Diabetes Mellitus Onset Prediction based on Neural Network

Weiqi Weng
College of Computer and Information Science
Northeastern University
weng.wei@husky.neu.edu

Yue Lyu
College of Computer and Information Science
Northeastern University
lyu.yue@husky.neu.edu

Abstract

To predict whether a patient has diabetes mellitus based on a variety of diagnostic measurements is a heated topic in medical informatics. hopefully doctors can first predict and then take some measurements for the patient to better control the disease and even get rid of it before being diagnosed. Neural Network comes naturally as a solution. It takes various curvature and interaction between features into account to form a complex hypothesis space. In this project, we are going to build a Back Propagation Neural Network after dimension reduction and do further prediction to evaluate model performance.

1. Introduction

In diabetes mellitus onset diagnosis, patients are advised to do a variety of medical tests in order to make convincing diagnostics and consequent measures. The process can be time-consuming and late for patients to better control and even get rid of it before being literally diagnosed. Given this point of view, works have been focused on diabetes mellitus prediction based on biological characteristics including weight, BMI and age, so as to take actions in advance and lower the risk of diabetes mellitus at best.

Relevant studies are boosted by different models. For example, a Logistic Regression model is used to predict the readmission of diabetic patients and discuss the impact of HbA1c measurement. (Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, and John N. Clore, 2014) Additionally, ADAP learning algorithm helps to improve the accuracy of diabetes mellitus onset forecasting. (Jack W. Smith, J.E. Everhart, W.C. Dickson, W.C. Knowler, and R.S. Johannes, 1988) Technically, if the prediction task is transformed into a classification problem, supervised learning methods such as logistic regression and perceptron can be helpful. Unlike these methods taking care of linear models in terms of weight or the coefficient of each explanatory variable, Neural Network automatically involves curvature and interaction between features through layer weights update. Of the very essence, Neural Network allows a diversity of architectures to expand hypothesis space and better simulate complex correlations within features and labels. Therefore, in this project, we aim to build a Neural Network model to predict based on biological characteristics.

2. Approach

2.1. Data Preprocessing

The first issue we encountered in data preprocessing is missing data imputation. The data set contains biological attributes such as blood pressure and glucose concentration which are impossible to be 0. An acceptable assumption would be that 0 stands for missing data. Specifically, there are 174 samples with 0 blood pressure, glucose concentration and insulin level. We can't afford to lose such significant information given a relative small data set so we fill missing data with a random value uniformly picked out from the mean of non-zero feature values within one standard deviation $[mean \pm std]$.

Next, the imbalance within response variable needs fixing. In the data set, only 34.89% of the samples have diabetes, which means a model predicting all samples to be non-diabetic will result in a 65.11% accuracy on the data set. We fix this problem with a combination of Synthetic Minority Over-sampling Technique (SMOTE) and Tomek Link, which is basically doing under-sampling and over-sampling at the same time. SMOTE plus Tomek Link is appropriate since it takes good control of the data set size. It increases the size to an proportional level. The balanced data set contains diabetic samples with a proportion of 49.7%.

With the balanced data set, we normalize numerical features into z-score, scaling out effect of feature values with overwhelming magnitude. Most normalized features are distributed roughly as a normal distribution. Finally the whole set is divided into training set, validation set and testing set with each part covering 80%, 10% and 10%.

2.2. PCA, Auto-encoder and Grid Search Cross Validation

Straightforward as the problem setup is, there is still possible redundant information in the features so two dimension reduction methods are introduced and compared.

First, Principal Component Analysis (PCA) is performed on the data set. For each possible number of component, we transform the data with PCA, feed the transformed data to train a Neural Network and run Grid Search Cross Validation to select the optimal hyper-parameters leading to the best accuracy. Specifically, for each possible number of component, we search hidden layer sizes from 5 to 21 and weight decay from 0 to 1 with a step 0.01. The framework is elaborated in the procedure PCA.

```
procedure PCA(data, feature size $N$, range of hidden layer size $S$, range of weight decay $D$)
Input: data frame, range of hidden layer size, range of weight decay
Output: the optimal hidden layer size $s_i^*$ and weight decay $d_i^*$ corresponding to principal component size $i$

    for all $1 \leq i \leq N$ do:
        for all $s \in S$ do:
            for all $d \in D$ do:
                train a neural network with $s$ and $d$
                do 5-fold cross validation on given data
                compute corresponding mean accuracy
                update best mean accuracy due to different $(s, d)$ combination
            end for
        end for
        record the best mean accuracy for $i$
    end for
return the best mean accuracy for $i = 1, 2, \cdots, N$

end procedure
```

Secondly, we use Auto-encoder to do dimension reduction. Similarly to PCA, for each middle layer size of Auto-encoder, we encode the given data as input data to train neural network. Then we run Grid Search Cross Validation to select the optimal hidden layer size and weight decay leading to the best accuracy. The framework is elaborated in the procedure Auto-encoder.

