Feature Extraction for Face Recognition and Implementation of small scale Face Recognition application

Machine Learning (1DT071)
Uppsala University – Spring 2011
Report for Project

Peramanathan SATHYAMOORTHY

Abstract

This paper is about the survey on feature extraction for face recognition. Face recognition is a challenging task for machines rather than that of human. For the past two decades lot of researchers developed various techniques and methods. Among all , this paper treat the novices with some significant information and the details. Main focus is to help the beginners, interested in developing face recognition tools and new methods, by providing them basic idea behind the face recognition, the place where one can start learn and exercise the skills he developed in this area of Face Recognition and other useful links. Finally implementation of small scale face recognition application is explained with enough detail

1 Introduction

Face Recognition is the compliment to face detection, where the faces that are feed or detected will be checked for matching with the faces that are stored already in a database. The tools and software that supports the face recognition , initially need help, a kind of training unlike some traditional software developed in some programming language where the algorithm works out fine for the single procedure call. Once the face recognition system got enough training , the system become more cost effective in many scenarios, as an ex-

ample, a system that analyse passenger passports including face recognition, can be more quicker than human being[1]

2 Facial Data Sets

Selecting suitable facial data set is one of the important and crucial steps, as face appearance can be affected by various factors including pose, illumination, expression,occlusion, hair. Hence to check the robustness of the algorithms, we need some well-sized facial database that is a careful mixture of these various factors[2]. It is also good practice to consider the type of problem to be solved and the corresponding requirements of the algorithm before selecting the database. In [2] there are 27 publicly available database are well explained

3 Feature Extraction

Feature extraction algorithms mainly divide into two categories: *1)Geometrical features extraction 2)Statistical (algebraic)* features extraction. Feature extraction process should be less time consuming and must be efficient with storage

3.1 Geometrical

The geometrical approaches make use of structural measurements and distinctive facial features . It involve checking similarities like distances and angles between nose, mouth, eyes and all kind of measurements of facial features that comply and help in mapping faces[3]

3.2 Photometric

This approach need algebraic methods to accomplish it's task. There are methods that are capable of reducing the dimensionality of the original feature space, namely Principal Component Analysis (PCA). Here images are encoded into suitable format so that all statistical analysis can be done in order to achieve the best mapping between faces. But light and changes of expressions may affect the performance of these methods. For feature extraction,Linear Discriminant Analysis (LDA) has higher performance than PCA in face recognition[3]

4 Algorithms

We need algorithms to extract features, to select best features among them, and finally to map the given face with the face stored in the database. Feature extraction is key part of the face recognition process.There are many algorithms existing which are less expensive. But selecting best features is problem and database dependent and also it is so expensive to afford in large scale applications. There are many algorithms to perform face recognition including:

- Eigenfaces or Principal Component Analysis method (PCA)
- Fisherfaces or Linear Discriminant Analysis method
- Kernel Methods
- 3D face recognition methods
- Gabor Wavelets method
- Hidden Markov Models
- Active Appearance Models

Most of these algorithms well explained in [4]. Interesting part is that necessary fundamental mathematics behind these algorithms also explained which will give the reader a good insight and overview.

5 Evaluation of Algorithms

There are few organizations volunteer in measuring the performance and accuracy of these face recognition algorithms with powerful tests such as Face Recognition Vendors Test(FRVT). FRVT2006 results are well documented in [5].Many efficient algorithms that are developed recently, though built on fundamental algorithms such as PCA,they are performing super sonically. Interesting question one may rise is what makes these algorithms robust and reliable. Whether it is because of super power computers,many dedicated hardware and software which are not available those days. The document [5] is holding these secrets as it is dealing with business vendors' algorithms and only exploring how well the algorithms over the period of time improved. Hence interested readers should dig the answers themselves.

