18-551, Spring 2002

Group #3, Final Report

Gender and Emotion Classification

Patrick Choi (pakyan@andrew.cmu.edu)

KaYoung Ky (kayoungk@andrew.cmu.edu)

Eric Lee (ericlee@andrew.cmu.edu)

Problem

The recognition of emotion and gender is an important perceptual ability. In many situations, systems require the knowledge of the emotion or the gender of the user to provide appropriate responses. For example, a nursebot needs the information of the emotion and gender of the patient to provide correct feedback in conversations.

There are many applications for determining gender and emotion from facial images. One application is that employers can use the emotion classification tool to rate the effectiveness of employees in handling customers. Moreover, obtaining information about gender and expression of users can help computers to respond to the users on a more personal level. Furthermore, venders can use this information to customize their services or product advertisement to address the mood and gender of the customers. For example, in supermarkets, there can be LCD screen, which display appropriate advertisements, such as chocolate for unhappy women, beer on sale for men, for each shopper walking by.

Solution

To address the need of gender and emotion information, we will construct a vision-based emotion and gender classifier based on neural network. We will consider three expressions: happy (positive), neutral and sad (negative). In order to simplify the project to be doable within the given timeframe, we assume the facial images to be classified are already located, centered, aligned and frontal.

Algorithm

The state of the art in gender and emotion classification makes use of different

classifiers such as neural network, Bayesian classifier, Mahalanobis, linear discrimant analysis, nearest neighbor, K-nearest neighbor. In this project, we chose to apply neural network to classify facial images.

Neural Network

Neural network learning provides a robust operate to approximating real-valued target function. In this project, the target function is a mapping of input image to a correct category of gender and emotion. Neural network is inspired by the observation that biological learning systems are built of complex, inter-connected webs of neurons. Neural networks are built out of a inter-connected set of simple units, where each unit takes a number of real-valued inputs from input to the network or outputs from other units and produce a real-valued output. In mathematical terms, the output of a unit is the output value of an activation function taking in a weighted sum of input values.

$$output = f(\sum_{i=0}^{n} w_i x_i)$$

In our project, we will construct a multi-layer feed-forward back-propagation network to approximate the target function. The back propagation algorithm learns the weights for a multi-layer network by employing gradient descent to attempt to minimize the squared error between the output values of network and the target values for these output values. The idea of gradient descent is search the hypotheses space of possible weight vectors to find the weights that best fit the training examples to make the neural network converges towards a best-fit approximation to the target function. However, if there are multiple local minima in the error surface, then there is no guarantee that gradient

descent will find the global minimum. Although there is no guarantee for reaching global minimum, back-propagation algorithm provided good results in many real-world applications. One of the possible explanations why neural network works well even it may reach local minima is that as the dimensionality of the error surface increase as the number of weights increase. When gradient descent reaches a local minimum with respect to one of the weights, it is not necessarily a local minimum with respect to all weights. Therefore, the gradient descent can move away from local minimum with respect to single weight when there are more dimensions. In our implementation, we will apply a repeated training approach to attempt to reduce the problem of local minima. We will initialize each network with different random weights and train it using the same training data set. If the different training results lead to different local minima, then we will pick the network with the best performance over a separate data set.

In the training process, the neural network may be trained to fit the characteristics of the training data set which are not present in the overall data. This problem is known as overfitting. Overfitting will cause the overall accuracy to decrease even though the training accuracy is high. To avoid the overfitting problem, we will provide the neural network a set of validation that has null intersection with the training set. The validation set is used to stop the training iteration once the trained weights reach a significantly higher error over the validation set than the previous iteration. The stored weight from previous iteration will be returned as the final network.

