PCML Project II Report

Wei Ma

December 2015

Abstract

The report demonstrates the analysis and results of the project-II. In the project, I use Logistic Regression,SVM and Neural Network to classify the images. In the project, I will discuss the parameters of each model and understand them how to affect the model by the practicality.

1 Data Description

The file, train.mat, contains three members,scilicet two feature matrixes X_{cnn}, X_{hog} and a label vector y. There are 6000 sample cases in total. In the matrixes X_{cnn} and X_{hog}, the columns represent the features and the rows represent the observation. There are four categories, Airplane,Car,Horse and the default type in the project. They are represented 1,2,3 and 4 respectively in the label vector y. Their proportions are showed in the Figure 1. The result will be compared to the predicting total number of each category in the classification test.

X_{hog} is 6000×5408 and it is not a sparse matrix. However, compared with X_{cnn}, we can visualize X_{hog} because HOG(Histograms of Oriented Gradients) can show the extracted the edges directions and the intensity based on the small connected regions. Figure 2 shows the car image train00566.jpg and its HOG figure. When looking at the right image of the Figure 2 carefully, you can find the edges of the car in the red frame. However, it is very hard to identity what the hog image is if we do not compare it with the original image. HOG is usually used with the other method such as Scale-invariant feature transform(SIFT).

Matrix X_{cnn} is 6000×36865. The data set is extracted by CNN(Convolutional neural network). It is a sparse matrix whose 95.52% entities are zero. In the project, all the methods will use the data set X_{cnn}. I also tried X_{hog} to train SVM and the neural network model(BNN) and find that it has a higher balance error rate than X_{cnn}, more than 20%. I think that it is caused that HOG features just can represent the gradient of object but CNN features contain more information such as the shape, color and brightness.In the HOG image(see Figure 2), the background of the image still exits and it is noise. Hence, X_{cnn} is better than X_{hog}.

About how to split the given training data to the training model data and the test validation data, I tried different splitting proportions on the given training data as the input of Logistic Regression and Feedforward Backpropagation Neural Networks(BNN). As a result, I find a good splitting proportion that 80% of the given training data is the training model data and the rest is the test validation data. I will use the splitting proportion. For all the models below, I will use the the training model data to train the models and the test validation data to test the models. Through the splitting method, the size of the test validation data is 1200, which is enough to

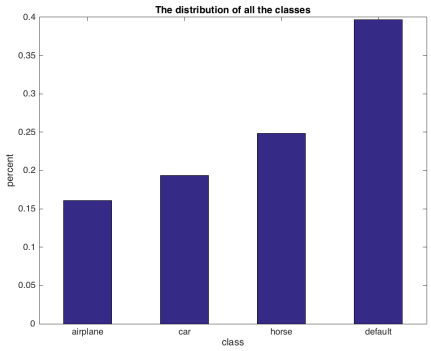

Figure 1: The number of the categories

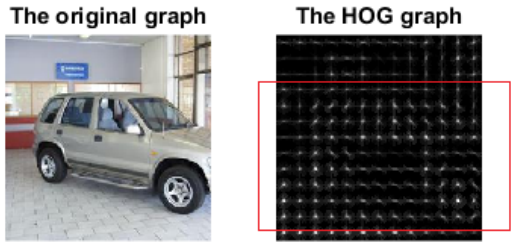

Figure 2: HOG image

test the model.

For all the methods, before running them, I will random all the examples by setting the seed as the system time. In the deep learning box, I read the code carefully and find that for each poch, it will example the cases from the train data(Stochastical Gradient Descent). Using the cross validation, running the code needs a little longer time. So I just use the cross validation in Logistic Regression. Before running the code, changing the random seed and stochastically ranking the examples have the same effect as the cross validation. I also use the isolated test validation data to test the trained model to avoid overfitting. So even though I don't use cross validation, I think I have taken the sufficient methods to replace it and the running time also gets shorter. As a result, I find there is no difference with the other teams that use the cross validation by comparing their results with mine.