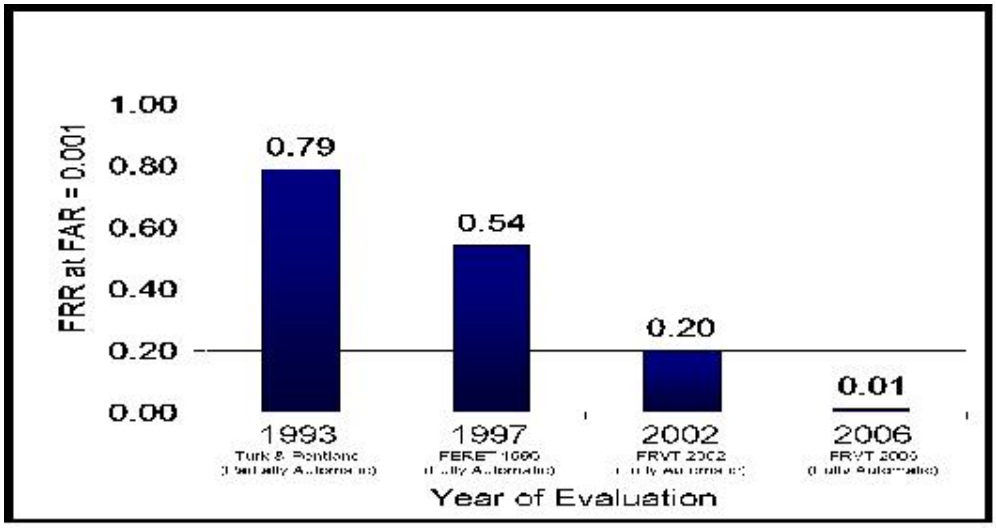


Figure 1: Reduction in Error Rate documented in [5]

6 Models

There are many candidates capable of solving face recognition problem, among them Artificial Neural Networks is realized well suitable for many problems, Hybrid or ANN based models are best employee in face recognition [10]. Hidden Markov Models[HMM] is also becoming popular candidate for face recognition in research community[11]. One should carefully design ANN to get a best result [9].

7 Applications

The earliest work on this field were done in Psychology [6]. After this work published in 1950's slowly researchers started engineering the face recognition problem and later 1980's the field drawn over many diverse areas and now it is active research area. Table from [4] is the quick view of the face recognition applications in various fields of study. Face Recognition Applications in various fields

8 Web Resources

As this paper is focusing in helping novices, students it make sense providing, in this section, some web resources where they can start with. One should not completely rely on web resources that he can not take it for granted as

Areas	Applications
Information Security	Access security (OS, data bases) Data privacy (e.g. medical records) User authentication (trading, on line banking)
Access management	Secure access authentication (restricted facilities) Permission based systems Access log or audit trails
Biometrics	Person identification (national IDs, Passports, voter registrations, driver licenses) Automated identity verification (border controls)
Law Enforcement	Video surveillance Suspect identification Suspect tracking (investigation) Simulated aging Forensic Reconstruction of faces from remains
Personal security	Home video surveillance systems Expression interpretation (driver monitoring system)
Entertainment - Leisure	Home video game systems Photo camera applications

Figure 2: Face Recognition Applications in various fields

they don't have proper citations and for, most of the information are personal opinions that are not scientifically proven or accepted by International Community. Hence we decided to choose carefully the best websites that are constantly evolving , providing citations and useful links and that are non-commercial, academic websites. It will give good insights to the novices

8.1 Department of Machine Learning ,Carnegie Mellon University,USA

This department provides lot of project exercises for both their students and outsiders. Many exercises are very well explained and provided with necessary source code skeletons and corresponding documentation. So this is good place to start the training. Exciting thing is that they keep older versions of the course instances which ensures the users to use them for long time, all one has to do is bookmark the web page. A small project exercise related to face recognition can be found in [12].

8.2 Face Recognition Homepage

It is purely academic website, designed as a portal. It contains necessary information, direct links to interesting papers. This site is constantly up-

dated so that one can get to know what has been done in the field of face recognition recently. Other features includes algorithms with representative papers, source codes, list of conferences and journals. The link to this site can be found in [13]

9 Implementation

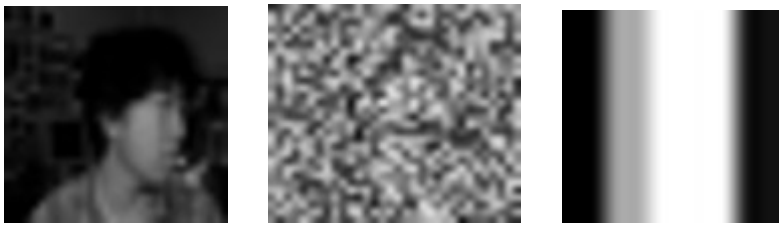
We have used a package(skeleton) written in C programming language at [12].The package contains code for three-layer fully-connected feed forward network. It uses back propagation algorithm to tunes its weights and sigmoid function as activation function. The documentation provided with this package is to exercise various aspects of ANN with respect to face recognition.So in order to understand the package one has to really go through the code and should know the concept of ANN and how the back propagation algorithm works.