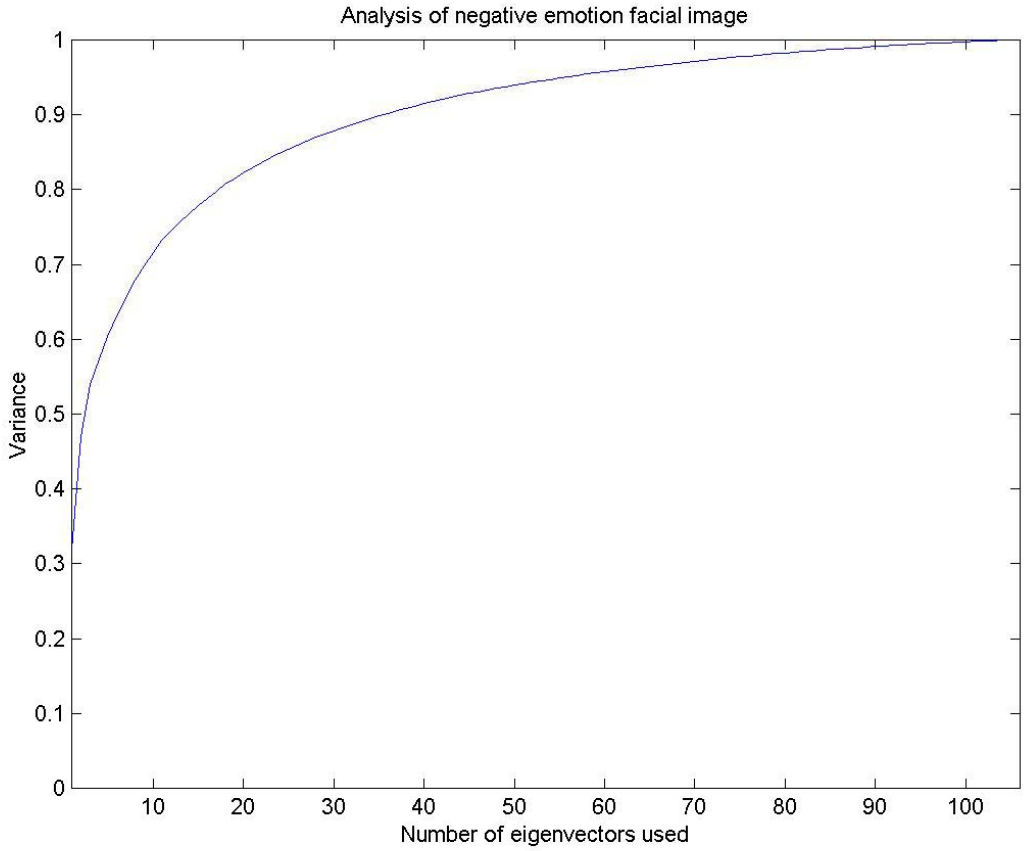
Principal Component Analysis

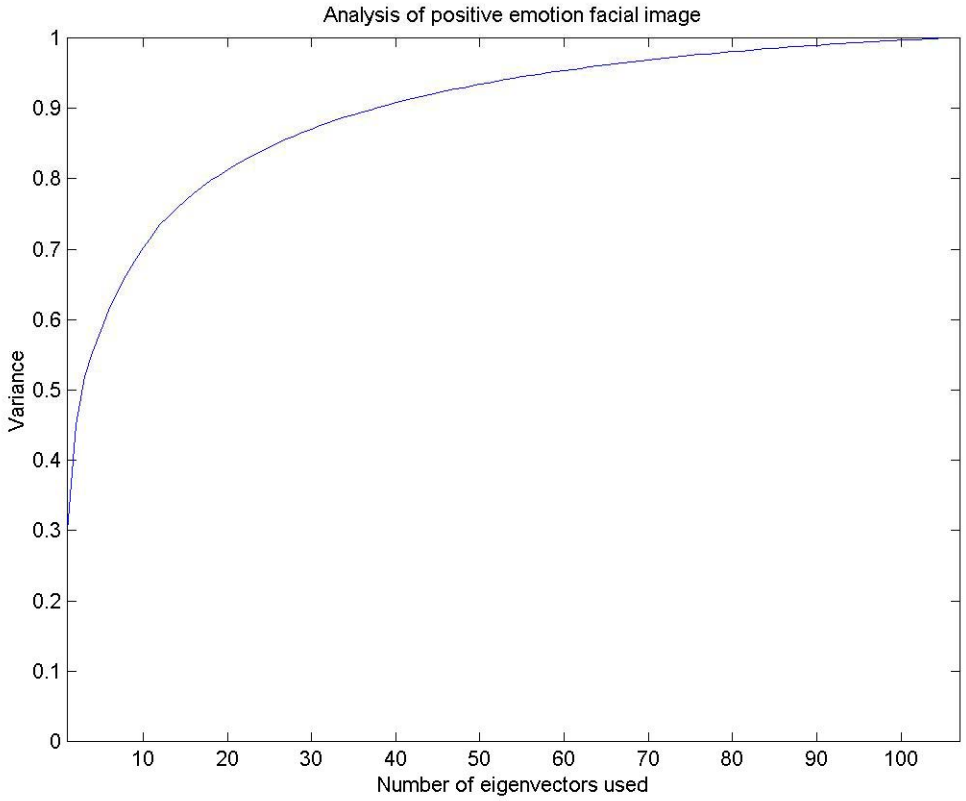
As the size of the input image grows, the size of the neural network becomes prohibitively large. To learn from inputs of 64 x 64 images, 4096 input neurons are needed and thus the computation time is long, the neural network is complex and thus the memory requirement to train the neural network may be too large for practical purpose. To deal with this problem of linearly growing of network size with respect to the size of input image, we need to reduce the dimensionality of the data. We will use a method called principal component analysis. The aim is to find a set of M orthogonal vectors in the data space that account for the data's variance as much as possible. Then we can represent each data sample by the projection of the data from the original N-dimensional space onto the M-dimensional subspace spanned by the M orthogonal vectors. This process allows us to represent each N-dimensional data by only M scalars.

