Final Project Report

CISC 5800 Machine Learning (Fall 2017)

Prof. Daniel D. Leeds

Muye Zhao

December 2017

1. Introduction

The goal of this project is to improve diagnosis of true positives of heart disease so that, in reality, the right patients can more promptly receive crucial treatment and have more time with their doctors. I aim to accomplish this by choosing some creating some algorithms that classify the cardiac Single Proton Emission Computed Tomography (SPECT) images as normal or abnormal. I use three machine learning methods, logistic classifier, support vector machine (SVM) and neural network, along with a data set of over 267 instances provided by UCI Machine Learning Repository. I use accuracy as my evaluation. logistic classifier, SVM and neural network classifier can achieve relatively good performance higher than 65% on the database. But the results got by SVM looks wired. Due to the small data size, to improve models in the future, we need feed our models more new data or consider other machine learning methods which may be more suitable for this kind of image data.

2. Dataset

2.1 Dataset Description

I got this dataset[1] from UCI Machine Learning Repository. This dataset describes diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images of patients. The database of 267 SPECT image sets (patients) was processed to extract features that summarize the original SPECT images. As a result, 44 continuous feature patterns were created for each patient. And each of the patients is classified into two categories: normal and abnormal. The dataset is divided into training data "SPECTF.train" which includes 80 instances, and testing data "SPECTF.test" which includes 187 instances. There are 55 normal instances and 212 abnormal instances in the whole dataset. In training dataset, the number of both of the two classes are equal, 40 normal instances and 40 abnormal instances, which is fair for training models. In testing dataset, much more abnormal instances are there, where the number of the normal and the abnormal are 15 and 172 respectively. After looking through the whole dataset, a good news is there is no missing attribute values, which means it saves me a lot of time of cleaning the data. Another good news is all continuous attributes have integer values from the 0 to 100. The same scale data can avoid some unexpected influence on the results and save time of normalizing and/or standardizing.

It is not a big size dataset but a good one for testing ML algorithms. Some researchers have done some interesting work on this dataset. Their machine learning algorithm achieved 77.0% accuracy[2]. Let's see how well my algorithms will unfold.

2.2 pre-processing

2.2.1 Data Separation

The first thing we are going to do is separate our training data into a training set and a validation set. Considering the size of our training data is small, bootstrapping seems to be a good choice. But that will change the original distribution of the training data, which will bring us bias. Then I tried both hold-out and k-fold cross validation. But k-fold doesn't show me a better performance than Hold-out, and sometimes even worse. So I choose hold-out way, the simplest but a very common a way, to separate the dataset. I set the size of testing set is 4/5 of the training dataset. And the rest 1/5 is validation set.

2.2.1 Feature Reduction

As mentioned about, the quality of the data is quite good. But no need for cleaning, normalizing and/or standardizing doesn't mean no need for any preparation before we feed it into our models. Every record has 44 features, which is not a huge number for image data. But it is better to apply feature selection methods to reduce the number of features, which can make the data easier to learn and the learning process more efficient. I choose a very commonly used feature selection method -- principal component analysis (PCA).

Before doing PCA, we need to set up number of principle component. The number is not selected randomly. I'd like to look at variances of the data and calculate their contributes to the total variance of the whole set, to see use how many dimensionalities can explain a certain percentage of the total variance. The relation between the percentage of explained variance and number of features shown as follow:

percentage	0.9	0.95	0.99
Number of Features	10	16	28

Table 2.2 PCA

We'd like to use lower dimensionalities to express the information but with as less as possible information loss. In this chart, let's see the third column. To explain 99% of the total variance, we need at least 28 features. It is not very few, but already 16 features, about 36.36% less, which is not bad. So in our project, we set the percentage of variance explained by each of the selected components as 0.99 and the number of principle component as 27.

3. Learning Method

3.1 Logistic Regression

Logistic regression might be the most commonly used linear model in machine learning. Unlike other sophisticated machine learning model, logistic regression is easy to implement and efficient. Although the accuracy performance is not as good as other non-linear model, the overall performance is acceptable.