9.1 Database

Image database is available at the same place where we download the package. This database directory contains 20 subdirectories, one for each person, named by userid. Each of these directories contains several different face images of the same person. Naming convention of the image is `<userid>_<pose>_<expression>_<eyes>_<scale>.pgm` which makes it easier to prepare training list and test lists and also to write program that access images. As you can see the images are all gray scale images of dimension 32x30 with PGM(Portable Gray Map) format[16].

9.2 Feature Extraction

In the image database we used, each image has four features : id of the person, the direction he/she is looking, expression, whether or not wearing sunglasses. The images named with the values for these features. Eg: `an2i_left_angry_sunglasses_4.pgm` means that the person with id *an2i* facing to his *left* with *anger* and wore *sunglasses*. Hence one can avoid writing algorithm to extract these features by using these tags. Look at the figure 3 and figure 4 to know how the hidden and output units learned the features. If one want to experiment extracting other features, then learning particular part of a face such as eyes would be a great choice. In such case one has to manually find the helping coordinates to restrict the learning region

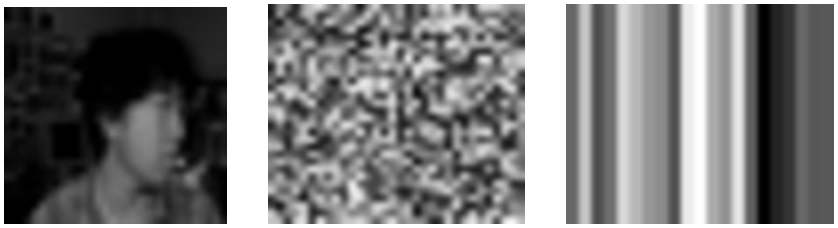


(a) Input Image-an2i_left_angry_open

(b) Weights of hidden unit 1 from input layer

(c) Weights of output unit 1 from hidden layer

Figure 3: Sunglass Recognizer: Learning features



(a) Input Image-an2i_left_angry_open

(b) Weights of hidden unit 10 from input layer

(c) Weights of output unit 3 from hidden layer

Figure 4: Face recognizer: Learning features

and set up the target according to this coordinates in order to work with this package.

9.3 Training and Testing

The documentation of the package suggests a strategy that roughly $\frac{1}{3}$ of the images have been held over for testing. The remaining $\frac{2}{3}$ have been used for a train and cross-validate strategy, in which $\frac{2}{3}$ of these are being used for as a training set and $\frac{1}{3}$ are being used for the validation set to decide when to halt training. Our experiments gave us good results when we applied this strategy.There are 20 persons and for each of them there 32 various type of images available. Here we have 3 types of *training set,validation set (test1)* and *test set (test2)*, namely *all,straighteven,straightrnd* in which *straighteven* lists have the images with `<pose>` straight(facing straight) and images are evenly distributed. *straightrnd* is same like *straighteven* but the images are randomly distributed.

No. of epochs	delta	perf. on straighteven_ test1.list	error	perf. on straightrnd_ test2.list	error
100	0.238773	100	0.000453722	97.5	0.00382158
200	0.189292	100	0.000591305	97.5	0.00403161
300	0.152758	100	0.000911683	97.5	0.00462989
400	0.128958	100	0.000928995	97.5	0.0046575
500	0.109478	100	0.000948846	97.5	0.00451811
600	0.0953549	100	0.00096598	100	0.00433362

Table 1: Performance of Sunglass Recognizer trained with *straighteven_train.list*

9.4 Sunglass recognizer

At initial phase we implemented *sunglasses recognizer* which tells whether the person in the given image wear sunglasses.Initially we built up the ANN with 960(32x30) Input units, one for each pixel, 2 hidden units and 1 output unit. We trained ANN by setting ***learning rate*** to 0.3 , ***momentum*** to 0.3 for 600 ***epochs***. It took about 3 minutes to complete training where there are 277 images in training set , 139 images in validation set, 208 images in test set. It only classified 69.78 % images correctly. As we realized the changes in these key parameters wouldn't help much, we increased number of hidden units to 4. For the same settings as mentioned above, the ANN trained on ***straighteven_train.list***(80 images). It classified all the images in ***straighteven_test1.list***(36 images) and in ***straightrnd_test2.list***(52 images) correctly.