2 Logistic Regression (implemented by myself)

The logistic regression is a binary classification model.The k-fold crossing validation is used to train the model. In general, 10-fold cross-validation is commonly used[1]. Based on the conclusion from the past research, we choose 10-fold cross-validation.To transform the multiple label to the binary label, we set 1,2 and 3 to 1 and set 4 to 0. In the new label vector, 60.3% labels are 1 type and the others are 0 type. The proportion will be used to compare with the testing result. The data need be normalized to avoid too large absolute values of the exponents for sigma function, $\frac{1}{1+e^{-x}}$.

I use non-penalized logistic regression model. The learning rate α is vital for the model.A fix learning rate α sometimes can be a problem.If α is large, the convergence speed will be accelerated but it has a probability to make the model unstable.Compared with the large learning rate, the small learning rate needs too long time for convergence. To address the problem, α should decay gradually as the time goes. The method combines the advantages of the large step and the small step. When we use the decreasing α method($\alpha = 2$), the model needs shorter time to converge than the fixed small learning rate and its test BER is around 11.5%.The classifying result on the test data shows that the ratio of these two categories is similar in the test result and the label y. In y, the proportion of 0 and 1 classes have 39.7% and 60.3% respectively. In my classification result,they are 41.8% and 58.2%.

3 Traditional SVM

Support Vector Machine(SVM) is an efficient model in theory for the binary classifying problem.In the part, I will use the function fitcsvm provided by the Matlab library. It provides the flexible parameter settings. In the project, I will try 3 important parameter settings, Kernel Function , BoxConstraint and KernelScale.BoxConstraint is a strategy to try a geometric squence of the box constraint parameter.Increasing BoxConstraint might decrease the number of support vectors, but also might increase training time.See below equation. ε is the slack variables.C is the parameter boxconstrait in the Matlab.

$$L = \frac{1}{2}W^2 + C\sum \varepsilon_i \tag{1}$$

If C is close to 0, then we don't pay that much for points violating the margin constraint.We can minimize the cost function by setting w to be a small vector - this is equivalent to creating a very wide "tube" or safety margin around the decision boundary (but having many points violate this safety margin). If C is close to inf, then we pay a lot for points that violate the margin constraint, and we are close the hard-margin formulation.

See Figure 3.Train BER is gotten from the train data.It can help us uderstand how the model fit the train data. Test BER is gotten from the isolated test validation data. When the boxconstrait is very small , the kernel linear function is the best when the data is X_{cnn}. In contrast, RBF function is not better than the linear kernel. I think the reason is that the size of features is far more than the size of observation but RBF kernel need more observation cases. Another reason is that too many observations are in the margin but there is no enough penalty for them. As the boxconstraints increase, the train BERs of the the three kernel function decrease. After the boxconstraint exceeds 2, the train BERs are close to 0. At the same time, the test BER of the three functions flows around the 8%.When the boxconstraint is more than 3, the test of RBF is the lowest one around 7.6%. To make sure the effect that the relationship between the size of the features and the number of the obeservation, I use the X_{hog} to validate my conclusions. See Figure 4. In the beginning of the lines in the Figure 4, the test BERs of the three kernels are very close. However, the trend of the linear's test and train BER is pretty different from the former. I think that is caused by too much penalty when the boxconstraint gets large and the model with linear kernel is underfitting. I also find Polynomial kernel has a stable performace in the data X_{cnn} and X_{hog}. I think that is because its complexity is between the linear and RBF. I set Polynomial Order as 2 because the larger degrees tend to overfit.

KernelScale is to try a geometric sequence of the RBF sigma parameter scaled at the original kernel scale. It can prevent the overfitting and uderfitting. In the project, I will use the 'auto' setting for KernelScale.As be shown in the Figure 3, the best BER for the binary classification is 7.6% when the kernel function is RBF and the boxconstraint is euqal to 5.

3.1 SVM for the multiple classification (implemented by myself)

SVM is for the binary class problem. If we do not add something to it, it cannot deal with the multiple classification problem. In the project, I extends SVM from the binary classifiers to the multiple classifiers through two methods implemented by myself.

