Gender Recognition using Facial Key Points
Project Report

Nicolas Lavderde Alfonso
BRS University of Applied Sciences

June 19, 2014

Supervise learning is normally use to solve problem of classification, clustering and regression. Its task aims to infers a function from labelled data. It achieves this by using a learning algorithm to evolve the inferred function, aiming to minimize the error between the predicted output and the real one (thus, a supervised process). Neural Network is a technique modelled after the central nervous system, and is useful for supervise learning problems. The report deals with the application of neural networks theory for a system capable to recognize the gender of a person using facial features.

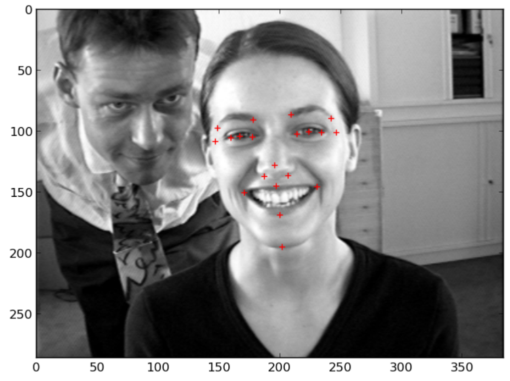


Figure 1: *Facial key points in one sample from the data.*

1 Data Set

A dataset from the BioID Face Database was used. The dataset consists in 1521 grey images with a resolution of 384x286 pixel. It has a frontal image of the face from 23 test subjects. Additional, text files for each one of the images are provided, with 20 manually placed points. The points correspond to x and y coordinates of some key features in the face, such as eye and eyebrows corners, chin, temple and lip edges.

2 Pre-processing

2.1 Labelling

The dataset was not labelled, so this task was done manually. A text file with the distribution of males and females was generated, with "0" for female and "1" for male. This file then was read by a function, which output is the text file "gender-labels.txt" with the column-vector labels for the complete dataset [see Snippet 1].

2.2 8-Metric

Going through the dataset, it is possible to notice that all the faces are not centred in the image following some rule. Thus, using only the identified key points would be useless, given that the key points came in form of coordinates in the images which depends in the way the photo was captured. Additionally, most of the images show the subjects making faces, making the key points again dependent of the configuration of the muscles.

For this reason, 8 metrics were chosen in order to reduce this dependency. They were chosen base on distances between points which have the less variation when the face configuration of the same subject changes. The decision dealt with the ”centralization” problem as well, because now we are talking about distances and no about coordinates.

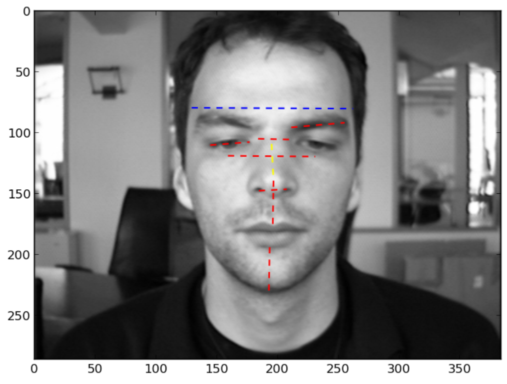


Figure 2: *Facial metrics chosen.*

The metrics were picked as well based on drawing theory for male faces and female faces. The distances chose represent the following proportion in the face: Face Width, Nose Height, Nose Width, Eye Separation, Average Eye Width, Brows Separation, Average Brow Width, Mouth-to-Chin Distance and Nose-to-Mouth Distance

2.3 Normalization

After dealing with the centralization of the faces problem, another one came up. Because the faces in all the images have different sizes, comparing distances between them makes no sense. To deal with this new problem, normalization was required. It was decided to chose the metric ”Face Width” to be the normalization value, and all the remain distances were divided by this value. A new dataset was then generated, which contains the 8 metric normalized values for the 1521 samples.

3 Approach: Neural Networks

The chosen approach was neural networks for classification of the data. This was decided because more knowledge about this technique was desired. After that, an analysis of the data was made to choose between a single perceptron network and a multilayer perceptron network.

3.1 Histogram Analysis

A histogram analysis was made to decide between this two models for the neural network. First, two subsets were created from the original data; one containing only male samples and the other female samples [Snippet 2].

From the histograms in Figure 3 and 4, it can be concluded that the data is not lineally separable. With this in mind, a multilayer perceptron network was chosen, given that the single perceptron uses a step function activation for the output, while the multilayer network uses smooth functions (such as log and tan) for the classification task.