Refer the table 2 for the performance of ***sunglasses recognizer***. When we trained the ANN using the images in which person looks straight and test on the remaining straight images the recognizer easily classified the images. But when we use ***all_train.list*** which contains the images of persons looking at different directions(left, right, straight, up) , it becomes very difficult for the learner to generalize thus learning features is a challenging task in this case. Consider the example scenario: Image of a person with sunglasses looking to his left(only one piece of sunglasses is visible) versus Image of a person with sunglasses looking straight to visualize the problem in generalization.

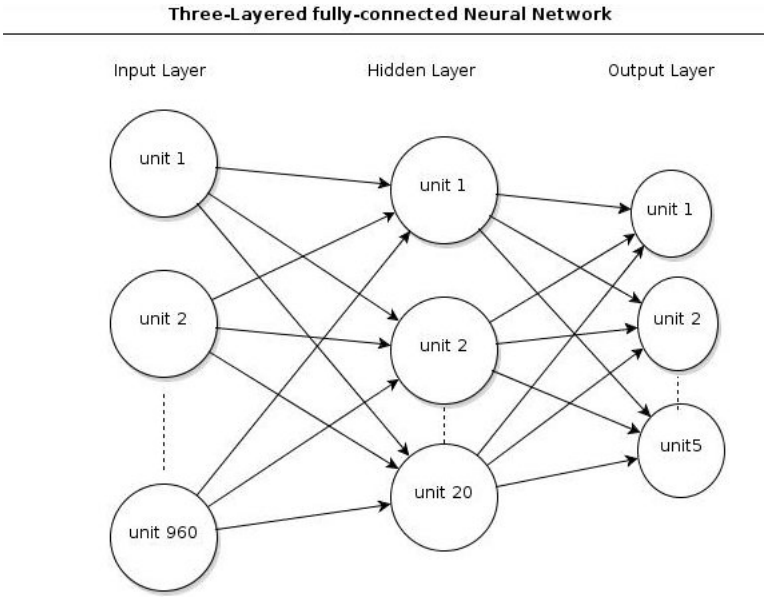
No. of epochs	delta	perf. on straighteven_ test1.list	error	perf. on all_ test2.list	error
100	1.99613	87.0504	0.0339695	79.3269	0.0472411
200	1.36948	89.2086	0.0320138	79.8077	0.046761
300	1.12189	88.4892	0.0319679	80.2885	0.0460522
400	0.982509	87.7698	0.0318961	80.7692	0.0457604
500	0.865011	88.4892	0.0317416	80.7692	0.0456072
600	0.768392	87.7698	0.0316132	80.7692	0.0455016
2000	0.322998	89.2086	0.0326147	81.25	0.0448968

Table 2: Performance of Sunglass Recognizer trained with *all_train.list*

9.5 Face Recognizer

In this phase, we first implemented a helper module ***lookup*** to help setting up the target vector according to the person. It provides the utility to use $< userid >$(a string) directly in *switch* statement.

9.5.1 Designing of ANN and encoding of target



(a) ANN with 5 output units

Figure 5: Face recognizer

.

Important task here is to design a ANN then comes finding good values for the key parameters. As the input is an image of size (32x30),number of **input units** is set to **960**. Then the number of **hidden units** is set to **20**, as we have 20 different patterns(persons),this idea is available in the lecture notes of this course[1DT071],provided to us. Finally for the number of output units and vector encoding there was no help we had. So, we initially started off with **20 output units**, one unit for each different person. As here we used sigmoid function as activation function, the value range is [0, 1]. Hence each time we load the image into input units we set the corresponding element of target vector HIGH depends upon the person in the image. For example, the target vector for the person *an2i* is [0.9,0.1,...,0.1]. We observed training was too time consuming for this ANN of size 960x20x20. But the trained network when *straighteven_train.list* was used for training, could capable of classifying all the remaining images of straight faces. It classified 99.04 % images in *straighteven_test2.list*.