2.3. BP Neural Network

In this part, we build two Neural Networks Based on the optimal configurations returned by PCA and Auto-encoder. PCA suggests an architecture with Rectified Linear Unit activation and a specific set of weight decay, input layer size (corresponding to principal components) and hidden layer size. Auto-encoder returns an architecture with Sigmoid activation and a specific set of weight decay, encoding layer size and hidden layer size. Weights of each layer in each architecture are estimated with Forward and Back Propagation.

```
procedure AUTO-ENCODER(data, feature size $N$, range of hidden layer size $S$, range of weight decay $D$)
Input: data frame, range of weight decay
Output: the optimal weight decay $d_i^*$ corresponding to middle layer size of Auto-encoder $i$

    for all $1 \leq i \leq N$ do:
        train an Auto-encoder and transform the data
        for all $s \in S$ do:
            for all $d \in D$ do:
                do 5-fold cross validation on transformed data
                compute corresponding mean accuracy
                update best mean accuracy for $(s, d)$ combination
            end for
        end for
        record the best mean accuracy for $i$
    end for
return the best mean accuracy for $i = 1, 2, \cdots, N$

end procedure
```

2.4. Cross Validation and Testing

In order to figure out which architecture performs better for our problem in general, we run k-fold Cross Validation with $k = 5, 10, \cdots, 50$, compute mean accuracy, standard deviation and report error bar plot for each architecture on the validation set. Finally we train the better model on training-plus-validation set and run it on testing set to get the confusion matrix along with specificity and sensitivity.

3. Experiment and Result

3.1. Data Set Summary

The data set used in this project comes from National Institute of Diabetes and Digestive and Kidney Diseases. The organization has been studying and collecting data of Pima Indian female population near Phoenix, Arizona, because of the high rate of diabetes among this population. (Knowler, W. C., D. J. Pettitt, P. J. Savage, and P. J. Bennett, 1981) The data set has 8 numerical features as explanatory variables and 1 label as response variable. Details are included as follow. (Jack W. Smith, J.E. Everhart, W.C. Dickson, W.C. Knowler, and R.S. Johannes, 1988)

- Pregnancies: Number of times pregnant

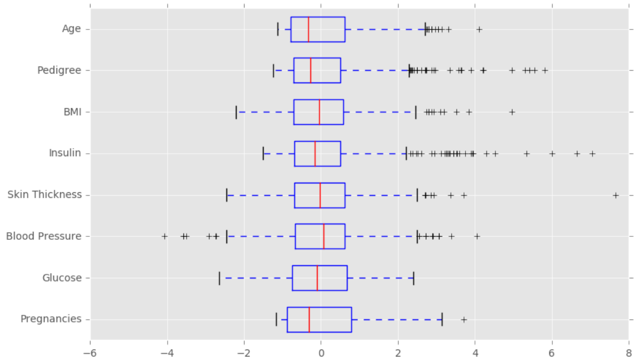

Figure 1. box plot of the features

- Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
- Blood Pressure: Diastolic blood pressure (mm Hg)
- Skin Thickness: Triceps skin fold thickness (mm)
- Insulin: 2-Hour serum insulin (mu U/ml)
- BMI: Body mass index (kg/m^2)
- Diabetes pedigree function: Providing a measure of expected genetic influence of affected and unaffected relatives on the subject's eventual diabetes risk
- Age: Age of the subject (years)
- Label: Class variable (0 for non-diabetic or 1 for diabetic)

After we fix imbalance, sample size grows from 768 to 956. Training set contains 764 samples while validation or testing set contains 101 samples respectively. After we fill in missing data and normalize data to z-score, we plot the box plot of each feature to see the rough distribution in the Figure 1. From Figure 1, we can see that the distributions of Glucose, Blood Pressure, Insulin and Skin Thickness have been significantly centered by missing data imputation. Otherwise, they would be severely right skewed due to high proportion of 0, symbol of missing data. However, the distribution of age is still a little skewed since the data set collects information of females from 20 years old. Consequently, pregnancies, skin thickness and insulin are also skewed since they are highly related to subject's age.