The mechanism behind logistic regression is fairly easy. Assuming we have a hyperplane to separate our training data. The goal for our hyperplane is to minimize the error classified labels.

For each data point, we use the $w^T x + b$ to determine its predict label.

$$\text{class}(x) = \begin{cases} 1, & w^T x + b \geq 0 \\ 0, & w^T x + b < 0 \end{cases}$$

Since the value of $w^T x + b$ can be extremely large, and we'd like to generate the probability for each classification. We introduced the sigmoid function to control our prediction value.

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

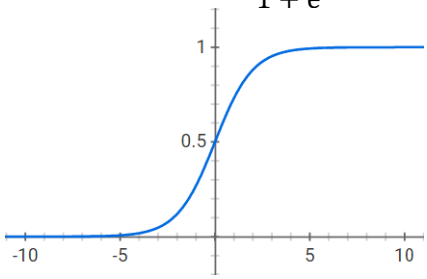

Fig 3.1 sigmoid fundtion

Then, our classification would be [4]:

$$\text{class}(x) = \begin{cases} 1, & \text{sigmoid}(w^T x + b) \geq 0.5 \\ 0, & \text{sigmoid}(w^T x + b) < 0.5 \end{cases}$$

After we made our decision model, the major task would be training the weight vector. The weight w can be trained by gradient ascent. The training step of gradient ascent is shown as follow:

$$\Delta w_j = \varepsilon x_j^i (y^i - \text{sigmoid}(w^T x + b))$$

However, if hyper parameter w is large enough, it may disregard the value of x and result in overfitting. We want to minimize these imbalanced parameters. Therefore, we introduce a penalty rule to reduce the weight in our hyperparameters: L2 regularization.

The general form of L2 penalty is like $\lambda \|w\|_2$. Applying it to gradient ascent, we can generate a new ascent rule for logistic regression. The training step is modified as follow:

$$\Delta w_j = \varepsilon \left(x_j^i \left(y^i - \text{sigmoid}(w^T x + b) \right) - \frac{w_j}{\lambda} \right)$$

3.2 Support Vector Machine (SVM)

Support vector machine (SVM), which is also called support vector networks, constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space. A good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

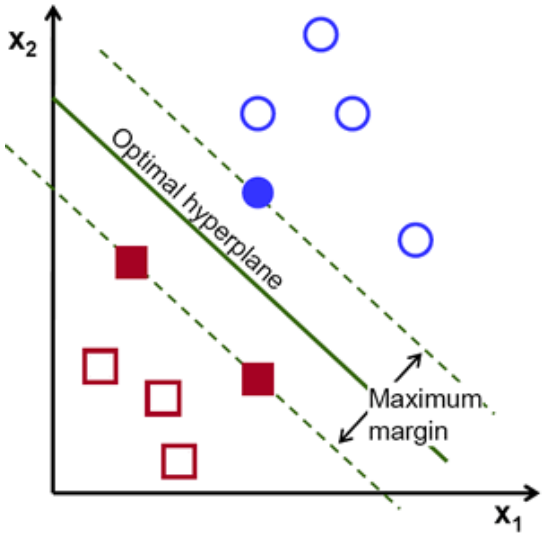

Fig 3.2 SVM

We want to maximize:

$$\text{margin} = \frac{2}{||\omega||^2},$$

which is equivalent to minimize:

$$\mathrm{L}(\omega) = \frac{||\omega||^2}{2}.$$

But subjected to the following constraints:

$$\mathrm{f}(\chi_i) = \begin{cases} 1, & \text{if } \omega \cdot \chi + \mathrm{b} \geq 1 \\ -1, & \text{if } \omega \cdot \chi + \mathrm{b} \leq -1 \end{cases}$$

In addition to performing linear classification, SVM can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

Hyper-parameters for learn in:

$$argmin_{w,b} w^T w + C \sum_i \varepsilon_i$$

where,

- C is influences tolerance for label errors versus narrow margins,
- $\boldsymbol{\varepsilon}$ is influences effect of individual data points in learning,
- T is number of training examples, L is number of loops through data -- balance learning and over-fitting,
- $\boldsymbol{\lambda}$ influences the strength of your prior belief.