Later we had an idea to reduce number of output units. We found that it is sufficient to use k number of output units if $2^k >= n$ and $2^{k-1} < n$, where n is number of patterns(persons). So for 20 different persons 5 output units are sufficient. For example, the target vector for the person *choon* (with index 7) is [0.9,0.9,0.9,0.1,0.1]. This technique we call it mirrored-binary system. The number 7 is 00111 in 5-bit binary system. But we encode 7 as 11100, as it is convenient in coding with array indices which is left to right as oppose to binary system which is right to left.

9.5.2 Results

The key parameters that affect the performance are *no. of epochs, learning rate, momentum,seed for random number generator*. Here same *seed* generates same sequence of random numbers as initial weights. So if one want to run the same experiment again, he should use same seed. Performance loss due to the change in seed could be recovered by increasing number of *epochs*. We fixed number of epochs to 2000 which take `2` minutes `29.368` seconds to complete. We reduced the no. of epochs only if over fitting observed or increased the no. of epochs if performance is poorer than expected. Refer the table 3 for the performance of face recognizer for the different values of *learning rate* and *momentum*. Under the following setting we achieved best results:

No. of Epochs	: 2000
Learning rate	: 0.4
Momentum	: 0.4

Learning rate	momentum	perf. on straighteven_ test1.list	error	perf. on all_ test2.list	error
0.3	0.3	91.6667	0.0400509	92.5	0.0617751
0.4	0.3	91.6667	0.037616	92.5	0.0548346
0.7	0.3	94.4444	0.0337538	90	0.0432343
0.4	0.4	97.2222	0.0302223	92.5	0.0446316

Table 3: Performance of Face Recognizer trained with *straighteven_train.list* for 2000 epochs

10 Conclusion

Active research in face recognition leads to various types of research, the area is becoming wide and advanced. It also initiated many new related ideas like facial expression recognition[7], caricature recognition [8],ethnicity recognition and Identical Twins recognition and so on. Among various techniques, Artificial Neural Networks specially Hybrid ANN is realized as the best employee for face recognition problem, provided that ANN should be designed and implemented efficiently.

Appendix A

Machine Configuration

Operating System	: Ubuntu 11.10
Memory	: 3.6 GiB
Processor	: Intel® Core™ i3 CPU M 350 @ 2.27GHz x 4
OS type	: 64 bit
Disk	: 30.7 GB

Tools

- **eclipse with c/c++ mode**
- **gdb debugger (@ linux terminal)** - very useful to find segmentation fault. Segmentation fault is more likely as we use *pointers*, *pointer of pointers* and structures extensively and we have to use lot of memory to load about 600 images. So, successful compilation and built is not guaranteed that it will work correct. It may get struck during training or testing. In such case gdb is handy and help us to fix the bug.
- **time command (@linux terminal)** - It help us to find the running time of the application
- **display command (@linux terminal)** - To view images, to view and save the weight updates of hidden units and output units as image.
- **python** - We observed it is very difficult and time consuming to tabulate or to draw graphs of the performance of this face recognition application from the output in `stdout`. For example if you want to tabulate the performance for every 500 epochs till 3000 epochs, then you have to run the application for 500 epochs, then for 1000 epochs, and so on, as we have the restriction on no. of lines that could be shown in terminal window. It is good idea to use **python for automatic result generation** in desired format (eg- **.tex** format for latex)

Makefile

In order built this project we use ***make*** command. It will create a executable `./facetrain`. In order to built a utility to view hidden units' weights, use ***make hidtopgm*** and to built a utility to view output units' weights, use ***make outtopgm***. If you add new modules make sure that you compile it in right order and linking them to right modules, since the modules we have interconnection between them.

