Artificial Neural Networks vs. Support Vector Machines for Image Classification

Joshua Frankie B. Rayo, Edward Nataniel C. Apostol
CS 180 THW

1 Summary

This paper shows the implementation of image classification using the two most important discriminative models: ANN and SVM. Training and testing data are made possible by using ANN and LibSVM libraries, respectively. This paper also shows the accuracy of these two discriminative models.

2 Steps in training and testing data

For every subdirectory of 'faces' directory, we removed all the small image files (those filenames that contains _2 or _4 in it). The original images are used for training and testing data. For the training data, we used only 24 .PGM images per class. So how do we form the feature vector? We studied the PGM image specification at *http://netpbm.sourceforge.net/doc/pgm.html*. When a PGM image is opened in a plain text editor, we observed the similar pattern:

```
P2
24 7
15
0 0 0 0 0 0 0 0 ...
0 3 3 3 3 0 0 7 ...
0 3 0 0 0 0 0 7 ...
0 3 3 3 0 0 0 7 ...
0 3 0 0 0 0 0 7 ...
0 3 0 0 0 0 0 7 ...
0 0 0 0 0 0 0 0 ...
```

The first three rows of the image is only a metadata. Since we are only concerned about the pixels of the image, we can omit them. We form the feature vector row-wise.

2.1 ANN

We trained and tested ANN using the code provided in a 32-bit Ubuntu operating system. We tried to vary the parameters to see its overall effect on the accuracy. Attached is the raw data obtained from the training and testing.

2.2 SVM

Given the configuration above, its feature vector to be feed in ANN is $F = [< class\_no. > 1 : 0\, 2 : 0 \ldots 25 : 0\, 26 : 3 \ldots]$. The remaining images not trained are used for testing. We gathered the images to be trained, obtained an SVM model for it and use the model to test the images to be tested.

3 Results and Discussions

3.1 ANN

3.1.1 Varying the parameters

1. Epoch

   By setting the number of hidden nodes to 200 and the learning rate to 0.1, here is the graph of the computed train_error and test_error with epoch=1000.

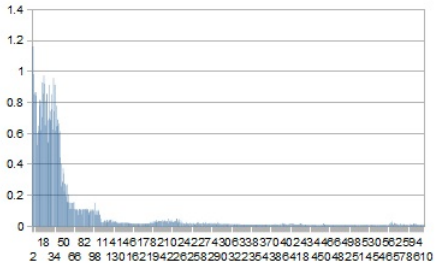

Figure 1: Graph of epoch vs. train_error

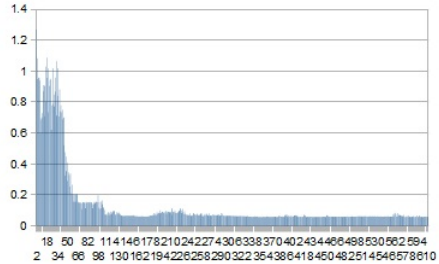

Figure 2: Graph of epoch vs. test_error

It can be observed in the graph that as the number of epoch increases, less sum of squares of errors are produced in the test and training instances.

By superimposing the 2 graphs (Figure 3), it can be observed that the test error, or the expected error over an independent test sample, is generally higher than the training error or the average loss over the training data (cc.gatech.edu)

2. No. of hidden nodes

   By setting the learning rate to 0.3 and momentum to 0.3, we picked 11 different values of the no. of hidden nodes. Here is the graph of the computed train_error and test_error after epoch=100

