Application of Neural Network on Classification

Kuan-Wei Lee

Abstract

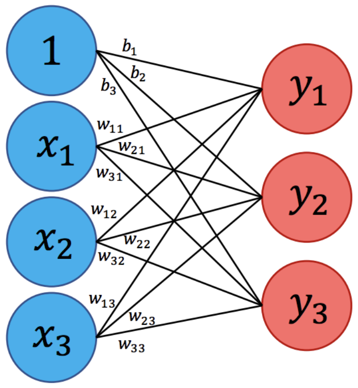
In this project, I am going to apply neural network on classifying data. I will try different setting on my neural network training algorithm to see which combination of the parameters give me the best result. The data I am trying to classify is the images of nine different items (T-shirt, trouser, pull-over, dress, coat, sandal, shirt, sneaker, bad, ankle boot). The training set contain 60000 images and the test set contain 10000 images. The goal is to use the 60000 images in the training set to train the neural network and test the result on the test set.

I. Introduction

Artificial neural network was inspired by the biological neural network. A good example of the biological neural network is our brain. Our brain contains billions of neurons firing, communicating with one another in every second to complete from tasks as simple as blinking our eyes to complex tasks like abstract thinking. Artificial neural network is basically imitating the connections of the neurons. If we feed data to artificial neural network and train it, it can 'evolve' and 'learn'. By constructing different complexity of the neural networks, it can achieve tasks that vary in difficulty. The application of the neural network that we concern in this project is data classification. I am going to construct a neural network and train it with some data so the network can perform data classification on unknown data.

II. Theoretical Background

The idea of artificial neural network is basically constructing multiple layers of 'neurons'. Each layer contains certain amount of neuron and each of them are connected to the neurons in the next layer. The connections between neurons contain certain amount of weight. When we feed some input data to the network, the input data will be weighted, summed, passed through an activation function, and fed to the next layer. The following is the visualization and the mathematical operation of the neural network:



$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \qquad A = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{pmatrix} \qquad b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \qquad y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

$y = \sigma(Ax + b)$ $\quad$ $\sigma(x)$ is the activation function

A common choice of $\sigma(x)$ is hyperbolic tangent $\tanh(x)$

After the input is weighted and pass through the activation function, the output will be 1 if the weighted input pass the threshold and will be 0 if it doesn't pass the threshold. The combination of 1s and 0s of the output will be used to classify the input data.

III. Algorithm Implementation and Development

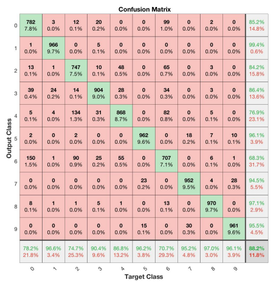
In this project, I utilize the neural network toolbox in MATLAB. I first reshape the data to the desirable dimension. Then I construct a neural network by specifying some parameters such as learning rate and optimizer. I change amount of layers and try different combination of the parameters to see if the accuracy is improved.

IV. Computation Results

After trying a few different combination of the parameters, I was only able to get an accuracy up to 88.2%. The following is the settings of my neural network:

```
options = trainingOptions('adam', ...
    'MaxEpochs',7,...
    'InitialLearnRate',1e-4, ...
    'L2Regularization',1e-5, ...
    'ValidationData',{X_valid,y_valid}, ...
    'GradientDecayFactor',0.9,...
    'ValidationFrequency',40,...
    'Verbose',false, ...
    'Plots','training-progress');
```

I used adam optimizer, set initial learn rate and L2 regularization to 10^-4. I also specify the gradient decay factor to be 0.9 and validation frequency to be 40. I have one input layers 6 six hidden layers and 1 output layer in my neural network. Each layer contains more than 100 neurons. With many different attempts, 88.2% accuracy was the best result I achieved. Adjusting all the parameters doesn't seem to affect the accuracy that much. One thing I noticed in the confusion matrix(Figure 1) is that there are 150 cases that the neural network predicted as T-shirt are actually shirt. These two kinds of images look very close to each other. Therefore, it is not a surprise that the neural network confuses these two categories. The images I am trying to classify are very complicated images. Therefore, it is hard to achieve high accuracy. To achieve a higher accuracy, we might need different type of neural network.