Sample outputs

Start up - Sunglasses Recognizer

This is the sample output of sunglass recognizer

Phase I - training

```
peramanathan@ubuntu:~Documents/ML/Sunglass_Recognizer/Program$
gdb --args time ./facetrain -n sunglasses.net -t straighteven_train.list
-1 straighteven_test1.list -2 straighteven_test2.list -e 10

(gdb) r

Starting program: /usr/bin/time
./facetrain -n sunglasses.net -t straighteven_train.list
-1 straighteven_test1.list -2 straighteven_test2.list -e 10

Loading 'faces_4/an2i/an2i_straight_sad_sunglasses_4.pgm'...done
Loading 'faces_4/an2i/an2i_straight_happy_sunglasses_4.pgm'...done
. . .
Loading 'faces_4/tammo/tammo_straight_neutral_sunglasses_4.pgm'...done

Random number generator seed: 102194

80 images in training set
36 images in test1 set
40 images in test2 set

Creating new network 'sunglasses.net'
Training underway (going to 10 epochs)
Will save network every 100 epochs

0 0.0 46.25 0.0932394 41.6667 0.0923597 60 0.0854248
1 9.84071 50 0.0847417 44.4444 0.0849039 55 0.0795298
. . .
10 7.89897 90 0.040425 86.1111 0.039227 90 0.0366873

Saving 960x4x1 network to 'sunglasses.net'
0.19user 0.00system 0:00.19elapsed 97%CPU
```

Phase II - testing

```
peramanathan@ubuntu:~/Documents/ML/Sunglass_Recognizer/Program>
./facetrain -n sunglasses.net -T -1  straighteven_test1.list
 -2 straighteven_test2.list

Loading 'faces_4/an2i/an2i_straight_sad_open_4.pgm'...done
Loading 'faces_4/an2i/an2i_straight_neutral_open_4.pgm'...done
Loading 'faces_4/at33/at33_straight_happy_open_4.pgm'...done
. . .
Loading 'faces_4/tammo/tammo_straight_neutral_sunglasses_4.pgm'...done

Random number generator seed: 102194
0 images in training set
36 images in test1 set
40 images in test2 set
Reading 'sunglasses.net'
'sunglasses.net' contains a 960x4x1 network
Reading input weights...Done
Reading hidden weights...Done
0 0.0 0.0 0.0

No sunglasses on an2i (image index 1 in the list)
No sunglasses on at33 (image index 3 in the list)
. . .
Sunglasses on tammo (image index 40 in the list)

Performance - 97.5 0.0079711

Failed to classify the following images from the training set:

Failed to classify the following images from the test set 1:
an2i_straight_neutral_open_4.pgm - outputs 0.658

Failed to classify the following images from the test set 2:
an2i_straight_happy_open_4.pgm - outputs 0.856
```

Face recognizer

This is the sample output of face recognizer

Phase I - training

```
peramanathan@ubuntu:~/Documents/ML/Program$
gdb --args time ./facetrain -n facereg.net -t straighteven_train.list
-1 straighteven_test1.list -2 straighteven_test2.list -e 5

(gdb) r

Starting program: /usr/bin/time
./facetrain -n facereg.net -t straighteven_train.list
-1 straighteven_test1.list -2 straighteven_test2.list -e 5

Loading 'faces_4/an2i/an2i_straight_sad_sunglasses_4.pgm'...done
Loading 'faces_4/an2i/an2i_straight_happy_sunglasses_4.pgm'...done
. . .
Loading 'faces_4/tammo/tammo_straight_neutral_sunglasses_4.pgm'...done

Random number generator seed: 102194

80 images in training set
36 images in test1 set
40 images in test2 set

Creating new network 'facereg.net'
Training underway (going to 5 epochs)
Will save network every 100 epochs

0 0.0 3.75 0.560466 5.55556 0.573432 7.5 0.547354
1 28.5452 3.75 0.583131 2.77778 0.567345 2.5 0.585561
2 27.4214 6.25 0.536838 5.55556 0.530314 5 0.549853
3 26.153 11.25 0.533097 8.33333 0.530757 5 0.549421
4 25.0757 15 0.5211 11.1111 0.521454 10 0.541516
5 23.8496 17.5 0.498752 19.4444 0.499761 12.5 0.524567

Saving 960x20x5 network to 'facereg.net'
0.40user 0.02system 0:00.95elapsed 44%CPU
(0avgtext+0avgdata 6944maxresident)k
1568inputs+320outputs (1major+659minor)pagefaults 0swaps
```