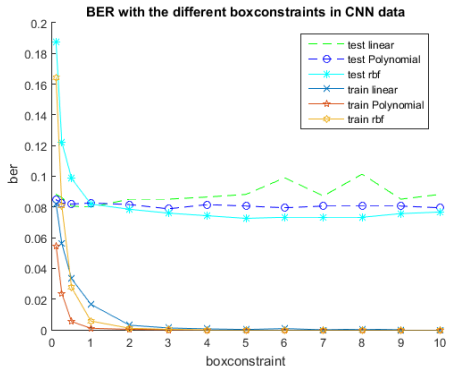

Figure 3: The BER of BNN with the different depths

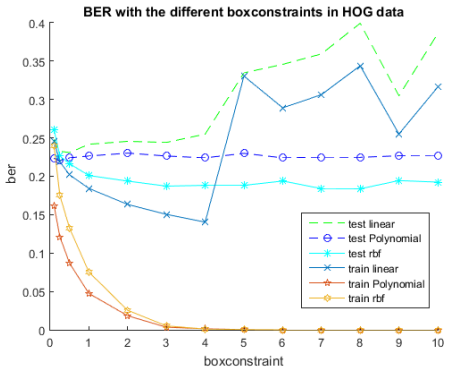

Figure 4: One of the structures of SAE

3.1.1 One to One Voting Method

The method's idea is very simple. At first, 6 binary-class SVM models can be gotten by combining two different labels from the total 4 labels. The six models are trained one by one on the training model data. For the new data, it will be processed by all the models. Every model will decide which label the input data should have and vote a ticket to the label.Finally, the label that gets the most tickets will win. If two or more classes get the same most tickets, one of them is selected randomly. According to the above content, when kernel is RBF and boxconstraint is equal to 5, the test BER is 10.70%.

3.1.2 The Directed Acyclic Graph-SVM Method

The first step is the same as the one to one voting method.The combination of the different two types is also needed. The difference with the former is the decision process. DAG(Directed Acyclic Graph-SVM, see Figure 5) will use a decision tree to replace the voting process. Every edge means a decision. The text 'Not 1' in the graph means the connected node will be chosen as the next node if the decision of the current node is not label 1. The text '2' in the graph means the decision will be made if the current SVM model decides the data's label is 2. In the end, all the data will get a label. In the best parameter settings in the part one of SVM, its test BER is 11.93%. Its test BER is higher than the One to One Voting because it uses less SVM models to classify and save time at the cost of accuracy. If non-leaf node makes a wrong decision, the decision is not redeemed but in One to One Voting, a wrong decision of a SVM model can be corrected by the other SVM models.

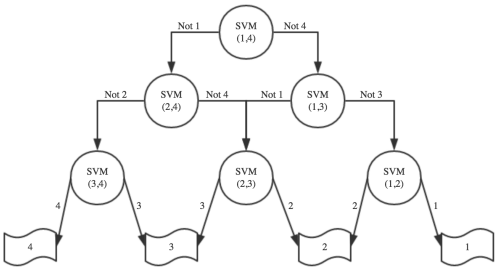

Figure 5: The Directed Acyclic Graph of SVM

3.2 Conclusion for SVM

The train data set X_{cnn} has too large features that is more than 30000 but we just have 6000 samples. Obviously, the number of the samples is not enough for a complicated kernel function. In that case,SVM should uses the linear kernel function but increasing BoxConstraint can be another solution so that RBF and Polynominal kernels can be used.To confirm the conclusion, X_{hog} is used to validate. X_{hog}s features and its observations have a close number,which is different from X_{cnn}. I also get the confirmation from Andrew Ng's course.In the part of SVM, to make sure the data is not too big for RBF, the "Standardize" parameter should be set to true. When it is true, the data will be normalized and rescaled.

4 Neural Network

Neural Network is more different than the methods before we used. Neural Network is based on the biology model and simulate the intelligent brain of human. It needs longer time to train than SVM and logistic regression if the hidden neurons are too many. In the part, I also implement the counter propagation network (CPN). CPN is a semi-supervised neural network.I use the feedforward backpropagation neural network(BNN) and stack-auto ecncoder (SAE) provided by the deep learning toolbox.Neural Network is non-convex so it is tricky to find the local optimal solution by setting its parameters.

