ANN3

Irene Volpe
r0740784

Exercise 1: Principal Component Analysis

Principal component analysis on Handwritten Digits

Principal Component Analysis (PCA) projects a set of high-dimensional vectors into lower dimensionality while preserving most of the information. The quality of projection can be done by doing a reconstruction of this projection. PCA is applied to a dataset containing 500 images of size 16x16 pixels and reshaped to be as a single 256-dimensional vector images of handwritten digit '3'.

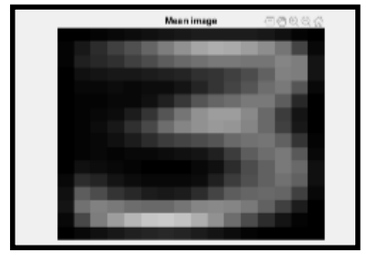

fig 1

Compute the mean 3 and display it

Centering the data on origin is the first step of PCA reduction that is done by subtracting mean of each image from all other images. Figure 1 shows the mean of the dataset i.e. image of digit 3.

Compute thecovariance matrix of the whole dataset of 3s, the eigenvectors of this covariance matrix and plot the eigenvalues.

The covariance matrix of the dimensions can now be computed which tells us whether two dimensions/data points increase or decrease together or are inversely proportional to one another. From this we calculate the eigenvectors and the eigenvalues. Eigenvalue is the sums of squares of the distances for the best fit-line for principle components. PCA components are eigenvectors with highest eigenvalues.

We select eigenvectors corresponding to the highest eigenvalues to project the original image in a lower dimensionality with as minimum loss as possible. These selected eigenvectors contain the most information of the original image. Figure 2 shows some reconstructions with the selected 'K' number of eigenvectors with highest eigenvalues.

fig 2

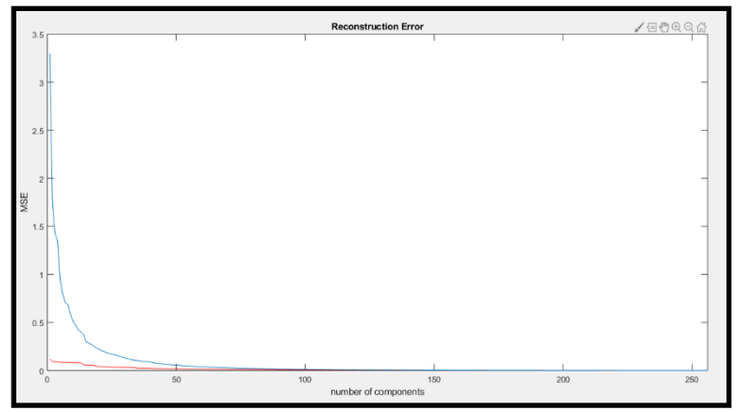

fig 3

What should the reconstruction error be if k = 256? What is it if you actually try it? Why?

The first selected eigenvector gives the most information of the original image as the reconstructed image looks the closest to the original image. As we select more and more eigenvectors the reconstruction of the original image keeps getting better. Even when using all the eigenvectors it is still not possible to get a reconstruction error of zero because the mean image added to the reconstructed image makes the projection to be slightly different than the original image. Overall, projection is performed very well. Figure 3 shows a comparison of the reconstruction error (taken as the Mean Squared Error) with the increasing number of eigenvectors.

Reconstructs the dataset and measures the reconstruction error.

It can be seen in Figure 3 that the reconstruction error decreases as the number of eigenvectors increase. After k=50 even though there are only a few eigenvectors there is a sharp decline in the MSE, going down to less than 0.02. This is followed by a low gradient slope that becomes almost a straight line by k=256. So, even with 20 highest eigenvectors we can almost perfectly reconstruct the handwritten images, thus reducing the dimensionality from 256 to just 20.

Exercise 2: Stacked Autoencoders

Several AEs are stacked together to form an SAE that has an input layer representing the original data and several hidden layers. The output values in the hidden layers represent the transformed features that can reconstruct the original data using the decoder. Stacked consist of multiple layers of sparse autoencoders such that the output of one layer becomes the input of the next layer. It works in an unsupervised mode of learning and apply backpropagation for accurate results. In order to obtain good classification results, each layer in the network can run separately at first to obtain weights and biases. All these layers with the obtained parameters are stacked and an initial performance score is obtained by using these parameters. Fine-tuning is then performed by using backpropagation to get a better final result.

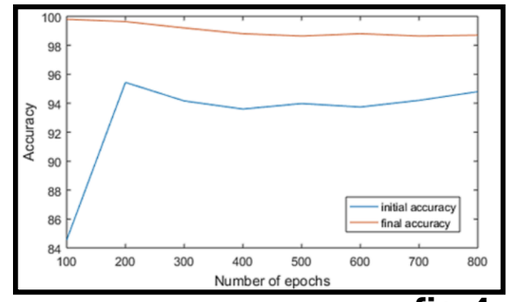

fig 4

Are you able to obtain better results?

In this section we examine the performance of classification of images of handwritten digits by using different parameters of stacked autoencoders (SAE).

