

ECE 1513 Introduction to Machine Learning

**Graduate Project: Classification Problem with Linear
Regression, CNN and SVM**



Dongxu Ren

1004799934

1. Problem Description

Binary classification is a popular topic in machine learning and there are many ways to solve it, in this project I used the SVHN data set as source data set and implemented linear regression, convolutional neural network and support vector machine algorithms on it and adjust the model parameters and problem definition to find the influence that these modifications have on the results and test the differences among the three algorithms.

2. Data Process

The SVHN data set has two attributes, 'X' for training data which has a shape of 73257*32*32*3 and 'y' for training labels which has a range from 1 to 10 and a shape of 73257*1, to run the classification, I extract two types labels, the first type has two kinds of labels which are the corresponding label of number 1 and 2, my algorithm will try to separate them from each other. The second type of label is image 1 which are set to 1 and all images which are not 1 those images are labeled as 0.

To simplify the calculation, it is necessary to convert the colored image to gray scale, as implemented below.

```
def rgb2gray(images):  
    return np.expand_dims(np.dot(images, [0.2990, 0.5870, 0.1140]), axis=3)
```

Fig 2.1 function to convert data to gray scale

The gray scale training data has a shape of 73257*32*32*1 and can reduce the computation complexity greatly.

In addition, some tensorflow functions require the data labels to be in one-hot format and the original label from SVHN is a simple number, thus I converted original label to one-hot format by adding a new dimension to it which is implemented as follows.

```
d1,d2,d3,l1,l2,l3=load_data()  
grey_test=np.reshape(rgb2gray(d3),[-1,1024])  
grey_valid=np.reshape(rgb2gray(d2),[-1,1024])  
grey_train=np.reshape(rgb2gray(d1),[-1,1024])  
label_2=np.ones([10000,1])  
label_2[l1==1]=0  
train_labels=np.concatenate((l1,label_2),axis=1)
```

Fig 2.2 convert label to one-hot scale

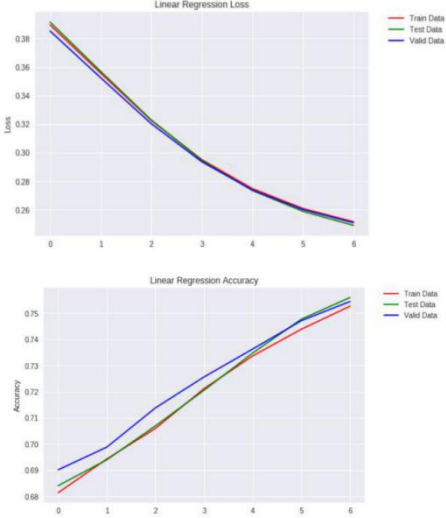
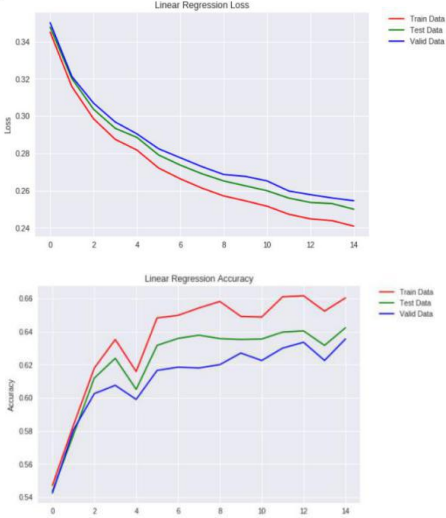
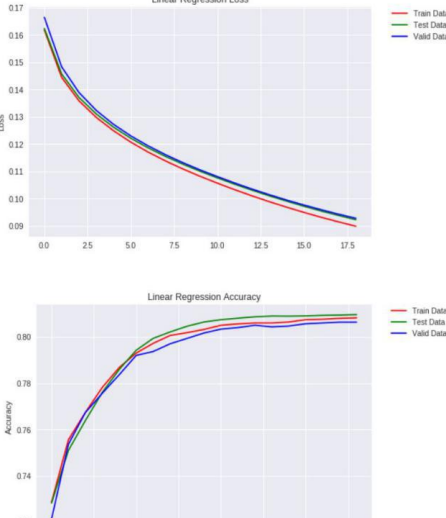
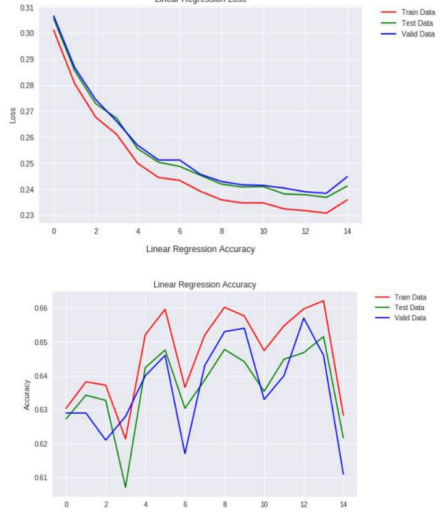
3. Linear Regression

