**ROCHESTER INSTITUTE OF TECHNOLOGY**

**DEPARTMENT OF COMPUTER ENGINEERING**

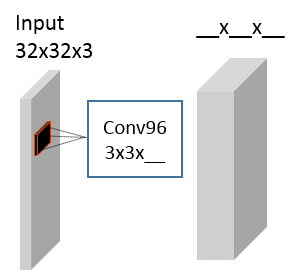
**CMPE 677 Machine Intelligence**

**HW #8  
  
Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

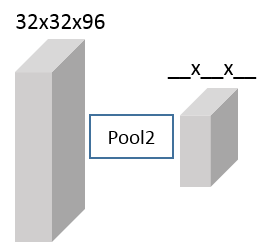
**Due Dec 12th, 11:55pm. Submit via Dropbox. Work alone. All questions pertain to Matlab/Octave.**

**Please note for this homework, especially for CNN related question, it is advisable to have the latest version of Matlab. If you have older releases and if you encounter errors in the given code, that might be because of incompatibility with the Matlab version. Matlab versions - R2016b, R2017a, R2017b should work fine.**

1. (4 pts) Given below is the input and output to a convolutional layer of a convolutional neural network. The input to this particular layer is a RGB image of dimensions 32×32×3. The layer has 96 filters each of spatial extent 3×3. The filters have a stride of 1 and padding of size 1 to the left, right, top, and bottom. Fill in the blanks in the below figure.



1. (3 pts) Given below is the input and output to a pooling layer of a convolutional neural network. The input to this particular layer has dimensions 32×32×96. The pooling layer has filters of size 2×2 and stride of 2. Determine the dimensions of the output of this pooling layer.



1. We will now explore and learn to use Neural Network Toolbox (by mathworks) for coding a CNN architecture. For this, you must install Neural Network Toolbox.

**Step-1** Follow the instructions below to download and install Neural Network Toolbox.

Note: Neural Network Toolbox release works best for R2016a and above. Preferably, try to work on latest version.

1. Go to the App tab in Matlab
2. Click on Get More Apps
3. Search for Neural Network Toolbox
4. Click on Install
5. To check if you have successfully installed Neural Network Toolbox

Run the following commands and check-

>> ver

**Step-2a (6 points)** In this part we will introduce Neural Network Toolbox while using the same MNIST dataset from the previous homework.

We have created a simple CNN architecture in the file ‘cnn\_architecture.m’ and the train code is in ‘mnist\_cnn\_cmpe677hw8\_nnt.m’. Read both files and make sure you understand various terms used and answer the following questions.

1. **How many activation layers are present in this network?**
2. **How many Pooling layers are present in this network?**
3. **What are the filter dimensions and number of filters in the first Convolutional layer? \_\_\_\_\_, \_\_**
4. **What is the size of the image after applying the Convolutional layer?**
5. **What is the batch size we are using in this model?**
6. **What is the learning rate value used in this model?**

Download the data ‘ex4data1.mat’ from the Dropbox. We will keep all files and data for this homework in one directory. Copy the downloaded data and files in this folder. This dataset contains 5000 samples from MNIST digit database where each sample is a grayscale image of dimensions 20×20. Execute the mnist\_cnn\_cmpe677hw8\_nnt.m file.

By look at the info struct and understand the information provided by the trained model.

1. **State and describe the info structure.**

You should get the Accuracy of **0.6610**

(You would get these values after you run the given code successfully) (Note: Different versions of laptop may give you slightly different values.

**Step-2b (10 pts)** Copy ‘cnn\_architecture.m’ to ‘cnn\_architecture1.m’. Modify ‘cnn\_architecture1.m’ such that the convolutional layer has 64 filters, epochs = 10 and batch size = 20. DO NOT make any other modifications to the architecture to obtain the correct values. Note that you keep all your files in one folder and check the path.

Update the following line in mnist\_cnn\_cmpe677hw8\_nnt.m w to call cnn\_architecture1.m

[layers, options] = cnn\_architecture(opts);

**What is the accuracy? Show your code.**

**Step-3a (10 pts) Note: This takes 20min – 1hr to run.**

In this step we will train a CNN for the CIFAR-10 dataset. Aftercopying the Matlab files from the Dropbox into the working directory, run the script cifar\_cnn\_cmpe677hw8\_nnt.m to train a CNN for CIFAR-10 image classification.