Three different settings are tested, for a stacked autoencoders with two hidden layers:

1) Maximum number of epochs: It can be seen (Figure 4) that the best final performance of 99.8% was achieved at epochs=100 even though the initial performance at the same epochs was 84.5%.

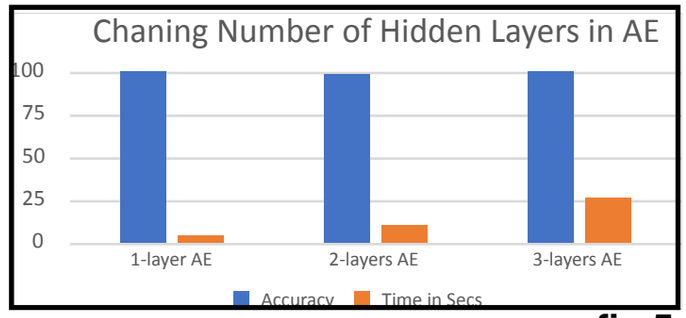

fig 5

Finetuning

In the further experiments we fix the number of epochs to 100. The improved performance is due to fine-tuning the stacked network. After all the weights have been initialized, the whole encoder-decoder network is treated as a multilayer feed-forward network and the weights are fine-tuned via the backpropagation algorithm i.e. minimizing the reconstruction error.

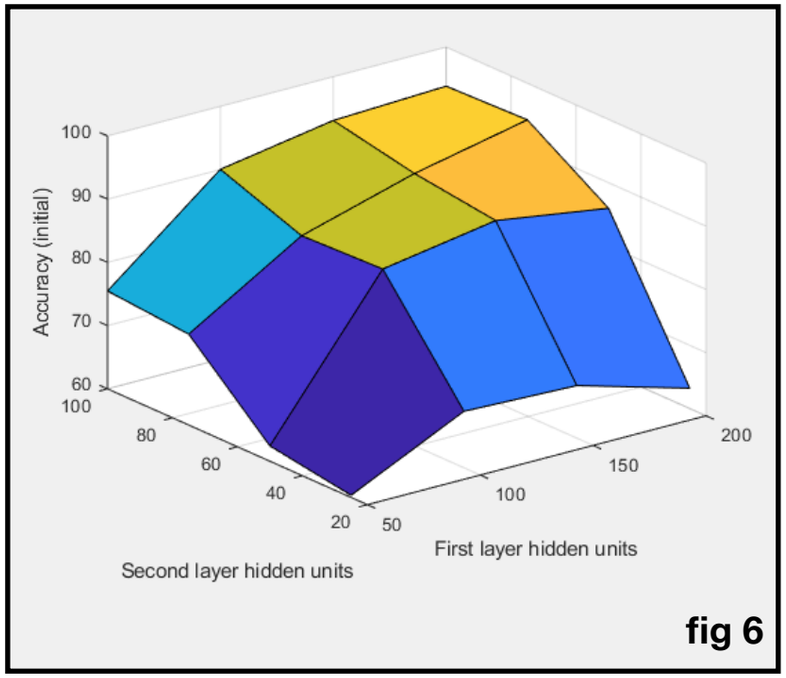

fig 6

Number of layers nedded to achieve a better performance than with a neural network

Size of hidden layers: Different combination of units in the two hidden layers [50,25], [100,50], [150,75], [200,100];

Figure 6 shows the initial accuracies while the Figure 7 shows the final accuracies achieved by our various combinations. With the combination of [150, 25], [200, 50], and [200, 75] number of neurons in hidden layers, we achieve an accuracy of 99.9%. The accuracy of reconstructions of the training set (less noisy) before the fine-tuning stage (backpropagation training) improves significantly by the application of the backpropagation algorithm. SAEs transform the input into a low or a high dimensional code through the encoder function, and then reconstructs the original data through the decoder function. The optimal reconstruction of the input is obtained by minimizing the reconstruction error in this stage using back propagation. Set of optimal parameters i.e. the weight matrices and biases of the encoder and the decoder and number of hidden layers in the SAE, required for best reconstruction of original images are obtained by back propagation.

Number of layers: Stacked autoencoders with one ([150 neurons]), two [150,75] and three [150 75 25] hidden layers are tested. Accuracy and computation time of the AEs with different number of layers is shown in Figure 5. We can see that although there isn't much difference in accuracy of the results, the computation time increases by adding more hidden layers. Using more layers can give better results but it can also lead to sparsity in the hidden layers causing drop in accuracy and it is computationally more expensive.

SAEs Vs Multilayer Neural Network: Compared to a normal multiple layer neural network, a stacked autoencoder architecture allows for low computation cost and the amount of training data. The representations generated by a stacked autoencoder are relatively robust and useful compared to a normal neural network.