3.2. Dimension Reduction

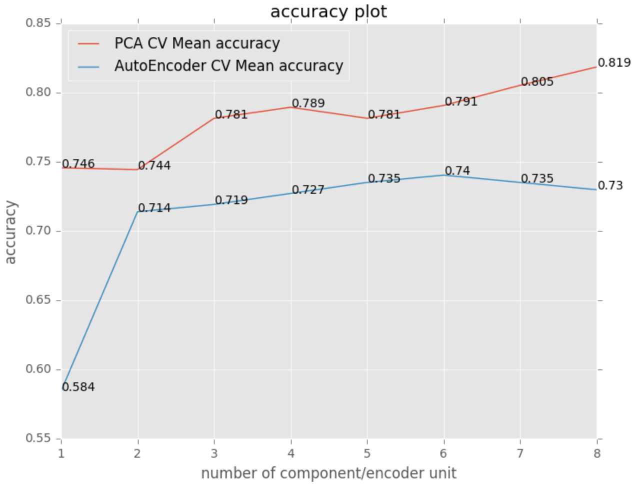

Figure 2. best mean accuracy

For PCA and Auto-encoder, we plot the best mean accuracy computed by Grid Search Cross Validation as a function of number of principal components or second layer size of Auto-encoder as presented by Figure 2. From Figure 2, we can tell that, in general, mean CV accuracy increases as number of principal components or second layer size of Auto-encoder grows. PCA performs better than Auto-encoder in terms of mean CV accuracy. The best mean CV accuracy of PCA is 0.819 which corresponds to inserting all the features. However, a principal component set of size 6 is powerful enough to have a 0.791 mean CV accuracy. Considering the little difference in mean CV accuracy and Occam's razor, we bring the model with a principal component set of size 6 to further study. It is achieved with an architecture with Rectified Linear Unit activation, weight decay 0.620, a 6-neuron input layer (corresponding to 6 principal components) and a 17-neuron hidden layer. The best mean CV accuracy of Auto-encoder is 0.740 which is given by an architecture with Sigmoid activation, weight decay 0, a 6-unit encoding layer and a 6-neuron hidden layer. The model returned by PCA will be further evaluated in the coming section because of its better performance. The architectures are presented in Figure 3 and Figure 4.

3.3. Cross Validation

Next we run n-fold $n = 5, 10, 15, \cdots, 50$ Cross Validation with both of architectures on validation set and plot out mean CV accuracy with error bar as a function of number of folds. On the training-plus-validation set, the optimal model suggested by PCA has a 86.62% training accuracy while the one by Auto-encoder has a 74.04% training accuracy. As it is shown by Figure 5 and Figure 6, both methods have its pros and cons. PCA performs better than Auto-encoder in terms of mean CV accuracy. Generally, accuracy standard deviation increases as with the expansion of fold size. For a given fold size, the accuracy standard deviation of Auto-

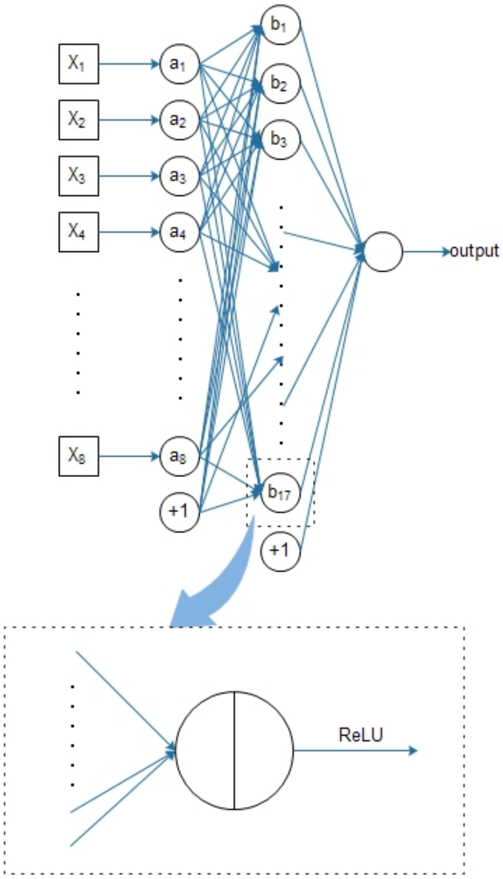

Figure 3. neural network architecture suggested by PCA

encoder is smaller than that of PCA. This can be explained by that

- On one hand, PCA compress the data information only once so it loses possibly useful information at only one place. As to Auto-encoder, it first compress the data information and then retrieve it by multiplying the weights and then sum it up, which brings two places where possibly useful information leak may happen. Therefore, the architecture built with PCA tends to be more accurate since it contains relatively more useful information.