Confusion Matrix

Output Class \ Target Class	0	1	2	3	4	5	6	7	8	9	
0	782 7.8%	3 0.0%	12 0.1%	20 0.2%	0 0.0%	0 0.0%	99 1.0%	0 0.0%	2 0.0%	0 0.0%	85.2% 14.8%
1	1 0.0%	966 9.7%	0 0.0%	5 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	99.4% 0.6%
2	13 0.1%	1 0.0%	747 7.5%	10 0.1%	48 0.5%	0 0.0%	65 0.7%	0 0.0%	3 0.0%	0 0.0%	84.2% 15.8%
3	39 0.4%	24 0.2%	14 0.1%	904 9.0%	28 0.3%	0 0.0%	34 0.3%	0 0.0%	3 0.0%	0 0.0%	86.4% 13.6%
4	5 0.1%	4 0.0%	134 1.3%	31 0.3%	868 8.7%	0 0.0%	82 0.8%	0 0.0%	5 0.1%	0 0.0%	76.9% 23.1%
5	2 0.0%	0 0.0%	2 0.0%	0 0.0%	0 0.0%	962 9.6%	0 0.0%	18 0.2%	7 0.1%	10 0.1%	96.1% 3.9%
6	150 1.5%	1 0.0%	90 0.9%	25 0.2%	55 0.5%	0 0.0%	707 7.1%	0 0.0%	6 0.1%	1 0.0%	68.3% 31.7%
7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	23 0.2%	0 0.0%	952 9.5%	4 0.0%	28 0.3%	94.5% 5.5%
8	8 0.1%	1 0.0%	1 0.0%	5 0.1%	1 0.0%	0 0.0%	13 0.1%	0 0.0%	970 9.7%	0 0.0%	97.1% 2.9%
9	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	15 0.1%	0 0.0%	30 0.3%	0 0.0%	961 9.6%	95.5% 4.5%
	78.2% 21.8%	96.6% 3.4%	74.7% 25.3%	90.4% 9.6%	86.8% 13.2%	96.2% 3.8%	70.7% 29.3%	95.2% 4.8%	97.0% 3.0%	96.1% 3.9%	88.2% 11.8%

Figure 1: This is the confusion matrix of the result predicted by the neural network versus the true result. Labels of the categories are: 0=T-shirt, 1=Trouser, 2=Pullover, 3=Dress, 4=Coat, 5=Sandal, 6=Shirt, 7=Sneaker, 8=Bag, 9= Ankle boots

In the second part of the project, I implement a convolution neural network to do the image classification. A convolution net works basically break the images into multiple regions and apply filters to each region to pick out certain features. By specifying different numbers of convolution layers and different size of the filter, we may be able to enhance the performance of our neural network.

The type of neural network I implement is called "LeNet-5". The architecture of LeNet-5 is shown in Figure 2. I used the overall structure of LeNet-5 but changed some parameters such as numbers of filters in each convolution layer, types of the pooling layer. I set the filter size in each convolution layer to 2×2. The first convolution layer contains 10 filters , the second one contain 22 filters, the the third convolution layer contains 130 filters. I changed the type of pooling layer from "average", to max. In the end the accuracy of the prediction made by the neural network is 89.9%. There are some improvements compared to the neural network I implemented in the first part. However, it is still hard to pass the accuracy of 90%. In the confusion matrix, we can see that the convolution neural network still confuses between T-shirt and shirt a lot (Figure 3).

V. Conclusion and Summary

In this project, we have seen the application of neural network on image classification. Although the performance of the two neural networks I implemented was not very high due to the complexity of the problem, the power of neural network on solving classification problems is still obvious. With more complex and well constructed neural network, I am confident that the accuracy of the prediction can pass 95%. The challenge in this particular project is that the images of T-shirt and shirt are very similar to each other and is hard for the neural network to tell them apart. To solve the problem, we might need more convolution layer or other types of layers besides convolution layers and pooling layers.