In this section two classification problem was implemented, the first classification is between the number 1 and all the rest numbers and the second classification is between number 1 and 2, the goal of change of classification is to find which classification can provide higher prediction accuracy since the data that labeled as 0 is different in this case while number 1 is both labeled as 1 in these two cases.

In the case of loss function, I use the mean square error to compute the loss and use tensorflow adam optimizer to minimize the loss.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2.$$

Fig 3.1 mean square error formula

Label:1 and the rest	Label:1 and 2
 <p>The figure above is has a learning rate of 0.0001, 800epoch and batch size=2500 from this graph we can know that the model is at under-fitting phase because the loss have not converged yet and its accuracy is around 0.78.</p>	 <p>With learning rate=0.0001, 800 epoch and batch size=500, and above is the loss and accuracy curve, we can see that the accuracy is lower than the other label type and its accuracy is around 0.66.</p>
 <p>Increase the epoch from 800 to 2000 now the accuracy is converged and has an accuracy around 0.81.</p>	 <p>Decrease the batch size to 166 and we have a problem of over-fitting. Thus I suppose the small batches are more likely to have over-fitting problems.</p>

To test each hyper-parameter's influence on the model, I chose several key parameters like: learning rate, batch size, data set size, epoch number and tune them to see their effects on the model in terms of loss and accuracy. The form below is a summary of change of hyper-parameters' influence on model performance and the graphs for the loss and accuracy are put in the appendix.

	Variation	Comment
Learning Rate	from 0.001 to 0.0001	High learning rate tends to make the loss descend and converge quickly, but its loss and accuracy curve have more variance.
Data Set Size	6000 to 10000	No significant difference, in this case.
Batch Size	500 to 166	As the batch size decreases, the curve becomes less smooth and may have over fitting.
Epoch Number	800 to 2000	In this case, obviously the model is under fitting for 800 epochs because the loss and accuracy curve are both not converged, this means there should be enough runs to make sure the model is converged.

4. Convolutional Neural Network

This section describes how to use a convolutional neural network to compute the same problem of classification.

The structure of neural networks can vary from one to another, below is the graph describing the CNN that I implemented.

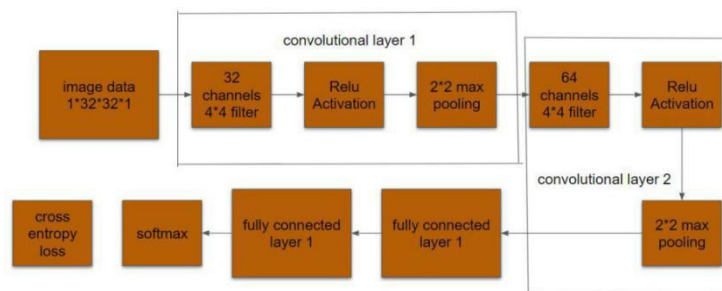


Fig 4.1 convolutional neural network structure

The CNN consists of 6 layers: one input layer, two convolutional layers, two fully connected layers, and an output layer. The first convolutional layer has 32 channels, a 4*4 filter, and uses ReLU as the activation function, followed by a 2*2 max pooling to shrink the data. Then the two fully connected layers compute the predictions, and the output layer computes the loss for the CNN. The loss function used for this neural network is cross entropy loss. Below is the loss and accuracy graph for learning rate=0.0001, epoch=5, batch size=50.