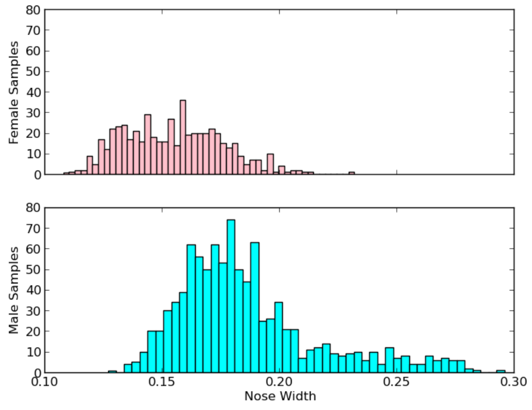


Figure 3: *Histogram for the Nose Width metric.*

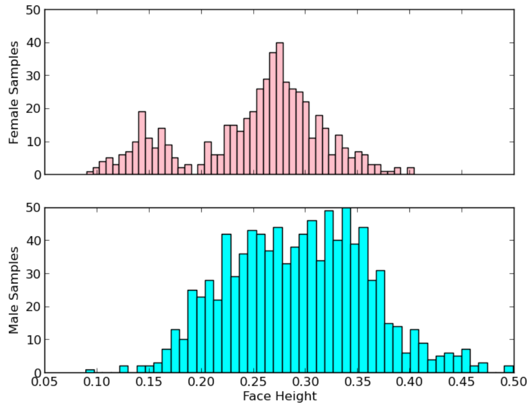


Figure 4: *Histogram for the Face Height metric.*

3.2 Multilayer Neural Network

Pybrain library was used to implement a multilayer neural network. A basic network was created, with 8 input neurons, one for each of the metrics, 2 output representing female and male gender, and 1 hidden layer with 4 neurons. A model of the network created can be seen in Figure 5, and the code in Snippet 3 deals with its creation.

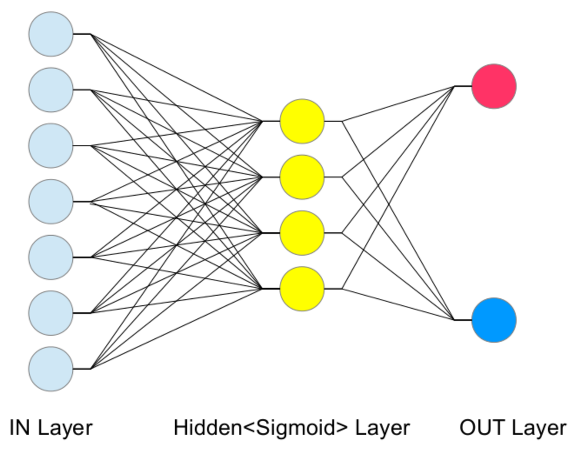


Figure 5: *Multilayer Neural Network proposed.*

4 Training

The next step after creating the network was to train it. For this, the dataset was divided in to two subsets: train set and test set. Once the network was trained with the train set, the test set was used to score the performance of the system. A back-propagation trainer was chosen, which implements the optimization algorithm of gradient descend to update the weights which convey the less error. The training code is showed in the Snippet 4.

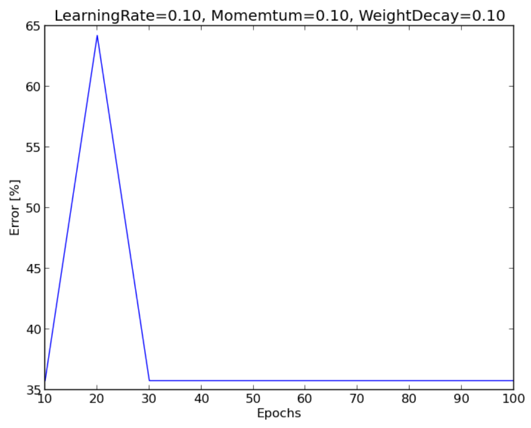


Figure 6: *Face classification error over the train set for 100 epoch of training.*

4.1 Train and Test Sets

As stated, the dataset was randomly divided into train set and test set, with a proportion of 70% : 30% respectively. In order to compare results, this subsets were save and used through the tweaking stage.

4.2 Tweaking

The trainer class form pybrain has some parameters which can be change in order to modify the performance of the network. They are the parameters learning rate, momentum and weight decay. Several tests were made to compare and found out which were the more optimal parameters for the overall performance of the system. This process threw the following conclusions for the optimal set of parameters: learning rate = 0.1, momentum = 0.1, weight decay = 0.001.