To obtain the M principal component eigenvectors, we will sort the eigenvectors of covariance matrix with descending eigenvalues and select eigenvectors of the top M eigenvalues. In this application, we construct three sets of principal components for the emotion and two sets for the gender. Principal component analysis will be performed on all the training data belongs to a target category. This will return one set of eigenfaces which captures the common features of happy faces, one set of the neutral faces and another set of the sad faces. The same procedure is applied to compute a set of eigenfaces for male faces and one set for female faces.

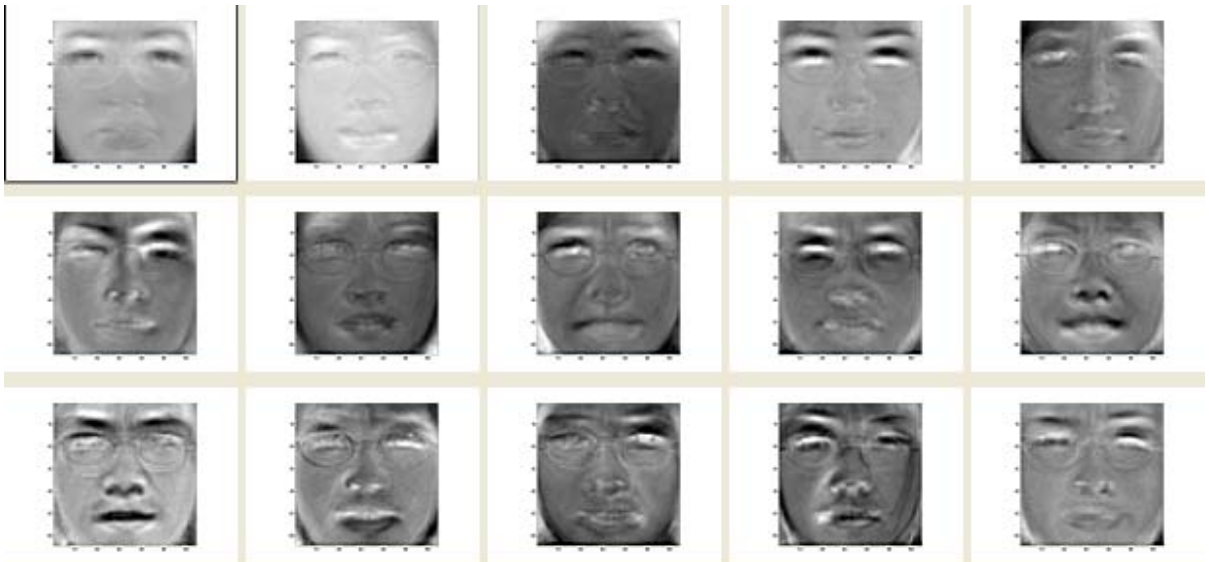
The image's eigenfeatures of target, which is the projection of the input image onto the three sets of principal component eigenvectors of facial emotion, will be fed into the emotion classifying neural network as the representation of the input. Projection onto the