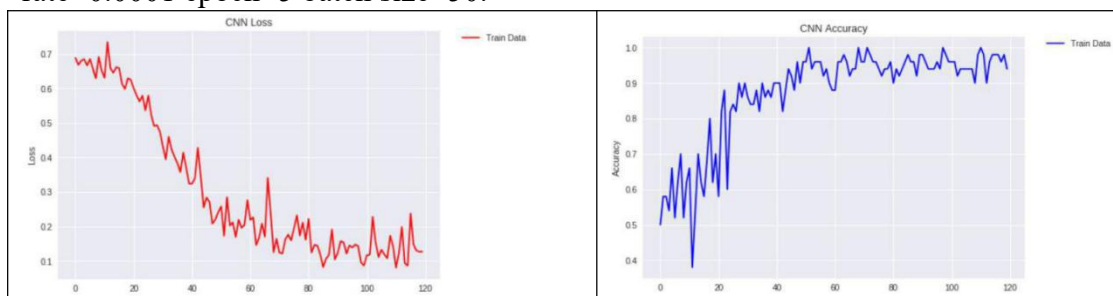


Fig 4.2 loss curve and accuracy curve for CNN

The rest of the training graphs are put in the appendix, from these graphs we can tell that as the dimension of the filter increases, the loss decreases more quickly and when

the dimension is decreased the loss also drop slower. In addition, the size of batches can effects the running speed, and the learning rate can also affects the outcome curve, a small learning rate makes it more easily for the model to converge but it also requires more runs before convergence, in addition, if the learning rate is set too big the loss may become 'nan'.

5. Support Vector Machine

As for SVM, there are two kinds of models, the linear type and non-linear type and I chose non-linear svm because the training data set has high dimensions and is very unlikely to be linear separable.

To downscale the dimension of the training data, I used the RBF (Radial Basis Function) kernel function, besides to testify whether the data set is not linear separable I also implemented the linear kernel and their kernel formula are shown below.

$$K(\vec{x}, \vec{z}) = e^{-\gamma(||\vec{x}-\vec{z}||)^2}$$

Fig 5.1 kernel formula of RBF

$$K(\vec{x}, \vec{z}) = \langle \vec{x}, \vec{z} \rangle$$

Fig.5.2 formula of linear kernel

$$L(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\vec{x}_i, \vec{x}_j)$$

Fig 5.3 loss function formula

To compute the accuracy of the training, I use the following prediction function to predict the corresponding outcome.

$$f(\vec{x}) = \sum_i \alpha_i y_i K(\vec{x}_i, \vec{x}) + b$$

Fig 5.4 prediction function

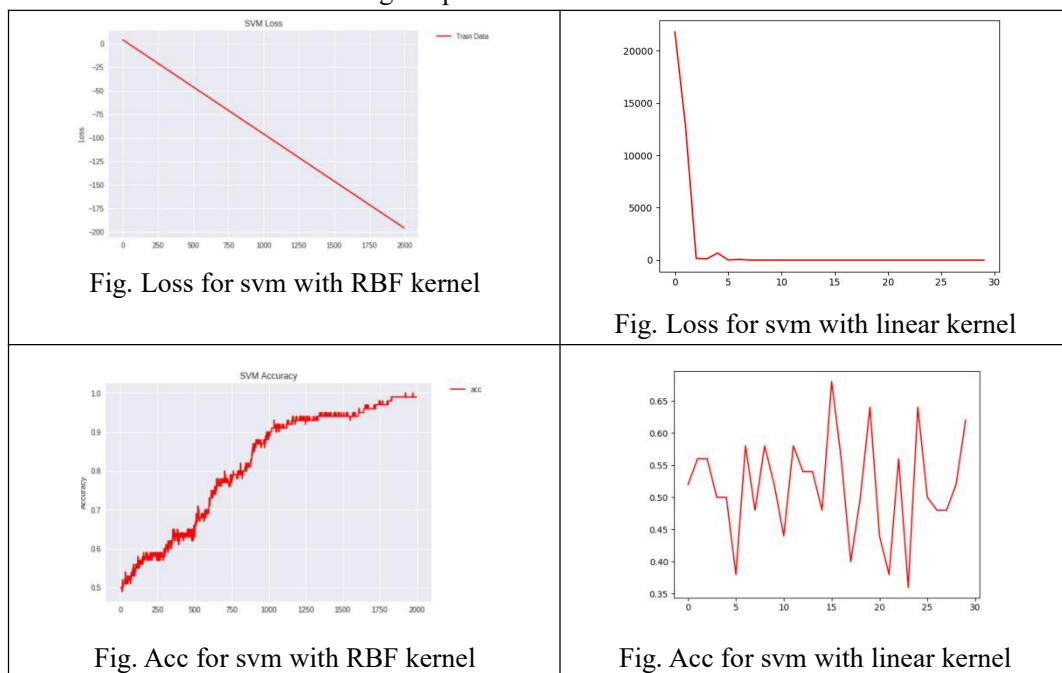


Fig5.5 Training loss and accuracy curve for RBF kernel and linear kernel

From the training outcomes we can see that loss computed by linear kernel is descending but the training accuracy is not increased, thus we can claim that this data set is not linear separable as we supposed. Also we can claim that the training outcomes of svm can be significantly influenced by the choice of kernel type, we should use RBF kernel when the data set is not linear separable.