- On the other hand, Auto-encoder encode, decode and feed the decoded data to the following layer. It's forming a really complex hypothesis space. The more complex the hypothesis space, the more likely the result will fall into local optimum. In our case, Auto-encoder estimation easily falls into local optimum so that there is less difference within each cross validation fold than PCA.

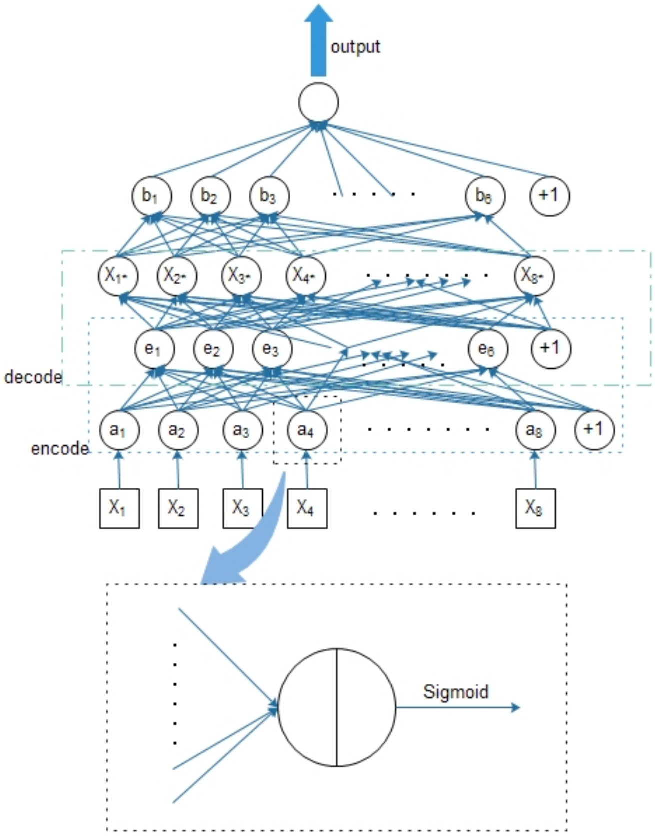

Figure 4. architecture of Auto-encoder

	Truth 0	Truth 1
Prediction 0	40	15
Prediction 1	11	35

Table 1. confusion matrix

3.4. Testing

In this part, we first merge training set and validation set, which brings a data set containing 855 samples. Then we perform Forward and Back Propagation with the architecture given by PCA on the merged set to estimate the weights. Since weight estimation is sensitive to initialization, we run 100 epochs of estimation for $k = 5, 6, \cdots, 104$. In epoch k, we estimate the weights, compute accuracy for k times and then get the mean training accuracy. To get an overall view of the model performance on the merged set, we plot the mean accuracy as a function of epoch in Figure 7. We can see that as we run more epochs, the mean accuracy will finally fall into $(85.50\%, 85.55\%)$. Next we run the model on testing set and report confusion matrix as Table 1. Again since weight estimation is sensitive to initialization, we run 100 epochs of estimation for $k = 5, 6, \cdots, 104$. In epoch k, we compute sensitivity and