two sets of principal component eigenvectors of gender data will be fed into the gender classifying neural network. For the PCA of the male images, female images, positive images, neutral images, and negative images, a total of 15 eigenvectors each were chosen. This was mainly so that around 70-80% of the variance was covered by the eigenvectors, as shown in the graphs below.





An example of what some of the eigenfaces encode is shown below:



You can see from the eigenfaces for the negative images that the faces appear to be upset, whereas in the positive images shown below, it is apparent that some of the eigenfaces encode smiles.

Prior 18-551 Work and Similar Research Topics:

The purpose of our project was to provide a system that would take in an image of human face, and then determine the emotion and gender of the person in the image. For the classifications of emotion and gender, we adopted different networks and datasets, even though we used the same kind of algorithms for input size reduction and classification. This is the only project of its kind ever attempted in this class that would test on two separate issues under the usage of the same algorithms. The class project done in spring 2001 named “Digital photo album using EPIC” (1) involved face detection that applied neural networks algorithm, which we used for both facial expression and gender classifications. However, they used the Face Detection software developed by Henry Rowley, an alumni of CMU, to deal with the face detection process. The way we did the

classifications was we actually built the networks through Matlab and then extracted the weights in the networks to the EVM for testing. Besides, the main purpose of our project was for facial identification and not face detection, therefore our project was significantly different from other previous projects.

There were researches in different places on the topic of facial expression recognition, for example, “A Neural Network Facial Expression Recognition System using Unsupervised Local Processing” (3). The idea behind it was that a local unsupervised processing stage was inserted within a neural network constructed to recognize facial expressions. The stage was applied in order to reduce the dimensionality of the input data while preserving some topological structure. We applied a concept that was similar to the research, but our structure of the network was different and we used a much larger database for both training and testing processes. Our neural net architecture included three layers, and it was well-tuned so that it was suitable for both facial expression and gender recognitions. Other than recognizing a small set of prototypic expressions, such as happy and sad, there was a research that tried to capture the full range of facial expression by using Facial Action Coding System (FACS) (4). The Facial Action Coding System (FACS) was a human-observer-based system designed to detect subtle changes in facial features. Viewing videotaped facial behavior in slow motion, trained observers could manually FACS code all possible facial displays, which were referred to as action units (AU). However, compared to the many possibilities of facial displays, we limited our classification to three basic emotions, which were happy, neutral and sad.

Our Databases:

One important part of this project was to carefully select the images for our databases. First of all, our databases should be large enough that would give statistically significant training and testing results. Besides, since we proposed to have three sets of data, which were training images, testing images and validation images, we had to have a large amount of images in order to provide a reasonable amount for each set of data. We decided to use 126 images for testing, 51 images for validation, and 252 images for training, and each of the images was randomly chosen from the pool of databases we got.

We collected the databases from different resources. One of the large ones was called the Postech Faces ’01 Database. It had a total of 309 images, which included 103 subjects, where 53 of them are male and 50 of them are female. For each of the subjects there were different expressions. Originally, the images were with dimensions of 1280x960 pixels and were not cropped to our desired size, 64x64 pixels, and they were 24 bits full-colored images as well.

Sample images of Postech Faces Database

The second database we had was the PICS(Stirling) Database, and it had a total of 68 images, where each of the 34 subjects had 2 expressions. The images were not cropped,

even though they were in grayscale. One of the biggest problem we found within this database was that it only had two emotions associated with each subject, and those two emotions were not all different either, for example some subjects had two expressions of being happy or normal. Due to the inconsistence of expressions of each subject, we decided not to use this database.