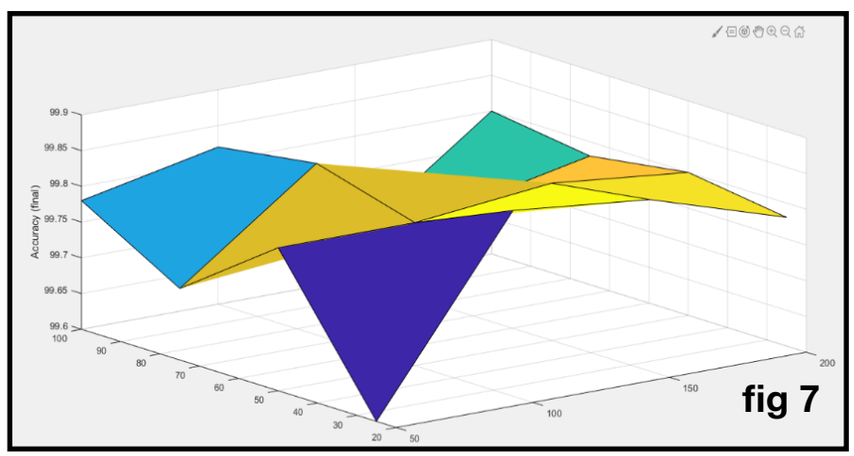

fig 7

Exercise 3: Concolutional neural networks

Convolutional Neural Networks are type of deep neural networks that consists of Convolution, ReLU, Pooling, Fully Connected layers stacked together in addition to the input and the output layer. CNNs can have different architectures with different combination of these layers in terms of number of each type of layer and their order in the network. CNNs layers have two types of parameters; one that are learned automatically by the CNN through backpropagation like feature pixels, voting weights and some tunable hyperparameters that the designer chooses like; number and size of features (convolution layer), window stride and size (pooling layer) and number of neurons (fully connected layer).

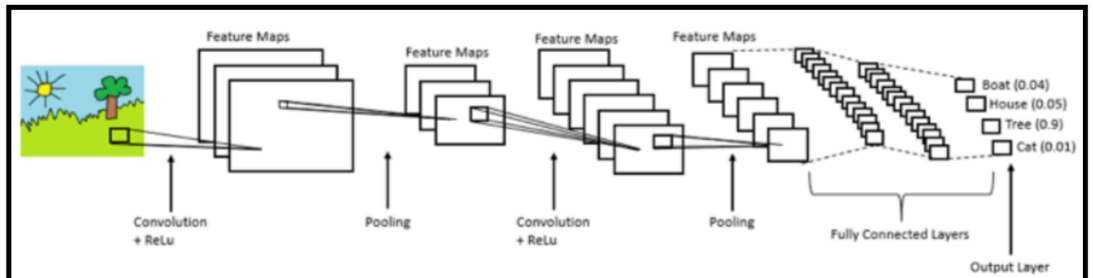

Figure 10 : Complete CNN architecture

fig 8

The CNN is tested for effect of different number of convolution, pooling and fully connected layers and dimensions of weights on accuracy of results of classification and time required for training. 28*28-pixel images of digits 0-9 are the inputs.

Number of Convolution Layers

It is observed that adding convolution layers followed by ReLU layers and a pooling, softmax, classification layer increases the training time required and thus are more computationally expensive. However, the performance of network in terms of percentage of correct classifications can decrease with addition of convolution layers.

1 Convolution Layer: Using one convolution layer with filter size 5*5 and 20 neurons, an accuracy of 98.80 % correct classification is achieved in 114.25 seconds - 15 epochs 870 iterations.

2 Convolution Layer With two convolution layers having 12 and 20 neurons and using same filter size 5*5 as before, there isn't much effect on accuracy (97.48%) but the training time is increased to 128.50 seconds with almost same number of iterations and epochs.

3 Convolution Layer: Using same filter size and 12, 20 and 50 neurons in layer 1-3 respectively accuracy is 93.88% in 238.12 seconds.

Number of Fully Connected Layers: With only one convolution layer and two fully connected layers between the max-pooling and the softmax layers decreases the accuracy of results (from 98.80% to 85.92%) but does not affect the training time much (from 114.25 to 84.61 seconds).

what weights represent

The weights of the convolutional layer in our first network are [5, 5, 1, 20] where (5, 5) represent the filter size, (1) the number of channels and the last number (20) is the number of filters.

dimension of the input at the start of layer 6 and why

In layers 1 to 5 there are two convolutional layers and a max-pooling layer which affect the size of the input to layer 6.
The first convolutional layer has 12 filters of size 5x5, so its output would be 12 filtered images of size 24x24 (the original images were of size 28x28).
After passing through the ReLU layer, these filtered images come into the max-pooling layer.
This layer uses a 2x2 filter with a stride of [2 2], therefore the output from this layer would be 12 filtered images of size 12x12.
Then comes the second convolutional layer. This layer has 24 filters of size 5x5, so its output would be 12*24 images of size 8x8. Hence the dimensionality of input of layer 6 is 12x24x8x8.

final dimension of the problem

The dimensions of the fully connected layer on the network would be 18,432. This is because, the outputs from the preceding ReLU layer (the 6th layer) would be vectorized, and as there a 12x24 filtered images coming from the previous layers, this vector stack size would be of 12*24*8*8 dimensions.

final vs initial dimension

The initial dimensions were of size 784 (28*28).