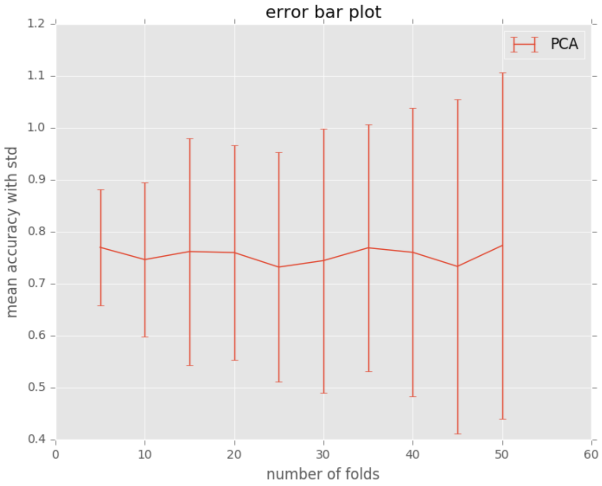

Figure 5. PCA: mean CV accuracy with error bar

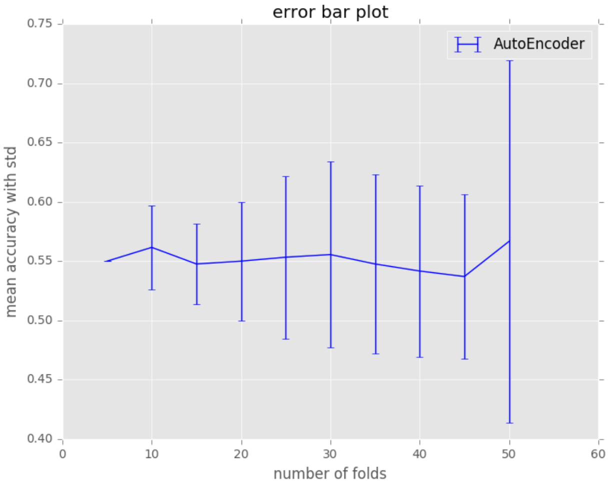

Figure 6. Auto-encoder: mean CV accuracy with error bar

specificity. Also we plot them as a function of epoch in Figure 8. Sensitivity converges to 0.73 while specificity is approximately 0.78. We can see that the model is similarly accurate when dealing with diabetic and non-diabetic samples. The performance is satisfactory and stable.

4. Conclusion

In this project, we mainly use BP Neural Network to predict diabetes mellitus onset based on biological features including blood pressure, skin thickness, insulin, etc. A combination of SMOTE and Tomek Link helps to fix imbalance within response variable and improve model reliability. Two methods, PCA and Auto-encoder, are called for dimension reduction purpose. In our case, the PCA result is better due to relatively less information loss and risk of local optimum. Grid Search Cross Validation and Cross Validation are fabulous tools to select hyper-parameters and promote model generalization. The final model driven by PCA is achieved with an architecture with Rectified Linear Unit activation, weight decay 0.620, a 6-neuron input layer and a 17-neuron hidden layer. It has an overall 85.50% training accuracy and 79.81% testing accuracy, which is a satisfactory, robust and convincing result.

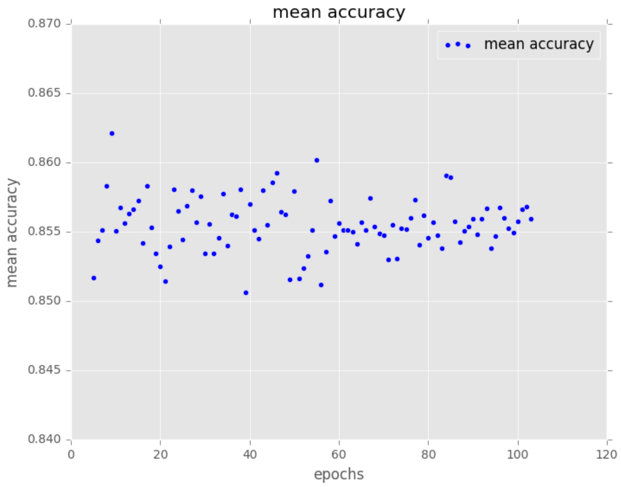

Figure 7. mean accuracy vs. epoch

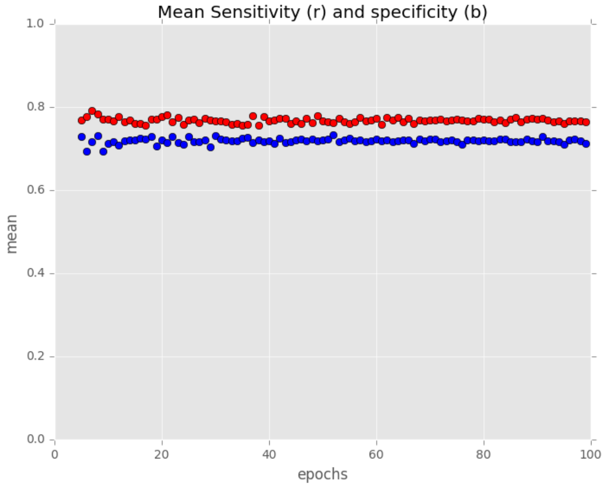

Figure 8. mean sensitivity and specificity vs. epoch

5. Reference

[1]Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, and John N. Clore, ”Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records”, BioMed Research International, vol. 2014, Article ID 781670, 11 pages, 2014.

[2]Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., Johannes, R.S. ”Using the ADAP learning algorithm to forecast the onset of diabetes mellitus”, In Proceedings of the Symposium on Computer Applications and Medical Care (pp. 261–265). IEEE Computer Society Press. 1988.

[3]Knowler, W. C., D. J. Pettitt, P. J. Savage, and P. J. Bennett. ”Diabetes Incidence in Pima Indians: Contributions of Obesity and Parental Diabetes”, American Journal of Epidemiology, 113(2), (pp. 144–156). 1981.