Samples from PICS Database

The third database was the MBDAS Database, which contained 30 subjects and each subject had 75 different expressions. The images were nicely cropped and were 64x64 pixels in dimensions. They were all in grayscale as well. Due to the many different expressions that each subject possessed, we had to carefully pick within each subject the three expressions that could mostly represent happy, neutral and sad.

Sample images of MBDAS Database

The last one was the Yale Database. It contained 15 subjects, and each subject displayed 4 different expressions. All the images in it were not cropped and were 320x283 pixels in dimensions. However, they were all in grayscale.

Samples from Yale Database

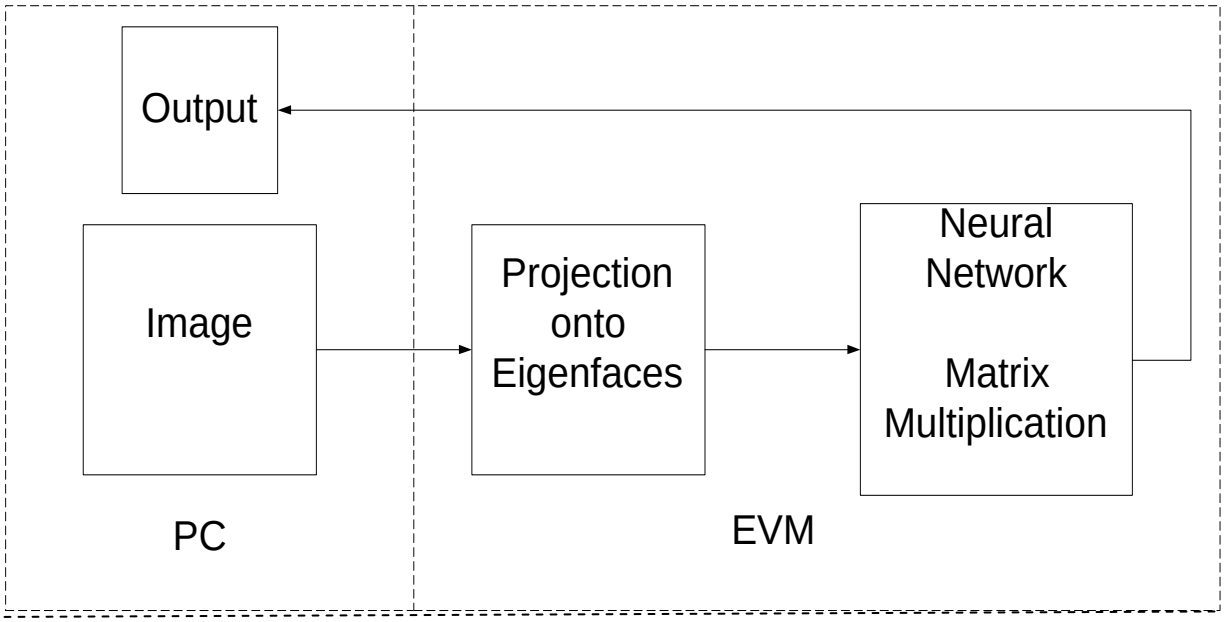
As a result of the large number of subjects we got from the databases, we could avoid doing cross-validation. Besides, we could retrieve better weights for neural network with large amount of training data, and also we would be able to provide a larger test set for showing the desired accuracy.

We standardized all our data images to 64x64 pixels in dimensions and also to grayscale. Also, we tried our best effort to manually crop the images to just capture the frontal faces of the subjects, and make sure they were all center-aligned. Creating the databases was time-consuming, since we basically handled each image manually to guarantee that it would fulfill the requirement for training and testing.