6. Compare and Analyse

To find out which model is better for this classification problem, we need to compare a few key outcomes, the training accuracy , the training time and the runs needed before converge, which are shown in the following table.

	Accuracy	Time	Runs	Learning rate
Linear	0.67	62.84s	800 epochs	0.0001
CNN	0.95	108.72s	5 epochs	0.00001
SVM	0.99	18.98s	2000 epochs	0.001

Form 6.1 Training data properties

Compare the three models, we can tell that linear regression need less epochs to converge and it take less time than cnn in terms of running time, but its training accuracy is smaller than the other two. As for cnn, its prediction accuracy is almost as high as svm, and it needs only several epochs to converge which is significantly smaller than that of svm and linear regression, but due to its high model complexity, it also took the longest running time. SVM produces a slightly higher prediction accuracy than cnn and it also took less time, however it has the biggest learning rate and epoch number which means it is the model which is the hardest to converge.

During the testing, another thing that I find is the loss curve's trend is related to the size of the data, if the loss is computed by the training batch then the loss curve is not smoothly descending as the optimizer minimize the training loss, however if the loss is computed using the whole data set, the loss curve tends to be smooth and descent strictly. The figures below shows how the two curve differs from each other.

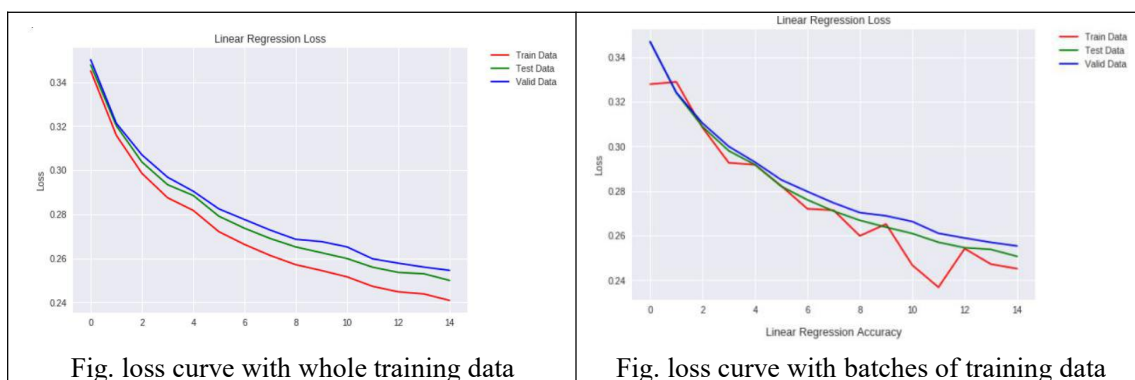


Fig. loss curve with whole training data

Fig. loss curve with batches of training data

In conclusion, in this case of classification problem the svm provides the highest prediction accuracy but it is the hardest one to converge and cnn provide the second highest accuracy and it needs least epochs to converge, linear regression provides the lowest accuracy but its faster than cnn and converges more easily than svm.