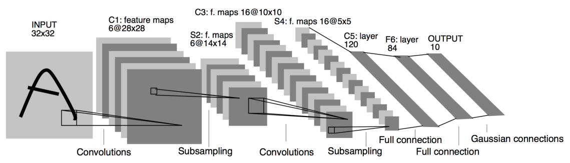
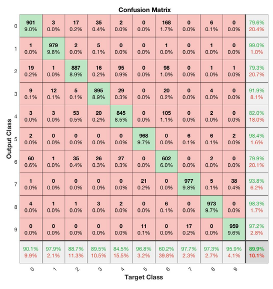


Figure 2: This is the architecture of LeNet-5



Confusion Matrix

Output Class

0	901 9.0%	3 0.0%	17 0.2%	35 0.4%	2 0.0%	0 0.0%	168 1.7%	0 0.0%	6 0.1%	0 0.0%	79.6% 20.4%
1	1 0.0%	979 9.8%	2 0.0%	5 0.1%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	1 0.0%	0 0.0%	99.0% 1.0%
2	19 0.2%	1 0.0%	887 8.9%	16 0.2%	95 0.9%	0 0.0%	98 1.0%	0 0.0%	1 0.0%	1 0.0%	79.3% 20.7%
3	9 0.1%	12 0.1%	5 0.1%	895 8.9%	29 0.3%	0 0.0%	20 0.2%	0 0.0%	4 0.0%	0 0.0%	91.9% 8.1%
4	3 0.0%	3 0.0%	53 0.5%	20 0.2%	845 8.5%	0 0.0%	105 1.1%	0 0.0%	2 0.0%	0 0.0%	82.0% 18.0%
5	2 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	968 9.7%	0 0.0%	6 0.1%	6 0.1%	2 0.0%	98.4% 1.6%
6	60 0.6%	1 0.0%	35 0.4%	26 0.3%	27 0.3%	0 0.0%	602 6.0%	0 0.0%	2 0.0%	0 0.0%	79.9% 20.1%
7	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	21 0.2%	0 0.0%	977 9.8%	5 0.1%	38 0.4%	93.8% 6.2%
8	4 0.0%	1 0.0%	1 0.0%	3 0.0%	2 0.0%	0 0.0%	6 0.1%	0 0.0%	973 9.7%	0 0.0%	98.3% 1.7%
9	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	11 0.1%	0 0.0%	17 0.2%	0 0.0%	959 9.6%	97.2% 2.8%
	90.1% 9.9%	97.9% 2.1%	88.7% 11.3%	89.5% 10.5%	84.5% 15.5%	96.8% 3.2%	60.2% 39.8%	97.7% 2.3%	97.3% 2.7%	95.9% 4.1%	89.9% 10.1%
	0	1	2	3	4	5	6	7	8	9	

Target Class

Figure 3: This is the confusion matrix of the result predicted by the convolution neural network versus the true result.

Appendix A.

```
layer = reluLayer;
layers = [imageInputLayer([28 28 1])
          fullyConnectedLayer(700)
          layer
          fullyConnectedLayer(600)
          layer
          fullyConnectedLayer(500)
          layer
          fullyConnectedLayer(400)
          layer
          fullyConnectedLayer(300)
          layer
          fullyConnectedLayer(100)
          layer
          fullyConnectedLayer(10)
          softmaxLayer
          classificationLayer];
```

The Matlab code above is the layers of the neural network I implemented in the first part. It has seven fully connected layers and the numbers in the parenthesis are the numbers of neurons in each layer.

```
net = trainNetwork(X_train, y_train, layers, options);
```