4.1 BNN

The deep learning toolbox(DLT) designs BNN complexly through some efficient techniques to improve the model.Before starting using it, there are some introductions about its important parameters. Input zero mask and dropout fraction can prevent the model overfitting.

BNN algorithm mainly contain three steps. To start BNN, its weights need be initialized randomly. The weights cannot be the same.Otherwise, all the outputs of the neurons are the same. See the Figure 6.It shows its structure and flow of BNN.It mainly contain 3 steps: i) Feed-forward computation.ii) Backpropagation to the output layer the hidden layer.iv) Weight updates.

The algorithm is stopped when the value of the error function has become sufficiently small. In the deep learning toolbox, the parameter numepoches is the iterations of the algorithm. The backpropagation procedure to compute the gradient of an objective function with respect to the weights of a multilayer stack of modules is nothing more than a practical application of the chain rule for derivatives. The key insight is that the derivative (or gradient) of the objective with respect to the input of a module can be computed by working backwards from the gradient with respect to the output of that module (or the input of the subsequent module)[2].

I tried some different number of the layers and the hidden neurons to find the best structure of BNN in the project. As the size of layers increases in a Neural Network , the capacity of the network will be enhanced.The neural network with more hidden layers and neurons can express more complicated patterns. However, it also can cause overfitting. There are some ways to solve the overfitting problem such as decreasing the capacity of the network, penalty,momentum,dropout and input mask.The momentum introduces a "damping" effect on the search procedure, thus avoiding oscillations in irregular areas of the error surface by averaging gradient components with opposite sign and accelerating the convergence in long flat areas. In some situations it possibly avoids the search procedure from being stoped in a local minimum, helping it to skip over those

regions without performing any minimization there.For the dropout fraction, each element of a layer's output is kept with probability p, otherwise being set to 0 with probability $(1-p)$. One reason why dropout gives major improvements over backpropagation is that it encourages each individual hidden neuron to learn a useful feature without relying on specific other hidden neurons to correct its mistakes. Input mask acts on the input layer in the similar way as the dropout. Input mask can help the network learn more about the features through setting some features to zero at each iteration.As can be seen from the Figure 7 in the next page, test BER arrives at the bottom at hidden layers=3 when momentum=0.5, learning rate =1, input mask=0.5, dropout fraction=0.5. They are the best parameters in my project. The right classification results have a similar proportion with the Figure 1.From the left to the right respectively, my classification proportions are 15%, 20%,21.25% and 35.42%. I do the indrect binary classification based on the result of the best multiple classification result mentioned above and its test BER is 8.14%. I use BNN to do the binary classification directly in the same parameter settings and its test BER is 8.18% that is lower than the former. The reason may be that it omits some information about the labels,which regarding 1,2 and 3 as an integration.

As above mentioned, the capacity of the network can be increased when we increase the number of the neurons but the related other techniques such as momentum must be used to prevent overfitting. In the project, I changed the neural network [N 200 100 4] to [N 500 250 2]. In the same parameter settings, its test BER just has 7.4% in the binary direct classification. In the project, I use BNN to give the classification result of the test.mat.

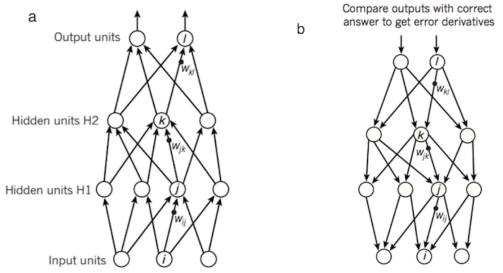

Figure 6: The steps of the backpropagation neural network

The model has two hidden layers.a) it is the feed-forward computation.w_{jk} is the weight between the neuron j in the current layer and the neuron k in the next layer. At each layer, we firstly compute the total input to each neuron, which is a weighted sum of the outputs of the neurons in the layer below. Then an active function is applied to the total input to to get the output of the neuron.b) At each hidden layer we compute the error derivative with respect to the output of each neuron, which is a weighted sum of the error derivatives with respect to the total neurons to the neurons in the layer above.