This script implements an entire CNN and will also download the CIFAR-10 dataset. CIFAR-10 data consists of 60,000 RGB images of dimensions 32×32×3 for ten classes- airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. 50,000 of these images comprise the training set and 10,000 as testing. More information on this dataset, please refer to: <http://www.cs.toronto.edu/~kriz/cifar.html>.

1. **What accuracy do you get after 1 epoch?**
2. **Go through the code thoroughly and answer which layers form the core building blocks of CNN?**

**Step-3b (13 pts)** Note: This takes 20min – 1hr to run.

In this part, we will modify and train our own CNN model for classifying CIFAR-10 image data set.

Copy ‘cifar\_cnn\_architecture.m’ to ‘cifar\_cnn\_architecture1.m’ and make the following changes and report the results.

The original network architecture in ‘cifar\_cnn\_architecture.m ’ can be represented as:

conv5(32)-2 – relu –pool3,2 – conv5(32)-2 – relu – pool3,2 – conv5(64)-2 – relu – pool3,2 – FC(64) – relu – FC(10) – Softmax

where:

conv**A**(**B**)-C represents a convolutional layer with filter size **A×A** with **B** filters; C-padding

pool**D**,**E** represents max pooling, filter size **D×D**, stride of **E**;

FC(**F**) is a fully connected to **F** neuron layer;

relu is the rectified linear unit activation function; and

Softmax is the softmax probabilistic classification function.

Modify your ‘cifar\_cnn\_architecture.m’ such that it represents the following network architecture:

Conv7(32)-3 – relu –pool3,2 – conv5(32)-2 – relu – pool3,2 – conv3(64)-1 – relu – pool3,2 – FC(64) – relu – FC(10) – Softmax

An additional modification is needed to be made in the architecture. The line regarding layer weight initialization should be layers(2).Weights=0.0001\*randn([[7 7] 3 numFilters]); to avoid a dimensionality error.

Note: In the pooling step, we need to use the padding to add a zero line to the right and bottom of the image such that the output size of pool3,2 is the same as with the pool2,2. maxPooling2dLayer function from Neural Network does it by default.

For, doing this, change the following line in cifar\_cnn\_cmpe677hw8\_nnt.m to call cifar\_cnn\_architecture1.m

[layers, options] = cifar\_cnn\_architecture(opts);

Also, you will have to change the name of the model from ‘cifar10Net’ to ‘cifar10Net1’ in all the following lines to get the correct result.

if exist('cifar10Net.mat', 'file')

    load cifar10Net;

else

%Train the network.

[cifar10Net, info] = trainNetwork( imdb.images.data\_train,...

                            imdb.images.labels\_train\_gt,...

                            layers, options);

save cifar10Net;

end

%Run the trained network on the test set that was not used to train the network and predict the image labels (digits).

YTest = classify(cifar10Net, imdb.images.data\_test);

**Show your code**

1. **Train this network for 1 epoch and report the accuracy. (Note you can get the same accuracy as previous architecture)**
2. **Show your code**
3. **Explain why you need to pad a zero line to the right and bottom of the image during the pooling step.**
4. (7 pts) The previous problem implements a mini-VGGNet architecture. We will try to understand this architecture in more detail. By looking at the layers defined in cifar\_cnn\_architecture.m answer the following.

Fill in the missing values in the table (some values have been filled in for you). Note that the ‘Data size’ and ‘Data depth’ values are image size and no. of channels respectively, at a layer output.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Layer No. | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| Layer Name | input | conv | relu | pool | conv | relu | pool | conv | relu | pool | FC | Relu | FC | Soft-  max |
| Filter size | n/a |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Filter dim. | n/a |  | n/a | n/a |  | n/a | n/a |  | n/a | n/a |  | n/a |  | n/a |
| No. of filters | n/a |  | n/a | n/a |  | n/a | n/a |  | n/a | n/a |  | n/a |  | n/a |
| Stride | n/a |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Padding | n/a |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Data size |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Data depth |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1. **(10 pts) Note: This takes 10min - 30 mins to run.**

**Note: AlexNet only works for R2017a, R2017b. Please upgrade your Matlab version.**

In this problem, we will use Neural Network Toolbox to visualize the architecture of a pre-trained CNN. We will also learn how to use a pre-trained model for transfer learning.

To use the pretrained model install ‘**Neural Network Toolbox Model for AlexNet Network’**

from Matlab’s Apps tab.

Or Type alexnet at the command line.