Phase II - testing

```
peramanathan@ubuntu:~/Documents/ML/Program$ time
./facetrain -n facereg.net -T -1 straighteven_test1.list
-2 straighteven_test2.list

Loading 'faces_4/an2i/an2i_straight_sad_open_4.pgm'...done
Loading 'faces_4/an2i/an2i_straight_neutral_open_4.pgm'...done
. . .
Loading 'faces_4/tammo/tammo_straight_neutral_sunglasses_4.pgm'...done

Random number generator seed: 102194
0 images in training set
36 images in test1 set
40 images in test2 set
Reading 'facereg.net'
'facereg.net' contains a 960x20x5 network
Reading input weights...Done
Reading hidden weights...Done

Classified List of Images:
1. an2i (image index 1 in the list)
2. an2i (image index 2 in the list)
. . .
35. tammo (image index 36 in the list)

Performance - 97.2222 0.0302772

Classified List of Images:
1. an2i (image index 1 in the list)
2. an2i (image index 2 in the list)
. . .
37. tammo (image index 40 in the list)

Performance - 92.5 0.0462984

Failed to classify the following images from the training set:

Failed to classify the following images from the test set 1:
choon_straight_sad_sunglasses_4.pgm - outputs 0.709 0.205 0.117 0.133 0.567
```

```
Failed to classify the following images from the test set 2:
cheyer_straight_sad_open_4.pgm - outputs 0.066 0.533 0.426 0.172 0.519
choon_straight_happy_sunglasses_4.pgm - outputs 0.485 0.362 0.389 0.248 0.591
choon_straight_angry_sunglasses_4.pgm - outputs 0.550 0.240 0.175 0.162 0.567
```

Appendix B

For the completeness we are copying the part of the documentation available at: `http://www.cs.cmu.edu/afs/cs.cmu.edu/user/mitchell/ftp/faces.html`

Face images

The image data can be found in `/afs/cs/project/theo-8/faceimages/faces`. This directory contains 20 subdirectories, one for each person, named by userid. Each of these directories contains several different face images of the same person.
You will be interested in the images with the following naming convention:
`<userid>_<pose>_<expression>_<eyes>_<scale>.pgm`

- `<userid>` is the user id of the person in the image, and this field has 20 values: an2i, at33, boland, bpm, ch4f, cheyer, choon, danieln, glickman, karyadi, kawamura, kk49, megak, mitchell, night, phoebe, saavik, steffi, sz24, and tammo.
- `<pose>` is the head position of the person, and this field has 4 values: straight, left, right, up.
- `<expression>` is the facial expression of the person, and this field has 4 values: neutral, happy, sad, angry.
- `<eyes>` is the eye state of the person, and this field has 2 values: open, sunglasses.
- `<scale>` is the scale of the image, and this field has 3 values: 1, 2, and 4. 1 indicates a full-resolution image (128 columns $\times$ 120 rows); 2 indicates a half-resolution image (64×60); 4 indicates a quarter-resolution image (32×30). For this assignment, you will be using the quarter-resolution images for experiments, to keep training time to a manageable level.

Appendix C

Running `facetrain`

The `facetrain` executable is the one we use for train and test the ANN we built. The default `facetrain` which comes with the package doesn't classify faces or recognize sunglasses. So one can use the skeleton and do implement something he want such as sunglass recognizer or face recognizer or pose recognizer. **The following instructions copied from the documentation of this package:**

`facetrain` has several options which can be specified on the command line. This section briefly describes how each option works. A very short summary of this information can be obtained by running `facetrain` with no arguments.

`-n <network file>` - this option either loads an existing network file, or creates a new one with the given name. At the end of training, the neural network will be saved to this file.

`-e <number of epochs>` - this option specifies the number of training epochs which will be run. If this option is not specified, the default is 100.

`-T` - for test-only mode (no training). Performance will be reported on each of the three datasets specified, and those images misclassified will be listed, along with the corresponding output unit levels.

`-s <seed>` - an integer which will be used as the seed for the random number generator. The default seed is 102194 (guess what day it was when I wrote this document). This allows you to reproduce experiments if necessary, by generating the same sequence of random numbers. It also allows you to try a different set of random numbers by changing the seed.

`-S <number of epochs between saves>` - this option specifies the number of epochs between saves. The default is 100, which means that if you train for 100 epochs (also the default), the network is only saved when training is completed.

`-t <training image list>` - this option specifies a text file which contains a list of image pathnames, one per line, that will be used for training. If this option is not specified, it is assumed that no training will take place ($epochs = 0$), and the network will simply be run on the test sets. In this case, the statistics for the training set will all be zeros.

`-1 <test set 1 list>` - this option specifies a text file which contains a list of image pathnames, one per line, that will be used as a test set. If this option is not specified, the statistics for test set 1 will all be zeros.