Implementation and Flow Diagram:

The development of both gender and facial expressions recognition systems involved implementations on PC and EVM. On the PC side, we would feed in raw image

data to the EVM. But before we could do that, we first preprocessed the data by PCA to reduce its dimensions. The raw data was then projected onto the eigenfaces that were stored in EVM. After that, the output of projection would input to the neural network simulator that involved matrix multiplication with that weights and biases that were stored in EVM. The decision output of the neural network simulator would then send back to PC and display to the monitor. The following is the flow diagram of our implementations:



Training and Testing

While training the neural network, 3 different data sets were used: a training set, a validation set, and a test set. There were a total of 429 images. Of these 429 images, 252 belonged to the training set, 51 belonged to the validation set, and 126 belonged to the test set. These images were randomly chosen so that each set would include images from the various databases, rather than just one.

All of the training was done in Matlab with the Neural Network Toolbox. The neural network object in Matlab consists of a number of sub-objects and sub-structures that can be changed. Therefore the network structure, training algorithm, etc for the neural network can be modified without much effort. This was one of the main reasons that it was done on Matlab rather than with C code or the EVM.

Although the network structure varied slightly among the three different networks that we created, it only differed in the number of perceptrons in each layer. The input layer is connected to a hidden layer which is then connected to an output layer. The activation function of the hidden layer is called “tansig”. This is known as the tan-sigmoid transfer function given by the following formula:

$$tansig(n) = \frac{2}{1+e^{-2n}} - 1$$

The activation function used in the output layer is called “poslin”. This is just a positive linear transfer function that returns 0 if the input to it is less than or equal to 0. Otherwise, it just returns the input value. In the network for emotion classification with 3 outputs, there were 45 input perceptrons, 15 hidden layer perceptrons, and 3 output perceptrons whereas the network with 2 outputs had the same structure except it only had 1 output

perceptron. The network for the gender classification had 30 input perceptrons, 15 hidden layer perceptrons and 2 output perceptrons.

The training algorithm we applied to the neural network required a training set and a validation set. Furthermore, each of these data sets required their target values. The algorithm runs until:

1) The max number of iterations is reached
2) Performance goal is reached (MSE is 0)
3) Gradient becomes to small
4) MSE of Validation set increases

The weights are only adjusted with the data from the training set. The validation set is there to make sure that the network doesn't become overtrained to the training data and thus it won't be as accurate with any other data.

The algorithm used in updating the weights was called 'trainlm' in Matlab. This is an implementation of the Levenberg-Marquardt back-propagation optimization. Furthermore, the algorithm in Matlab requires a large deal of memory (to speed up training). If there isn't enough memory, speed can be sacrificed to compensate via one of the parameters (mem_reduc).

Since there are a lot of weights to learn and not enough training images, we rely somewhat on the randomization of the weights when the network is initialized since depending on where the network starts, it might not end up on the same local minimum. The network is created, and trained until one of the conditions mentioned above occurs. The accuracy is computed. If the network has the best accuracy seen so far, it is saved. This process is repeated until a sufficient network is found.

After training the network, we then simulate it with the test data to calculate its accuracy. The following tables are the results:

Gender	Training Set	Validation Set	Test Set
Male	93.33 %	90.00 %	90.12 %
Female	94.95 %	85.71 %	94.44 %
Total	93.98 %	88.24 %	91.85 %

3 Emotion Output	Training Set	Validation Set	Test Set
Positive	92.77 %	76.47 %	82.22 %
Neutral	77.11 %	35.29 %	51.11 %
Negative	56.63 %	52.91 %	48.89 %
Total	75.50 %	54.94 %	60.74 %

2 Emotion Output	Training Set	Validation Set	Test Set
Positive	100.00 %	76.47 %	82.22 %
Neutral	100.00 %	82.35 %	91.11 %
Negative	93.98 %	100.00 %	82.22 %
Total	98.00 %	86.27 %	85.19 %

The network for classifying gender performed very well, getting around 90% accuracy. It is apparent that the network with 3 outputs for the emotion worked well for positive images, but poorly for neutral or negative images. We hypothesized that this was due to the fact that some negative images appeared to be neutral, or sometimes a few even appeared to be happy. This is probably due to the fact that the images are still images and thus it may seem to be happy or neutral when taken out of context, since there is no information as to what the face looked like before or after. The example images below are all supposed to be negative. However, they appear don't really seem negative. The subjects are trying to appear annoyed/irritated/sad, but instead they seem neutral or happy.

