Cardiovascular Disease Classification

Roberto Romero

Dept. Biomedical Engineering

UW Madison

Rromero4@wisc.edu

Introduction:

Heart disease is still the leading cause of death among all people in the United States. Heart disease accounts for 25% of deaths in the United States every year. The goal of this project was to create a binary classifier that will predict if someone has cardiovascular disease or not. We know some contributors to heart illnesses, but the effect of the combination of features is not easily understood. The goal of this program is to see how these different contributing factors combine to affect the heart. A bonus to this project would be to determine which factors have the highest impact on heart health.

A data set containing 70000 record patients will be used to train and test the different algorithms that will be used in this project. Of these 70000 patients, 34979 have heart disease and 35021 are healthy. The eleven contributing factors that will be examined are age, height, weight, gender, systolic blood pressure, diastolic blood pressure, cholesterol, glucose level, smoking, alcohol intake, and physical activity. A limitation of these features is that alcohol intake, smoking, and physically active are binary, yes or no, and subjective of how the patient sees it. A smoker can vary from someone smoking a cigarette every few days to someone smoking a whole pack a day. Cholesterol and glucose levels are classified as 1:normal, 2: above normal, and 3: well above normal. Different combinations of these parameters will be used to see which algorithms work best. This data set is available in Kaggle and a link is in the references.

I approached this project by using three different algorithms. I used linear regression through least squares, support vector machine, and neural networks. A 10-fold cross validation will be used over the training set to find the best parameters for each learning algorithm. Each algorithm will then be trained with their corresponding best parameters.

Pre-Processing on data:

The original data set is a 70000 by 13 table is excel. I imported this data into MATLAB to convert it into a .mat file for easier processing. After this I removed the first column of the dataset which was just the id number of the subject as this will have no affect on heart disease. I also made a separate dataset where I only included the features that were recorded at the hospital: age, height, weight, gender, systolic blood pressure, diastolic blood pressure, cholesterol, and glucose level. I also converted the age from days to years to make this more generalizable. The two data set were then split into 10 different training, validating, and testing data sets.

Algorithms:

Least Square:

The rank of the matrix was calculated and found to be full rank. Since there is a solution to the least square problem a simple least square was performed. To improve the error rate a SVD of the data was used. From there I used 10-fold cross validation to find the best truncated SVD and ridge regression solution. I used lambda values in the range of 0 – 20 for ridge regression. This range is large enough to determine if minimizing the norm of our solution is more ideal than minimizing our cost function.

Support Vector Machines:

Support vector machines (SVM) in general are a better binary classifier than least squares. Unlike least squares, support vector machines decision boundary is not influenced by the extreme values of the data at each end. The SVM models that were used were imported from the sklearn.svm library. A linear SVM was used to see if there is some linear relationship in the data that can then be compared to the linear least squares solution. Then a gaussian svm method was used to classify the dataset. The decision boundary for a gaussian svm does not have to be linear, so this can fit the data if it is not linear.

Neural Network:

A simple 3 layer neural network was used as the last classifier. A sigmoidal function was used as the activation function which is shown below. A stochastic gradient descent was used to update the weights. A step size of 0.15 was chosen. Then ridge regression was needed to control the norm of the weights.

$$z = x^T w$$

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$\delta = (Yhat - Y) * Yhat * (1 - Yhat)$$

$$W^{(t+1)} = W^t - \alpha * X * \delta$$

Results:

The simple least square solution always had an error rate of 50.03%. The truncated SVD method and ridge regression did not change the accuracy of the least square solution. Linear

SVM solution showed a lower error rate of 39% across the 10 different training and testing sets. The Gaussian SVM also showed an error rate of 39%. All these methods performed the same when the subjective features were removed.

The neural network method had an error rate of 50%. At first this was due to our delta function. Since we used a sigmoidal function, python at first was issuing an error because it could not store all the bits of the extremely large and small numbers due to the exponential. To solve this issue I modified the sigmoidal function, if z > 10000 then the output is 1, if z < -10000 then the output is 0. If -10000 < z < 10000 then the original sigmoidal function would be used. The difficulty with this approach is using an appropriate initial guess for the weights. The delta function is as follows:

$$\delta = (Yhat - Y) * Yhat * (1 - Yhat)$$

At first I set all the weights to zero. Using the Sigmoidal function this predicts all the predicted values to 1. This makes the last term in the delta function equal to zero (1 – Yhat) = 0. When I predicted all the weights to equal one, this produced a large z value, which will then predict all the values to be 0. The second term in the delta function is the predicted value, which means we will be multiplying by zero. When random values for weights were assigned the classifier tended to classify all the data to 0 or 1, leaving us with the same issues as before. Since this will all make the delta functions equal to zero. We were never updating our weights, making stochastic gradient descent impossible no matter how many runs we did. To solve this problem I updated the stochastic gradient to include a ridge regression. The updated weights were then:

$$W(t+1) = W - \alpha * X * \delta - \ 2 * \alpha * \lambda *$$

This updated the values of the weights. However, this did not improve the accuracy of our model.

Discussion:

The SVM method proved to be the best algorithm for classifying the data. I believe this is because the data has a lot of extreme values, and this is harder for the other algorithms to distinguish. The least square solution is influenced to much by the extreme values in the data set. The sigmoidal activation function was not the best since it pushed most of the data to be either 0 or 1, and this greatly affects the stochastic gradient descent. The derivative at these values is then 0 making stochastic gradient descent inefficient. Even when trying to control the weights using a regularization term proved to be ineffective. In the future to improve the neural network a better activation function can be used, and more hidden nodes can be added to the network. The current neural network is limited to classifying linear data. A more complex model will probably be a better fit.

There are some concerns with the dataset. The publisher of the dataset on Kaggle did not mention where they got the data set. There were also some training samples that did not make a lot of sense. For example, one of the subjects had a height of 2.4 feet and a weight of 370 lbs, and some subjects weighed as low as 22 lbs.

Conclusion:

In conclusion the SVM method is best due to the dataset we currently have. If we want to improve our models I would suggest to first improve our data set. Other factors that would be interesting to see are family history of heart health, stress, and diabetes. Family history and diabetes are easier feature to measure, while stress can be subjective. The way they measure blood pressure can also be changed as it was only recorded at the visit and not over a period of time. A perfectly healthy patient could be stressed at the time of the diagnostic, elevating his blood pressure. Controlling the variability in these features can be time consuming and could take a long time.

After we have this data set then we could try our models again. The formatting of the data will reduce the extreme values we have which could improve the least squares method if it the data is linear. From there we could fine tune the models and determine if we need to reassess the data again.

References:

CDC Heart disease in the US: https://www.cdc.gov/heartdisease/facts.htm#:~:text=Heart%20disease%20is%20the%20leading,1%20in%20every%204%20deaths.

Kaggle Data set: https://www.kaggle.com/sulianova/cardiovascular-disease-dataset

Project Github: https://github.com/rromero4/CardiovascularDiseasePrediction