4.2 SAE

SAE can extract the main features from the input. In the project, I tried to do PCA for the input data by pca function provided by Matlab but unfortunately I didn't get the desired result because it deat with too much data at a time, which exceeds its maximum limitation.One way to address the problem is dividing the features of the input to a few groups. However, the different group is isolated. It is possible to eliminate some important features. So I didn't use PCA to reduce the dimension of the input. And fortunately, I find the method SAE.

Autoencoders are a method for performing representation learning, an unsupervised pretraining

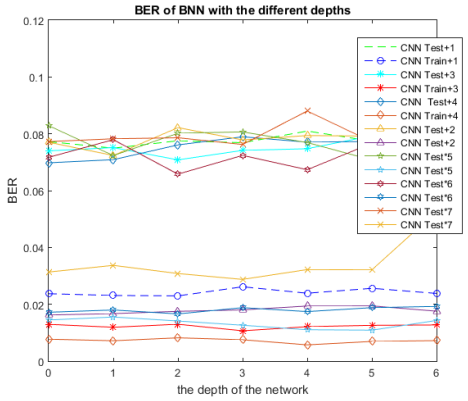

Figure 7: The BER of BNN with the different depths

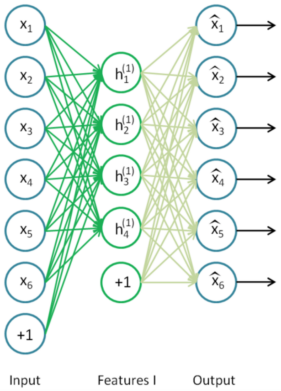

Figure 8: One of the structures of SAE

Figure 7).In the project, I set the input mask equal to the dropout fraction. '+' and '*' represent the different parameter sets. '*' parameter settings has an additional momentum parameter. +1,+2,+3 and +4 represent different values of the input mask and dropout fraction,respectively 0.2,0.3,0.5 and 0.8. (CNN Test+1, CNN Train+1) is a pair of lines. The former is the test BER line and the later is the train BER line.I also set different values of the momentum based on the parameter setting of (CNN Test+3, CNN Train+3) which has the lowest test BER among +1,+2,+3 and +4. *5,*6 and *7 represent the different momentum values,respectively 0.1 0.5 and 0.8. The learning rate is set to 1. The prefix indicates the data set X_{cnn} is used in the project. The x-axis represent the number of the hidden layers.The structures of the network is [N 4],[N 100 4],[N 200 100 4],[N 400 200 100 4],[N 500 400 200 100 4],[N 500 400 200 100 50 4],[N 500 400 200 100 50 4]. N represents the number of the features of the input. For example, for the structure [N 100 4], from the left to right, they are the output layer, the first hidden layer and the output layer. The number N, 100 or 4 is the number of the neurons in a layer. Figure 8).It has one hidden layer.Each circle represent a neuron.It has six input neurons $x_1, \ldots, x_6$ and one hidden layer that has 4 neurons $h_1^1, \ldots, h_6^6$. Its output layer has six neurons,$\hat{x_1}, \ldots, \hat{x_6}$.The neurons labeled $+1$ are called bias units, and correspond to the intercept term.

process during which a more useful representation of the input data is automatically determined.For many supervised classification tasks, the high dimensionality of the input data means that the classifier requires an enormous number of training examples in order to generalize well and not overfit. One solution is to use unsupervised pretraining to learn a good representation.For ordinary autoencoders, we usually want that $n < m$, where n is the number of observation and m is the dimension of the input, so that the learned representation of the input exists in a lower dimensional space than the input. This is done to ensure that the auto encoder does not learn a trivial identity features. For X_{cnn} data, it has much more dimensions than the number of the observations. Through SAE, X_{cnn} can reduce its dimension and keep the nontrivial features.