3.3 Neural Network

Neural network is no doubt a little overwhelming in recent years. The generality and distributed attribute made it work in many research field like computer vision, natural language processing and bioinformatic. There are hundreds of neural networks currently, in our project we only use the traditional multi-layer perceptron.

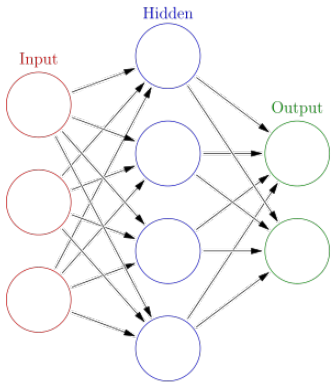

Fig 3.3 A typical Neural Network Structure[6]

The basic mechanism behind the neural network is feedforward and back propagation. Each neuron, we still pick sigmoid function as our activation function. Therefore, our feed forward function is shown as follow:

$$r_k^m = \text{sigmoid}(\Sigma_j w_{k,j}^m r_j^{m-1} + b_k^m)$$

The major difficulty is to update the weight of every pair of neuron. The updating rules called back propagation, rules as follow:
For top layer:

$$\Delta w_{k,j}^m = \varepsilon \delta_k^m r_j^{m-1}$$
$$\delta_k^m = (1 - r_k^m)(y - r_k^m) r_k^m$$

For non-top layer:

$$\Delta w_{k,j}^m = \varepsilon (1 - r_k^m)\left(\Sigma_n w_{n,k}^{m+1} \delta_n^{m+1}\right) r_k^m r_j^{m-1}$$
$$\delta_k^m = (1 - r_k^m)\left(\Sigma_n w_{n,k}^m \delta_n^{m+1}\right) r_k^m r_j^{m-1}$$

4. Result

We use accuracy as our evaluation method.

$$\text{accuracy} = \frac{\text{number of correct label}}{\text{number of testing data}} \times 100\%$$

The matrix of each classifier is provided as follow:

4.1 Logistic Regression

I use my self-defined logistic classifier for the method. Its definition is in the file “logistic.py”.

Let’s set epsilon=0.0001, max iteration=10 and use L2 penalty. The λ relation with the accuracy is shown as follow:

λ	1	3	5	10	15
Validation Accuracy	0.764706	0.882353	0.882353	0.882353	0.823529
Test Accuracy	0.566845	0.566845	0.513369	0.609626	0.524064

Table 4.1.1 Relation between λ and accuracy

Let's set max iteration=10, λ =10 and use L2 penalty. The relation of epsilon and accuracy is shown as follow:

ε	0.1	0.01	0.001	0.0001	0.00001
Validation Accuracy	0.882353	0.764706	0.882353	0.705882	0.764706
Test Accuracy	0.673797	0.636364	0.513369	0.524064	0.593583

Table 4.1.2 Relation between ε and accuracy

We set epsilon=0.1, max iteration=10, λ =10. The relation of penalty method and accuracy is shown as follow:

Penalty	No Penalty	L2	L1
Validation Accuracy	0.764706	0.882353	0.764705
Test Accuracy	0.641711	0.663102	0.668449

Table 4.1.3 Relation between penalty method and accuracy

4.2 Support Vector Machine (SVM)

I use scikit-learn library svm as my classifier. I choose three parameters, kernels, penalty coefficient C and distance coefficient γ. Their default values are 'rbf', 1.0 and auto (If gamma is 'auto' then 1/n features will be used instead). Let's just use the default values for the parameters who are not under test so far, if not specified.

kernel	linear	rbf	Sigmoid	poly
Validation Accuracy	0.764705	0.411764	0.705882	0.823529
Test Accuracy	0.604278	0.919786	0.748663	0.502674