`-2 <test set 2 list>` - same as above, but for test set 2. The idea behind having two test sets is that one can be used as part of the train/test paradigm, in which training is stopped when performance on the test set begins to degrade. The other can then be used as a "real" test of the resulting network.

Interpreting the output of `facetrain`

When you run `facetrain`, it will first read in all the data files and print a bunch of lines regarding these operations. Once all the data is loaded, it will begin training. At this point, the network's training and test set performance is outlined in one line per epoch. For each epoch, the following performance measures are output:

`<epoch> <delta> <trainperf> <trainerr> <t1perf> <t1err> <t2perf> <t2err>`

These values have the following meanings:

`epoch` is the number of the epoch just completed; it follows that a value of 0 means that no training has yet been performed.

`delta` is the sum of all δ values on the hidden and output units as computed during backprop, over all training examples for that epoch.

`trainperf` is the percentage of examples in the training set which were correctly classified.

`trainerr` is the average, over all training examples, of the error function $\frac{1}{2}\sum(t_i - o_i)^2$, where t_i is the target value for output unit i and o_i is the actual output value for that unit.

`t1perf` is the percentage of examples in test set 1 which were correctly classified.

`t1err` is the average, over all examples in test set 1, of the error function described above.

`t2perf` is the percentage of examples in test set 2 which were correctly classified.

`t2err` is the average, over all examples in test set 2, of the error function described above.

References

[1] Rein-Lien Hsu. *Face Detection and Modeling for Recognition*, Department of Computer Science & Engineering, Michigan State University, USA, 2002.

[2] Ralph Gross. *Face Databases*, Handbook of Face Recognition, S.Li, A.Jain, ed., Springer, 2005.

[3] Rabab M. Ramadan and Rehab F. AbdelKader. *Face Recognition Using Particle Swarm Optimization-BasedSelected Features*, International Journal of Signal Processing, Image Processing and Pattern Recognition Vol. 2, No. 2, June 2009.

[4] Ion Marques. *Face Recognition Algorithms*, Proyecto Fin de Carrera June 16, 2010.

[5] P.J. Phillips, W.T. Scruggs, A.J. O'Toole, P.J. Flynn, K.W. Bowyer, C.L. Schott, M. Sharpe. *FRVT 2006 and ICE 2006 Large-Scale Results*, 29 March 2007, NISTIR 7408, National Institute of Standards and Technology.

[6] J S. Bruner and R. Tagiuri. *The percepton of people. Handbook of Social Psycology*, 2(17), 1954.

[7] Amit Kagian, Gideon Dror, Tommer Leyvand, Daniel Cohen-Or, Eytan Ruppin. *A Humanlike Predictor of Facial Attractiveness*, School of Computer Sciences, Tel-Aviv University,Israel.

[8] Robert Mauro, Michael Kubovy. *Caricature and face recognition*, memory & Cognition 1992, 20(4), 433-440.

[9] F. Smach, M. Atri, J. Mitran and M. Abid. *Design of a Neural Networks Classifier for Face Detection*, Journal of Computer Science 2 (3): 257-260, 2006 ,ISSN 1549-3636.

[10] P.Latha, Dr.L.Ganesan, Dr.S.Annadurai. *Face Recognition using Neural Networks*, Signal Processing: An International Journal (SPIJ) Volume (3) : Issue (5).

[11] Ara V. Nefian , Monson H. Hayes III. *Hidden Markov Models for Face Recognition*, Center for Signal and Image Processing, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta,GA 30332.

[12] Face Recognition Project exercise. *http://www.cs.cmu.edu/~guestrin/Class/10701/projects.html*, 25 May,2011.

[13] Face Recognition Home Page. *http://face-rec.org*, 25 May,2011.

[14] Matlab source for face recognition *http://download.cnet.com/Face-Recognition-System/3000-2053_4-10420859.html*, 29 May 2011.

[15] Bonsor, Kevin, and Ryan Johnson. *How Facial Recognition Systems Work*, HowStuffWorks.com. *http://electronics.howstuffworks.com/gadgets/high-tech-gadgets/facialrecognition.htm*, 29 May 2011.

[16] PGM format *http://netpbm.sourceforge.net/doc/pgm.html*, 29 May 2011.