>>alexnet

If Neural Network Toolbox Model for AlexNet Network support package is not installed, then the function provides a link to the required support package in the Add-On Explorer. To install the support package, click the link, and then click Install. Check that the installation is successful by typing alexnet at the command line.

This example shows how to use transfer learning to retrain AlexNet, a pretrained convolutional neural network, to classify a new set of images.

We also learn how to unzip and load new images as an image datastore.

**About Dataset:**

MerchData is inbuilt dataset in Matlab which is often used to demonstrate and understand the transfer learning in MathWorks tutorials. This dataset has 5 labels {Cap, Cube, Torch, Screwdriver, playing cards}, 55 training images and 20 validation images.

**Run the Transfer\_learning\_AlexNet.m, make changes in the code to display four sample validation images with their predicted labels.**

1. **Show the code and figure showing the 4 random sample validation images with their predicted labels in 4X4 subgraph.**
2. **Write a one line (very short) description for the graph?**
3. **Can you use this trained model to run with CIFAR-10 images that are 32×32×3 in size?**
4. (12 pts) Dimensionality reduction is the task of representing input samples in an alternate lower dimensional space. Which statements concerning dimensionality reduction are true:
5. Principal component analysis maximizes class separation in a lower dimensional space.
6. The eigenvalues are inversely proportional to their corresponding eigenvector importance towards reconstructing the original sample.
7. The eigenvector with the largest eigenvalue is called the principal component of the data.
8. If all eigenvectors are used, one can transform back and forth between original and PCA space without any loss of information (subject to floating point precision on the machine)
9. Linear Discriminant Analysis solves for projection vectors, such that when data is projected into this alternate space, the class means are as far apart as possible and within class variances are as large as possible.
10. Linear Discriminant Analysis is essentially the same thing as Fisher Linear Discriminant Analysis.
11. Manifold learning is based upon the assumption that your data is actually a lower dimensional object residing in a higher dimensional representation.
12. Locally Linear Embedding and Isomap assumes the lower dimensional representation is linear.
13. Linear extension of graph embedding techniques are linear approximations to non-linear manifold methods.
14. Linear extension of graph embedding techniques only apply to unsupervised clustering of data.
15. Linear methods such as Locality Preserving Projections can be used to increase class separability, but at the cost of higher dimensions than techniques such as PCA.
16. Linear methods such as Grassman Manifolds can be used to increase class separability, similar to methods such as Linear Discriminant Analysis.
17. (7 pts) Sparse representations represent signals as a linear combination of basis functions. Which statements concerning sparse representations are true:
18. The sparsest solution is the *l*-1 (norm-1) solution.
19. The basis functions of natural images are similar to the responses found in V1 of the mammalian brain.
20. Sparse solutions are not only an efficient representation of objects, but a discriminative representation as well.
21. Coefficient contamination is an artifact in which one trait of a sample, such as object pose, strongly influences secondary object traits.
22. K-SVD is a popular method for simultaneously solving basis functions along with sparse representations of a training set.
23. Minimum reconstruction error is a common technique for determining the class of an object when the dictionary is built using K-SVD.
24. (18 pts) Dimensionality reduction is a powerful technique that not only reduces the feature dimensions, but often makes the resulting unsupervised clustering or supervised classification more accurate. The following code loads image data, normalizes and vectorizes the data, then demonstrates how to perform dimensionality reduction using unsupervised PCA and supervised LPP. At the end, the top three dimensions of the dimensionality reduced data are shown in a plot. Step through line, by line, making sure to understand each step. (Note: do not step into any of the functions that are called, such as cov.m, eig.m, construct.m, or lpp.m.)

%% Dim Reduction

close all ; clear all;

%Download hwk8files\_forMycourses.zip and place them into an appropriate

%hwk8 directory. Update the two paths below for your machine

cd C:\Users\rwpeec\Desktop\rwpeec\rit\CMPE-677\_MachineIntelligence\hwk\hwk8

addpath C:\Users\rwpeec\Desktop\rwpeec\rit\CMPE-677\_MachineIntelligence\hwk\hwk6\libsvm-3.18\windows

% This cell will demonstrate a comparison of unsupervised PCA vs.