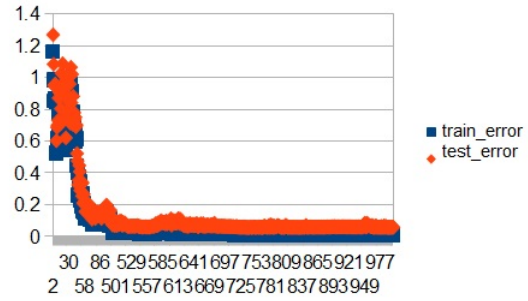

Figure 3: epoch vs. train_error and test_error

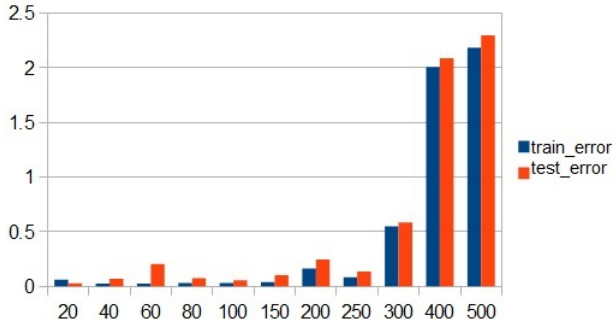

Figure 4: No. of hidden nodes vs. train_error and test_error

From the graph, 100 is the best number of hidden nodes since both the train_error and the test_error are very low. Very small number of hidden neurons might cause underfitting, that is when the neural network has limited capacity to learn the complicated data set. Also, note that increasing the number of hidden nodes may cause the errors to increase. Overfitting happens whenever the artificial neural network has large processing capacity that the number of information in the training set is too small to train all the neurons in the hidden layer (heatonresearch.com). In the graph, it can be seen that choosing more than 300 hidden nodes may cause very large errors.

Changing the values of the hidden nodes affect the training time. Larger number of nodes results in longer training time.

3. Learning Rate and Momentum

Momentum adds a factor of the previous obtained weight to the current weight. Its' value may range from 0 to 1. Using some constant values for the learning rate, number of hidden nodes, and epoch, here is the graph of the computed train_error and test_error using different momentum values.

A very small or very large momentum generated a lot of errors. 0.25, 0.5 and 0.75 are good choices for the momentum. Changing the momentum also affected the training time. Higher momentum resulted in longer training time.

Learning rate affects the speed of learning. Training using small learning rates resulted in very long training time. But having small learning rates increases

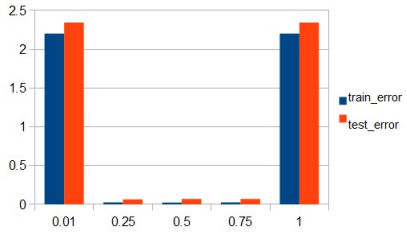

Figure 5: Momentum vs. train_error and test_error

the accuracy.

ANN is more prone to overfitting. Some parameters must be chosen so that it is not too small nor too large for the given problem.

3.2 SVM

For a fixed kernel but different parameters, the accuracy of classification changes. Here is the result from different kernels used in SVM:

Kernel	Param	Correct	Accuracy
linear		175/180	97.2222%
polynomial	d=3	175/180	97.2222%
	d=6	174/180	96.6667%
	d=10	175/180	97.2222%
RBF	g=1/(128*120)	53/180	29.4444%
	g=0.1	9/180	5%
	g=0.25	9/180	5%
	g=0.5	9/180	5%
	g=0.75	9/180	5%
	g=1.0	9/180	5%
sigmoid	g=1/(128*120)	9/180	5%
	g=0.1	9/180	5%
	g=0.25	9/180	5%
	g=0.5	9/180	5%
	g=0.75	9/180	5%
	g=1.0	9/180	5%

For the SVM, when the training data is used for testing data, there is no discrepancy, accuracy is 100%. Since the images to be tested is not in training data, there is a discrepancy in the test accuracy. As the table shows, accuracy ranges from 5% to 97

It is a very good idea to treat each pixel as a feature, because each person has different facial features. It is very important to notice the person's difference with each other. Another good thing is, even if that person uses sunglasses or change his facial expression, the SVM can nearly recognize the face of that person. For the sigmoid kernel, which appear to be the worst kernel, it does not seem to have a change. We can say that the linear function is the best SVM kernel for image classification (for PGM files). SVM trained faster than ANN specially if the parameters of ANN are not optimized. ANN tests faster.