this procedure except for DNN, which has the biggest number of parameters to adjust and has the highest accuracy as proven later.

The numbers of neurons in the hidden layers are parameters of great importance, for they determine the structure of the neural network. Another effective parameter is the optimizer of the neural network, that is the way how the algorithm minimizing the value of the loss function. The choice of optimizer directly determines how fast and how well the algorithm fit to the best solution. Meanwhile, the activation needs to cooperate properly with the optimizer. Moreover, the times of epoch and batch size are also affecting classification accuracy. Epoch means the times that the whole data set is processed through the algorithm, and batch size means the number of data that are being processed at a time.

In order to obtain the best parameters, theoretically, we should test all the parameters at the same time, that's because the parameters have cross-interference. But practically, with the restriction of computing power, we have to omit the cross-interference factor and optimize small numbers of parameters at a time.

To start with, we try to optimize the numbers of neurons in both hidden layers. Since these two numbers are connected closely, we shouldn't consider them separately. So we consider the pair of numbers in the Cartesian product$\{8, 10, 12, 15, 20\} \times \{2, 4, 8\}$, and by comparing the accuracy of each pair, we get the approximately best pair $(15, 4)$. When we examine the following parameters, we will use this pair as default.

Then we optimize each other parameters respectively, with the order activation first, followed by the optimizer, and finally, the epoch and batch size. During this procedure, we carefully choose the best parameter every step and fix it in the following steps, trying to get to the best solution as close as we can.

The final best combination we get is that {activation: 'relu', optimizer: 'Nadam', epoch: 100, batch size: 10}.

D. Determine Classifier

As we have optimized each parameter in each classifier, the accuracy of each algorithm has increased slightly, shown in Fig. 4.

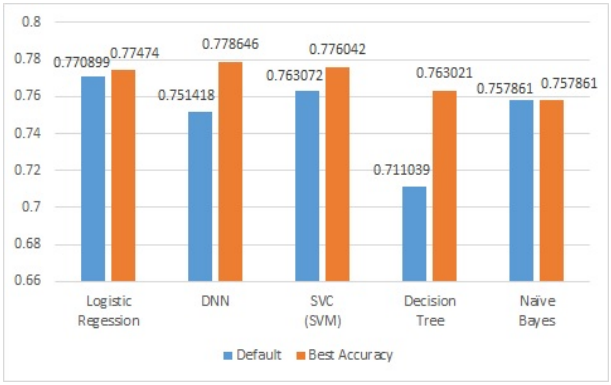

Fig. 4. Accuracy of Techniques after Parameters Optimization

It reads that after parameters optimization, DNN reaches the best accuracy among these five classifiers.

Then we want to understand how DNN achieve the accuracy over the times of epoch, so we plot the accuracy changes after each 10 times of epoch. The tendency line is graphed in Fig. 5.

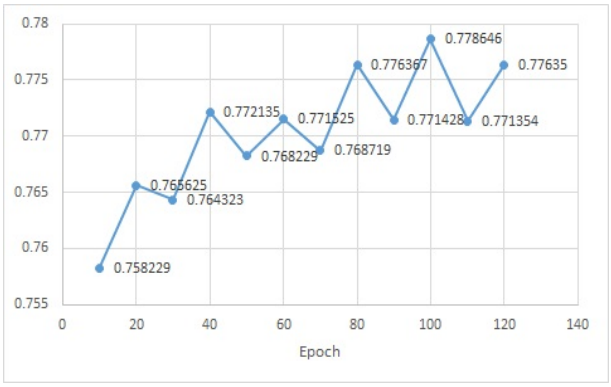

Fig. 5. Accuracy Tendency over Epoch

We can see from the graph that the accuracy does not necessarily increase with more epoch times. It is more likely to fluctuate within a small interval, and it reaches it maximal at epoch times equal to 100.

E. Features Effect

After we have identified the most accurate technique and learned how it reaches the best accuracy, we also want to further understand the importance of each feature. We use the best technique that we have found above, to test every feature separately, which means we use each feature as the single input to classify the samples in the data. The result we get is shown in Fig. 6. Notice that the serial number of the feature is in the same order as listed in IV-A.