An autoencoder is a neural network with a single hidden layer and where the output layer and the input layer have the same size. SAE will be used to pretrain the weights BNN except the weights between the last hidden layer and the output layer. SAE does the best to make the output close to the input so that the hidden layer can learn the primary features of the input. To describe the SAE model, I will give an example.A four-layer target network wants to be pretrained. The input layer has 6 input neurons, the first hidden layer has 5 neurons, the second layer has 4 neurons and the output layer has 2 neurons. At first, there are two sub network that will be built. The first one network's structure is Figure 8. Both of the input and output layers have 6 neurons and the hidden layer has 4 neurons. Next, the first sub network is trained by BNN and then We can get the weights between the input and the first hidden layers in the target network. Similarly, through training the second sub network, we can get the weights between the two hidden layers in the target network. Then, we can get all the initial weights of the target network except the weights between the last hidden layer and the output layer. In the end, the target network will be trained again by BNN.

Sparsity is a sparsity parameter, typically a small value close to zero (Sparsity is usually set to 0.05). In other words, we would like the average activation of each hidden neuron to be close to 0.05 (default value in the deep learning toolbox). Some conclusions from the BNN part can be used directly here. The learning rate is 1 and the momentum, the input mask and the dropout fraction is equal to 0.5. Its test BER of the multiple classification is 10.55%. SAE should be better than BNN in theory but actually it is not. I think it is caused by the parameter settings. The best parameters for BNN may be not good for SAE.

4.3 CPN (implemented by myself)

The counter propagation is a semi-supervised algorithm. It has three layers in total. The unsupervised Kohonen algorithm will be used between the input layer and the hidden layer. Kohonen Algorithm can find the clusters of the data. From the input to the hidden level,its strategy is winner-take-all.See below the equation 1. Winner is the selected hidden neuron. V_i represent the winner neuron. Ω_{hidden} is the set of the hidden layer neurons. W_i is the weight matrix between the hidden neuron i and the input layer. X is an input observation and it is a vector.

$$Winner = V_i, \textbf{if for } \forall j \in \Omega_{hidden}, W_i X \geq W j X \tag{2}$$

To make sure when the hidden layer converges(Kohonen layer), I come up with an solution.

$$difference_t = \frac{1}{t} \times \sum_{0 \leq k \leq t} \left| C_{k+1}^i - C_k^i \right| \tag{3}$$

Here, the C_{k+1}^i represents the output of the i_{th} neuron in the hidden layer at the $k+1$'s iteration. t is the current iteration time. If the average difference between C_{k+1}^i and C_k^i is close to one constant after n iterations, we can conclude that it has converged. In practicality, the method is validated. When Kohonen layer keeps stable, the hidden neuron is the center of some cluster in the training data(see equation 3). HW_i^k is the connection weights vector between the i_{th} hidden neuron and the input layer. k represent the k_{th} cluster. N^k is the number of the input that the k_{th} cluster contains. X_i^k is an observation in the k cluster.

$$HW_i^k = \frac{1}{N^k} \sum_{i=1}^{N^k} X_i^k \tag{4}$$

From the hidden level to the output level, I adopt the linear output function. The learning process between the hidden layer and the output layer is supervised learning.The multiple classification's test BER is around 84.62%. CPN is implemented without penalty or other parameters to avoid the overfitting or the underfitting so it should not be better than BNN and SAE.

5 Conclusion

Logistic regression and SVM are used for the binary classification. SVM is also extended for the multiple classification by the one to one voting and the directed acyclic graph. For SVM, I explores Kernel funciton and BoxConstraint parameters. For the neural network, I tried BNN,SAE and CAP. The neural network is non-convex. To find the best parameter setting of the neural network, I tried different network structure and different values of the parameters.In my future work, I think that the classification result can be improved by facilitating the extraction of the features.

6 Reference

[1].McLachlan, Geoffrey J.; Do, Kim-Anh; Ambroise, Christophe (2004). Analyzing microarray gene expression data. Wiley.
[2].Petrini M. Extended Network for Backpropagation Algorithm[J]. OF THE UNIVERSITY OF PETROAN ECONOMICS,2012:177.
Announcement
Figure 5 and Figure 7 come from the Internet. When writing the report, I also refer to Andrew Ng's course.