% Supervised methods such as supervised LPP

load LPP\_example\_data.mat

% 1072 faces from Kohn-Canade (they have been cropped and resampled)

% GT- ground truth:

% 100: angry

% 200: happy

% 300: neutral

% 400: sad

% 500: surprised

% allfaces20- 26x20 images

[numfaces,height,width] = size(allfaces20);

allfacesMean128\_Std100=allfaces20; % this allocates memory, makes it faster

faceu = uint8(zeros(height,width));

for i=1:numfaces

faceu(:) = uint8(allfaces20(i,:,:));

%imshow(faceu);pause %use this line if you want to display the faces

%fix mean to be 128, std dev to be 100

allfacesMean128\_Std100(i,:,:) = uint8((double(faceu)-mean(reshape(double(faceu),height\*width,1)))\*100/(std(reshape(double(faceu),height\*width,1)+0.000001))+ 128);

end

imshow(faceu); %display the last face

% Now we vectorize our data: numSamples\*numDim; numDim=26\*20=520

allfacesVnorm = zeros(numfaces,height\*width);

allfacesVnorm(:) = allfacesMean128\_Std100;

fea\_train = allfacesVnorm; %fea\_train = <num\_train\_samples> x <D>

%

% First do PCA Analysis

% You will have to set up training and test sets using some sort of k-fold

% cross validataion. This is skipped here for simplicity

%

cov\_mat\_fea\_train = cov(fea\_train);

% eigvector = <DxD>, eigvalue = <DxD>

[PCA\_eigvector, PCA\_eigvalue] = eig(cov\_mat\_fea\_train);

% eigvalue = <Dx1>

PCA\_eigvalue = diag(PCA\_eigvalue);

%sort so most important is top to bottom, left to right

PCA\_eigvalue = flipud(PCA\_eigvalue);

PCA\_eigvector = fliplr(PCA\_eigvector);

% Now apply PCA eig matrix on data and look at output

PCA\_output = fea\_train\*PCA\_eigvector; %[nxD]\*[DxD] --> [nxD]

%

% Now call LPP code in supervised mode

%

options.Metric = 'Euclidean';

options.NeighborMode = 'Supervised';

options.gnd = GT'; %gnd = 1x<num\_train\_samples>

options.bLDA = 1;

% W is the Laplacian matrix:

% In supervised mode, this is from ground truth

% In unsupervised mode, this is from neighbor distances

W = constructW(fea\_train,options); % W is <num\_train\_samples> x <num\_train\_samples>

options.PCARatio = 0.99;

%options.PCARatio = 1.0;

data=fea\_train;

[LPP\_eigvector, LPP\_eigvalue] = lpp(W, options, fea\_train); % eigvector = <Dxd>, eigvalue = <dx1>

% Now apply LPP eig matrix on data and look at output

LPP\_output = fea\_train\*LPP\_eigvector; %[nxD]\*[Dxd] --> [nxd] top d dims

%

% Plot output

%

numclasses = max(GT)/100; %assumes 100, 200, 300, ... for each class

class\_symbol{1}='ro';

class\_symbol{2}='gs';

class\_symbol{3}='bp';

class\_symbol{4}='cd';

class\_symbol{5}='m+';

method='PCA';

abc123 = PCA\_output(:,1:3)'; %restrict to 3 dims

% Note: PCA needs more than 3 dimms to classify data

% Use plot(PCA\_eigvalue) to see variance of each dim

[numdim,numsubjects] = size(abc123);

abc123\_avg = zeros(numclasses,3);

classcount = zeros(numclasses,1);

figure

for k=1:numsubjects

subj=zeros(1,3);

class = GT(k)/100;

%[i j k abc123(:,count)']

str = class\_symbol{class};

plot3d(abc123(:,k)',str,'linewidth',2); %%this looks better if standalone plot

hold on

abc123\_avg(class,:) = abc123\_avg(class,:) + abc123(:,k)';

classcount(class) = classcount(class)+1;

end

for i=1:numclasses

abc123\_avg(i,:) = abc123\_avg(i,:)./ classcount(i);

end

xlabel(sprintf('%s: Dim 1',method));

ylabel(sprintf('%s: Dim 2',method));

zlabel(sprintf('%s: Dim 3',method));

title(sprintf('%s on Expression Data: r=angry; g=happy; b=neutral; c=sad; m=surprised',method));

grid on

% This will save plot to disk, png is best for vector graphics

% jpeg is best for images

% print -dpng sampleoutputPCA.png

method='SLPP';

abc123 = LPP\_output(:,1:3)'; %restrict to 3 dims

[numdim,numsubjects] = size(abc123);

abc123\_avg = zeros(numclasses,3);

classcount = zeros(numclasses,1);

figure

for k=1:numsubjects

subj=zeros(1,3);

class = GT(k)/100;