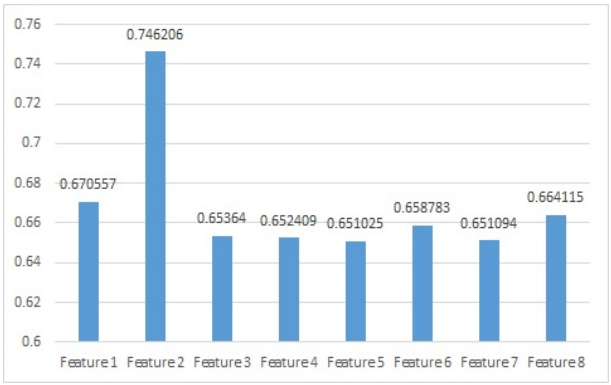

Fig. 6. Accuracy of Each Feature Individually

V. Conclusion

A. Best Accuracy and Features

Through the result in Fig. 4, we learned that DNN performs best. It's achieves the best accuracy of 77.86% in this comprehensive study. And it is a practical technique that can be used directly on any new data set, and the data should be preprocessed with 'Scale'.

According to the result in Fig. 6, the three most important features in this data set are: 'Plasma Glucose Concentration a 2 Hours in an Oral Glucose Tolerance Test', 'Number of Times Pregnant', and 'Age'. The least important feature is '2-Hour Serum Insulin'. But actually, all the features have similar importance except for the most important one, and they all have the classification accuracy above 65%. It verifies that the data set is chosen carefully, with all features relevant to the diagnose of diabetes.

B. Future Work

The most accurate classifier, DNN, however, still have potential to improve further. One way of doing that is to add the number of hidden layers, a couple more of layers can be added. But the cost of that may beyond the benefit it earns, so if we want to improve DNN with more hidden layers, some advanced tricks will get involved such as drop out layer, or more kinds of regularization terms.

VI. Acknowledgement

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IDM Futures Funding Initiative.

Though DNN is the best common technique in this study, it still has shortcomings like comparatively more computation time and more parameters to be adjusted. The choice of classifiers may also depend on other factors. For example, SVM and Decision Tree run much faster than DNN, with only a bit less accuracy, and Naive Bayes doesn't need any optimization on parameters unless we have information about prior probability. It makes sense to do further experiments on exploring techniques for diabetes identification by multi-factors including precision, recall, and efficiency rather than just accuracy.

References

[1] World Health Organization, "Report of a study group: Diabetes Mellitus," World Health Organization Technical Report Series, Geneva, 727, 1985.

[2] Kemal Polat, Salih Gunes, and Ahmet Arslan, "A cascade learning system for classification of diabetes disease: Generalized Discriminant Analysis and Least Square Support Vector Machine," Expert Systems with Applications, vol. 34. 1, January. 2008, pp. 482-487.

[3] Kayaer K and Yildirim T, "Medical diagnosis on Pima Indian diabetes using general regression neural networks," Proceedings of the international conference on artificial neural networks and neural information processing, 2003, pp. 181-184.

[4] Jack W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes, "Using the ADAP learning algorithm to forecast the onset of diabetes mellitus," Proc. Annu. Symp. Comput. Appl. Med. Care, November 9. 1988, pp. 261-265.

[5] Karegowda A. G., Manjunath A. S. and Jayaram M. A., "Application of genetic algorithm optimized neural network connection weights for medical diagnosis of pima Indians diabetes," International Journal on Soft Computing, vol. 2. 2, 2011, pp. 15-23.

[6] Carpenter G. A. and Markuzon N., "ARTMAP-IC and medical diagnosis: Instance counting and inconsistent cases," Neural Networks, vol. 11. 2, 1998, pp. 323-336.

[7] Wold S., Esbensen K. and Geladi P., "Principal component analysis," Chemometrics and intelligent laboratory systems, vol. 2. 1-3, 1987, pp. 37-52.

[8] Balakrishnama S. and Ganapathiraju A., "Linear discriminant analysis-a brief tutorial," Institute for Signal and information Processing, vol. 18, 1998.

[9] Deng L. and Yu D., "Deep learning: methods and applications," Foundations and Trends in Signal Processing, vol. 7. 3-4, 2014, pp. 197-387.

[10] Lee H., "Tutorial on deep learning and applications," NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning, 2010.

[11] Safavian S. R. and Landgrebe D., "A survey of decision tree classifier methodology," IEEE transactions on systems, man, and cybernetics, vol. 21. 3, 1991, pp. 660-674.

[12] Suykens J. A. K. and Vandewalle J., "Least squares support vector machine classifiers," Neural processing letters, vol. 9. 3, 1999, pp. 293-300.

[13] Hosmer Jr. D. W., Lemeshow S. and Sturdivant R. X., "Applied logistic regression," John Wiley & Sons, 2013.

[14] Lin Y., "Support vector machines and the Bayes rule in classification," Data Mining and Knowledge Discovery, vol. 6. 3, 2002, pp. 259-275.