Therefore, we went to 2 outputs for emotion, either positive or non-positive (neutral or negative). As can be seen from the results in the previous tables, this greatly increased performance.

The results in the tables that follow include all images from the training, validation, and test sets.

In/Out	Positive	Neutral	Negative
Positive	127	10	8
Neutral	21	93	31
Negative	27	41	77

Our previous hypothesis was confirmed from the data above. The neutral images when misclassified were misclassified as negative images about 60% of the time. The negative images were misclassified almost half the time, mainly as neutral images. The next table shows that the number of misclassifications go down if we combine neutral and negative together and create a network.

In/Out	Positive	Neutral or Negative
Positive	133	12
Neutral or Negative	20	270

In/Out	Female	Male
Female	162	12
Male	9	252

Performance	Gender		3 Emotions		2 Emotions
		Positive	88 %		
Male	97 %	Neutral	64 %	Positive	93 %
Female	93 %	Negative	53 %	Neutral/Neg	92 %
Overall	95 %	Overall	68 %	Overall	93 %

Matlab Code (See commented Matlab code for more detail).

The main scripts in training the networks are go_old.m, go_new.m, go_gender.m which are for the emotion classifier with 3 outputs, the emotion classifier with 2 outputs and the

gender classifier respectively. Each of the scripts basically does the same thing. The first thing the scripts do is load the images. Then PCA is run on training images. After PCA finishes, each of the images from the training, validation, and test set is projected onto the eigenvectors. Target values for each of the images is created and then a network is created. The test and validation sets are used for training and when complete, various data is saved in Matlab format. In addition, after training is complete, the network is simulated with the test set and statistics are outputted.

- Together, the pc_evectors.m and sortem.m files compute the PCA and returns the requested number of eigenvectors.
- acc.m and accuracy.m compute the accuracy of the network.
- loadimg.m and loadimg_gender.m load the images for creating the emotion and gender networks, respectively.
- myproject.m projects an image onto the eigenvectors.
- randomfiles.m randomizes the subjects so that the various data sets include subjects from more than one database
- makeRaws.m and createRaw.m converts the database of bmp files into raw files for the PC and EVM version.
- exportWeights.m takes the saved Matlab files generated by the main scripts and exports everything into text files to be loaded by the PC and EVM version
- showImage.m is used to read and display the image from the raw files. This is useful for the PC and EVM when we want to see what image we are classifying.

C Code (See commented code for more detail)

PC Only Version

The program begins with an initialization function. This function reads in the eigenvectors that came from doing PCA on the images. It also reads in all the weights and biases of the neural network. Then it prompts the user for input. At this point, the user can run the classification on all the images, on a specific image, or quit. The image(s) are projected onto the appropriate eigenvectors and then sent to a function that simulates the behavior of a neural network. From this an output is generated. This process is repeated for each of the three networks. After the classifications are complete, the results are outputted to the screen and the user is prompted for input once again.

PC and EVM Version

This works very similar to the PC Only Version. The PC side of the program initializes by reading in the eigenvectors, weights, and biases for each of the networks. It then loops and prompts for an image to be classified or to quit. The EVM requests for the eigenvectors, weights, and biases once. It then loops by repeatedly requesting for an image and performing the calculations described above.

The only code that we did not write was the code for performing PCA. This code was found at the following page: http://ai.ucsd.edu/Tutorial/matlab.html

EVM Memory Usage

In order to make the EVM run as fast as possible, everything should go on on-chip memory. However, when this isn't possible, as in our case, we must decide what data goes where. The main problem for us is the PCA data. We have 45 eigenvectors used in emotion classification and 30 eigenvectors for gender classification. Therefore, there are (45x64x64 + 30x64x64) or 307,200 values at 4 bytes each since floats are being used. This gives us a total of 1200 KB. Accordingly, we placed these eigenvectors on

SDRAM0. The weights and biases all fit on-chip since there weren't too many values; 45x15+15x1+15x1+1+45x15+15x3+15x1+3x1+ 30x15+15x2+15x1+2x1=1941 values at 4 bytes each gives us 7764 bytes. We also decided to store the image that is being classified on on-chip memory as well. This takes up 64x64x4 bytes = 16 KB. Thus, there is about 8 KB of memory left on-chip for constants and temporary data.