The code above is for training the neural network. It takes training data, neural network layers, and training options as input and output the trained neural network.

```
layers = [
     imageInputLayer([28 28 1],"Name","imageinput")
     convolution2dLayer([2 2],10,"Name","conv_1","Padding","same")
     tanhLayer("Name","tanh_1")
     maxPooling2dLayer([2 2],"Name","maxpool_1","Padding","same")
     convolution2dLayer([2 2],22,"Name","conv_2")
     tanhLayer("Name","tanh_2")
     maxPooling2dLayer([5 5],"Name","maxpool_2","Padding","same")
     convolution2dLayer([2 2],130,"Name","conv_3")
     tanhLayer("Name","tanh_3")
     fullyConnectedLayer(90,"Name","fc_1")
     tanhLayer("Name","tanh_4")
     fullyConnectedLayer(10,"Name","fc_2")
     softmaxLayer("Name","softmax")
     classificationLayer("Name","classoutput")];
```

The code above is the convolution neural network layers exported from the deep network designer of Matlab.

Appendix B.

```
clear all
close all
clc

load('fashion_mnist.mat')

X_train = im2double(X_train);
X_test = im2double(X_test);

X_train = reshape(X_train,[60000 28 28 1]);
X_train = permute(X_train,[2 3 4 1]);

X_test = reshape(X_test,[10000 28 28 1]);
X_test = permute(X_test,[2 3 4 1]);

X_valid = X_train(:,:,:,1:5000);
X_train = X_train(:,:,:,5001:end);

y_valid = categorical(y_train(1:5000))';
y_train = categorical(y_train(5001:end))';
y_test = categorical(y_test)';
layer = reluLayer;
layers = [imageInputLayer([28 28 1])
        fullyConnectedLayer(700)
        layer
        fullyConnectedLayer(600)
        layer
        fullyConnectedLayer(500)
        layer
        fullyConnectedLayer(400)
        layer
        fullyConnectedLayer(300)
        layer
        fullyConnectedLayer(100)
        layer
        fullyConnectedLayer(10)
        softmaxLayer
        classificationLayer];

options = trainingOptions('adam', ...
    'MaxEpochs',7,...
    'InitialLearnRate',1e-4, ...
    'L2Regularization',1e-5, ...
    'ValidationData',{X_valid,y_valid}, ...
    'GradientDecayFactor',0.9,...
    'ValidationFrequency',40,...
    'Verbose',false, ...
    'Plots','training-progress');

net = trainNetwork(X_train,y_train,layers,options);

y_pred = classify(net,X_test);

plotconfusion(y_test,y_pred)

%%
clear all
close all
clc

load('fashion_mnist.mat')

X_train = im2double(X_train);
X_test = im2double(X_test);

X_train = reshape(X_train,[60000 28 28 1]);
X_train = permute(X_train,[2 3 4 1]);

X_test = reshape(X_test,[10000 28 28 1]);
X_test = permute(X_test,[2 3 4 1]);

X_valid = X_train(:,:,:,1:5000);
X_train = X_train(:,:,:,5001:end);

y_valid = categorical(y_train(1:5000))';
y_train = categorical(y_train(5001:end))';
y_test = categorical(y_test)';

layers = [
    imageInputLayer([28 28 1],"Name","imageinput")
    convolution2dLayer([2 2],10,"Name","conv_1","Padding","same")
    tanhLayer("Name","tanh_1")
    maxPooling2dLayer([2 2],"Name","maxpool_1","Padding","same")
    convolution2dLayer([2 2],22,"Name","conv_2")
    tanhLayer("Name","tanh_2")
    maxPooling2dLayer([5 5],"Name","maxpool_2","Padding","same")
    convolution2dLayer([2 2],130,"Name","conv_3")
    tanhLayer("Name","tanh_3")
    fullyConnectedLayer(90,"Name","fc_1")
    tanhLayer("Name","tanh_4")
    fullyConnectedLayer(10,"Name","fc_2")
    softmaxLayer("Name","softmax")
    classificationLayer("Name","classoutput")];

options = trainingOptions('adam', ...
    'MaxEpochs',5,...
    'InitialLearnRate',1e-3, ...
    'L2Regularization',1e-4, ...
    'ValidationData',{X_valid,y_valid}, ...
    'Verbose',false, ...
    'Plots','training-progress');

net = trainNetwork(X_train,y_train,layers,options);

y_pred = classify(net,X_test);

plotconfusion(y_test,y_pred)
```