%[i j k abc123(:,count)']

str = class\_symbol{class};

plot3d(abc123(:,k)',str,'linewidth',2); %%this looks better if standalone plot

hold on

abc123\_avg(class,:) = abc123\_avg(class,:) + abc123(:,k)';

classcount(class) = classcount(class)+1;

end

for i=1:numclasses

abc123\_avg(i,:) = abc123\_avg(i,:)./ classcount(i);

end

xlabel(sprintf('%s: Dim 1',method));

ylabel(sprintf('%s: Dim 2',method));

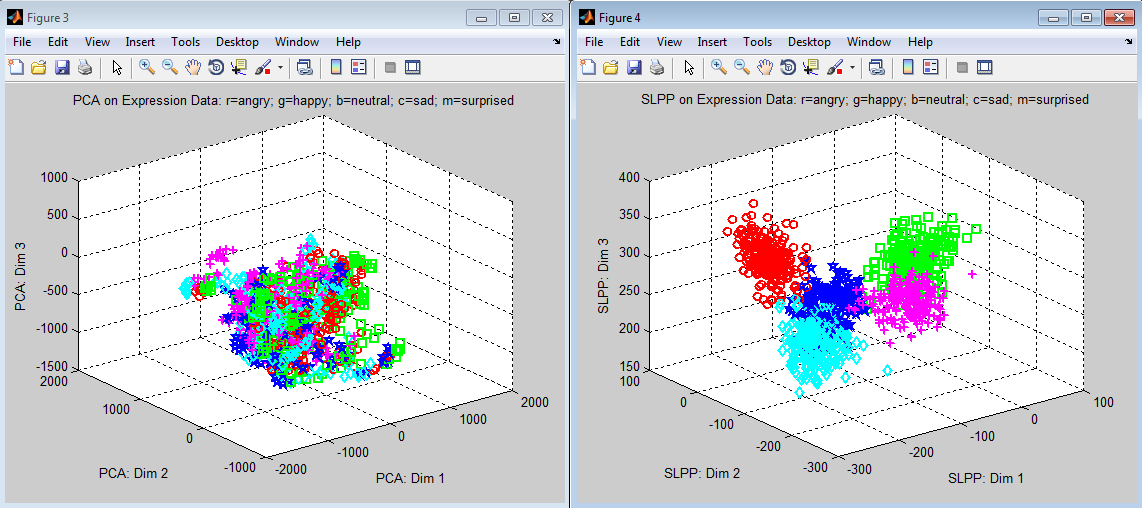
zlabel(sprintf('%s: Dim 3',method));

title(sprintf('%s on Expression Data: r=angry; g=happy; b=neutral; c=sad; m=surprised',method));

grid on

% print -dpng sampleoutputSLPP.png

You will see two plots of the sort:



Note the unfair advantage SLPP has on separating the five facial expression classes of angry, happy, neutral, sad, and surprised in extremely low dimensions. Keep in mind PCA tries to find the optimal reconstruction function for input samples while LPP is hedging its bets between optimal reconstruction and maximum class separation. The function classify677\_hwk8.m supports dimensionality reduction using Principal Component Analysis, Supervised Locality Preserving Projections, Spectral Regression, and Neighborhood Preserving Projections.

Load, ex4data1.mat, then call classify677\_hwk8.m using (linear) SVM and nnets (1 hidden layer, 25 nodes) with 5-fold cross validation. Show results for SVM **and** nnets under the three conditions:

1. No dimensionality reduction
2. PCA – For PCA, set the dimensionality reduction fields as:

options.useDR=1;

options.dim\_reduction='PCA';

options.PCARatio=0.9;

options.nnet\_hiddenLayerSize = 25;

[PCAconfusionMatrix\_nnet1,PCAaccuracy\_nnet1] = classifiy677\_hwk8(X,y,options);

1. SLPP- for LPP, set the dimensionality reduction fields as:

options.useDR=1;

options.dim\_reduction='SLPP';

options.SLPP\_bLDA=0.7;

options.PCARatio=0.9;

[SLPPconfusionMatrix\_svm,SLPPaccuracy\_svm] = classifiy677\_hwk8(X,y,options);

(12 pts) What are the classification results for each of the above six conditions?

(6 pts) What is the dimension of the original feature, and the approximate dimension of PCA and SLPP?