7. Appendix

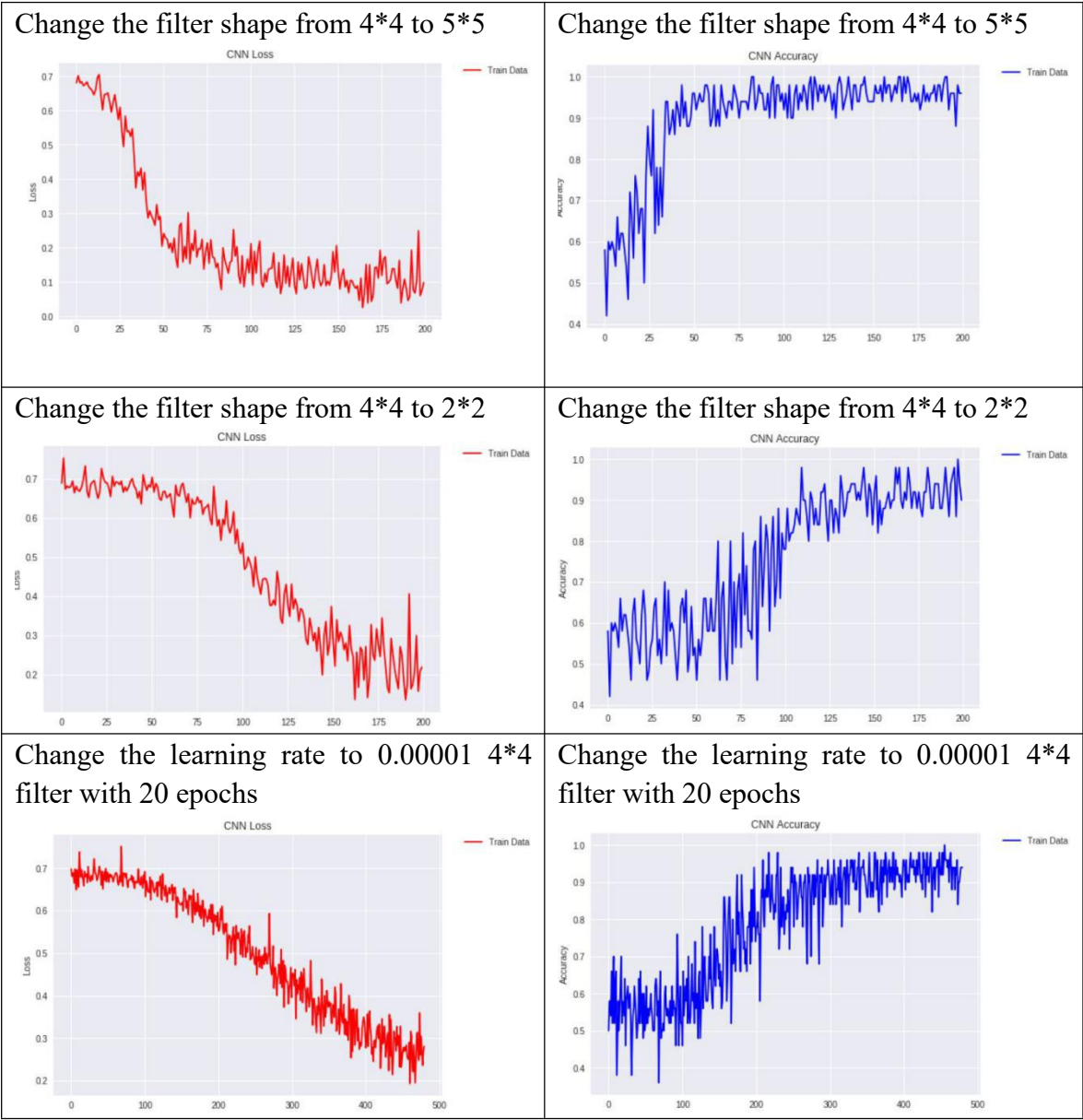


Fig 4.3 modifications on cnn

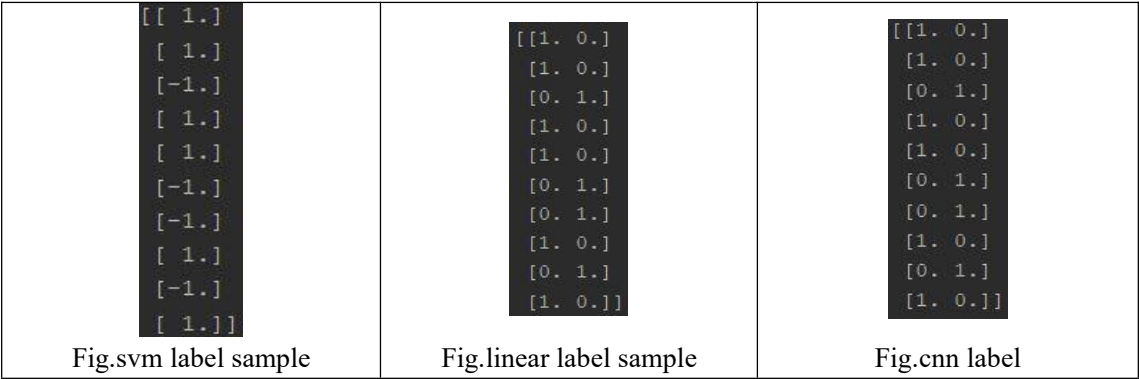


Fig 2.3 label samples