EVM Speed

The main functions that would be called over and over again on the EVM after initialization are calcGender, calcMood1 and calcMood2. The profile results for these three functions on optimization level 3 are as follows:

	Number of Cycles
calcGender	3,404,724
calcMood1 (3 outputs)	3,398,409
calcMood2 (2 outputs)	2,277,906

We didn't end up having enough time to optimize performance on the EVM by using the ~8 KB of space left for memory paging. However, we believe that the performance time of our code is sufficient. There were only a few main loops in our code. One of them was for doing the projection onto the eigenvectors. Since the eigenvectors were orthonormal, all that was required was a dot product between the image and each eigenvector (the images and eigenvectors were converted to column vectors). The loop ran for 1024 iterations (64x64 image = 1024 column vector). It was unrolled by a factor of 4 and ran with 4 instructions in parallel. The limitation was only due to the availability of registers. The only other loop for optimization was located in the code that multiplied the weights and added the biases for the three networks. However, these loops were fairly small and thus couldn't be optimized as much.

Original Schedule

Week	Tasks
3-4	Face Images, EVM Communication, PCA code, Matrix ops on EVM
3-11	
3-18	Build Neural Network in Matlab
3-25	Oral Project Updates + Demo of PCA
4-1	Spring Break
4-8	Train and Test Neural Network for Emotion Classification
4-15	Test on DSP
4-22	If time, repeat for Gender Classification. Otherwise, Debug Time
4-29	Demo
5-6	Written Report

Due to projects and assignments from other classes, we were behind schedule in the period after spring break. However, we did fit in gender classification into our schedule. The original task breakup in the schedule was as follows:

Images – Eric
EVM – Kay
Matrix stuff – Kay
PCA – Patrick (and others if too difficult)
Build, Train, and Test Neural Network – All
Test on EVM – All

Reference

Databases:

MBDAS (CMU): http://amp.ece.cmu.edu/projects/FaceAuthentication

Postech: http://nova.postech.ac.kr/archives/imdb.html

PICS (Stirling): http://pics.psych.stir.ac.uk/

Yale Faces: http://cvc.yale.edu/projects/yalefaces/yalefaces.html

Neural Network Tutorials:

http://www.ida.his.se/ida/kurser/ai_ann/kursmaterial/tutorial/node1.html

http://carol.wins.uva.ml/~portegie/matlab/nnt/

Papers

(1) Rucchi Mittal, Jennifer Wong, "DIGITAL PHOTO ALBUM USING EPIC"
https://blackboard.andrew.cmu.edu/courses/1/S02-18551/content/_43043_1/Group6.pdf

Gender Classifaction: Introduction
http://white.stanford.edu/~dicarlo/ee368/intro.html

Sun, Zehang et al. "Neural-Network-Based Gender Classification Using Genetic Search for Eigen-Feature Selection"
http://www.cs.unr.edu/~bebis/genderIJCNN.pdf

Eigenflow Based Face Authentication Project Database
http://amp.ece.cmu.edu/projects/FaceAuthentication/download.htm

Dailey, M.N. & Cottrell, G.W. (1999), "PCA = Gabor for Expression Recognition"
http://www.cse.ucsd.edu/users/mdailey/papers/UCSD-CS-629.pdf

Y. Tian, T. Kanade, and J. Cohn (2001), "Recognizing action units for facial expression analysis"
http://www.ri.cmu.edu/pub_files/pub2/tian_ying_li_2001_2/tian_ying_li_2001_2.pdf

(3) L. Franco, A. Treves (1998), "A Neural Network Facial Expression Recognition System using Unsupervised Local Processing"
http://www.sissa.it/~lfranco/ipsa.pdf

(4) Lien, Kanade, Cohn, & Li, "Detection, Tracking, and Classification of Action Units in Facial Expression", in *Journal of Robotics and Autonomous Systems*
http://www-2.cs.cmu.edu/~face/Papers/RA_Journal99b.PDF