Table 4.2.1 Relation between kernel type and accuracy

C	1	3	5	10	15
Validation Accuracy	0.411764	0.411765	0.411764	0.411764	0.411764
Test Accuracy	0.919786	0.919786	0.919786	0.919786	0.919786

Table 4.2.2 Relation between penalty coefficient C and accuracy

γ	1	3	5	10	15
Validation Accuracy	0.411764	0.411764	0.411764	0.411764	0.411764
Test Accuracy	0.919786	0.919786	0.919786	0.919786	0.919786

Table 4.2.3 Relation between distance coefficient γ and accuracy

4.3 Neural Network

I use scikit-learn library MLPClassifier as my classifier. We choose number of neurons, number of layers, alpha value(L2 penalty parameter) as our hyper parameters. By default, we set max iteration=500, tolerance=0.0001, activation function='logistic', using stochastic gradient descent as our back-propagation method.

We set the number of neurons in each layer=50, alpha=0.0001, the relation between number of layers and accuracy as follow:

number of layers	1	3	5	8	10
Validation Accuracy	0.647058	0.764706	0.823529	0.823529	0.764706
Test Accuracy	0.080213	0.663102	0.657754	0.625668	0.625668

Table 4.3.1 Relation between layers and accuracy

We set the number of layers=3, alpha=0.0001, the relation between number of layers and accuracy as follow:

number of neurons	10	20	50	80	100
Validation Accuracy	0.882353	0.764706	0.823529	0.823529	0.823529
Test Accuracy	0.614973	0.652406	0.647059	0.620321	0.625668

Table 4.3.2 Relation between neuron in each layer and accuracy

We set the number of layers=3, number of neurons in each layer=20, the relation between alpha and accuracy as follow:

alpha	0.00001	0.0001	0.001	0.01	0.1
Validation Accuracy	0.764706	0.764706	0.764706	0.764706	0.764706
Test Accuracy	0.657754	0.663102	0.652406	0.647059	0.759358

Table 4.3.3 Relation between alpha and accuracy

5. Conclusion

For this dataset, logistic classifier, SVM and neural network classifier can achieve relatively good performance higher than 65%. But the results of SVM looks wired: making changes of C

and gamma even cannot see any influence on the results; and the gap between validation accuracy and test accuracy is relatively bigger than others. Maybe there's some error there. We need to find some ways to explain this.

Due to the small data size, to improve models in the future, we need feed our models more new data or consider other machine learning methods which may be more suitable for this kind of image data.

Work Cited

[1] original owners: Krzysztof J. Cios, Lukasz A. Kurgan
University of Colorado at Denver, Denver, CO 80217, U.S.A.
Krys.Cios@cudenver.edu
Lucy S. Goodenday
Medical College of Ohio, OH, U.S.A.
-- Donors: Lukasz A.Kurgan, Krzysztof J. Cios

[2] Kurgan, L.A., Cios, K.J., Tadeusiewicz, R., Ogiela, M. & Goodenday, L.S.
"Knowledge Discovery Approach to Automated Cardiac SPECT Diagnosis"
Artificial Intelligence in Medicine, vol. 23:2, pp 149-169, Oct 2001

[3] Discriminative Model & Generative Model. Retrieved Dec 4, 2017
http://www.cnblogs.com/TenosDoIt/p/3721074.html

[4] A Beginner's Guide to Neural Networks in Python and SciKit Learn 0.18. Retrieved Dec 4, 2017 https://www.springboard.com/blog/beginners-guide-neural-network-in-python-scikit-learn-0-18/

[5] Choosing the Number of Components or Factors to Include in a PCA or EFA. Retrieved Dec 4, http://hosted.jalt.org/test/bro_30.htm

[6] Artificial neural network. (2017, Dec 9). In Wikipedia, The Free Encyclopedia. Retrieved Dec 9, 2017, from https://en.wikipedia.org/w/index.php?title=Artificial_neural_network&oldid=778883975