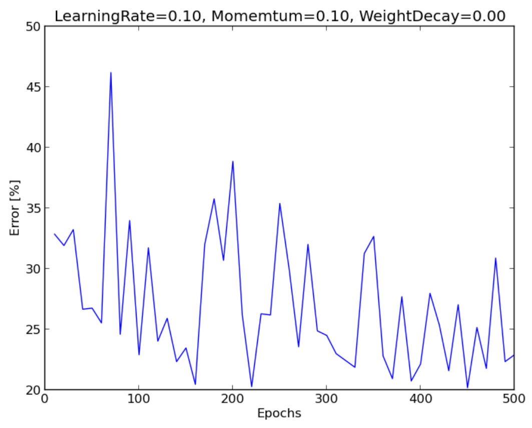
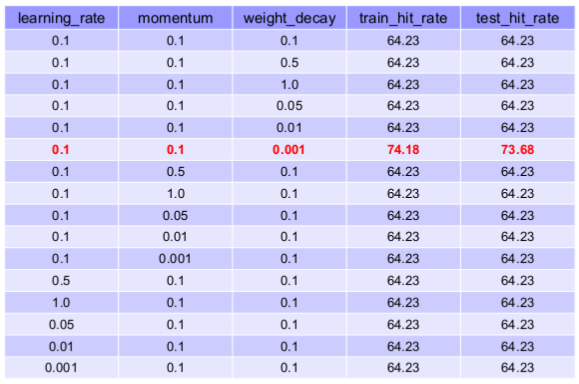


Figure 7: *Face classification error for 500 epoch with optimal parameter values*

5 Results

Plots of the error between the predicted label and the real one over the training set were taken. Figure 6 shows the error of the network in 100 epoch of training. The plot shows that the average error is around 36%, obtaining an accuracy rate of 64%. Later inspections showed that the networks was labelling every single face as male face, but given the gender distribution in the train set, the accuracy was still high (64%).

Figure 7 shows the error plot of the network for the optimal parameters and 500 training epochs. It is easy to observe the improvement of the performance of the network. Figure 8 contains the results of the training stage of the network for a given triple of values for the parameters.



learning_rate	momentum	weight_decay	train_hit_rate	test_hit_rate
0.1	0.1	0.1	64.23	64.23
0.1	0.1	0.5	64.23	64.23
0.1	0.1	1.0	64.23	64.23
0.1	0.1	0.05	64.23	64.23
0.1	0.1	0.01	64.23	64.23
0.1	**0.1**	**0.001**	**74.18**	**73.68**
0.1	0.5	0.1	64.23	64.23
0.1	1.0	0.1	64.23	64.23
0.1	0.05	0.1	64.23	64.23
0.1	0.01	0.1	64.23	64.23
0.1	0.001	0.1	64.23	64.23
0.5	0.1	0.1	64.23	64.23
1.0	0.1	0.1	64.23	64.23
0.05	0.1	0.1	64.23	64.23
0.01	0.1	0.1	64.23	64.23
0.001	0.1	0.1	64.23	64.23

Figure 8: *Face classification performance for a given set of parameters.*

6 Metric Relevance

6.1 Correlation

The correlation matrix for the 8 metrics of the face was created to analyse which features were dominant over the others. It was shown that for the bin 9, representing the gender of the individual, not all the metrics were as decisive as others. With that in mind, the network was tested using only the metrics with the highest correlation with the gender, and then it was tested with the remaining metrics (i.e. the less correlated).

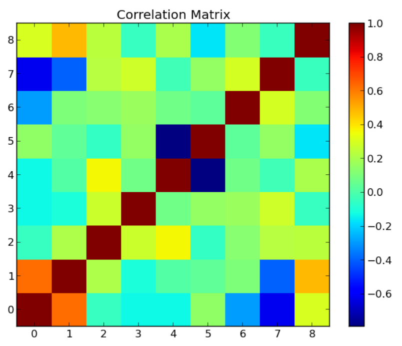
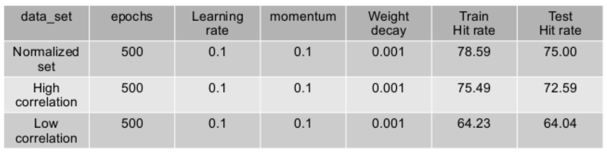


Figure 9: *Face classification performance for a given set of parameters.*

For the highest correlation metrics, the matrix showed that Nose Height, Nose Width, Eye Separation and Brows Separation were the defined distances which apparently tell apart male from female faces.

6.2 Performance

The same performance analysis was made for the two new subset of data: highest correlation set (4 inputs for the network) and lowest correlation set (4 remaining metrics as inputs). The results show that using only four metrics, the results are as good as using all the metrics. Reducing the inputs of the network, the computation time and memory is reduced.



data_set	epochs	Learning rate	momentum	Weight decay	Train Hit rate	Test Hit rate
Normalized set	500	0.1	0.1	0.001	78.59	75.00
High correlation	500	0.1	0.1	0.001	75.49	72.59
Low correlation	500	0.1	0.1	0.001	64.23	64.04

Figure 10: *Face classification performance for reduced metric values.*

7 Custom Multilayer Neuronal Network

7.1 Design

Finally, one last test was made. A custom multilayer neuronal network was created in order to improve the performance. This network consists in 13 inputs, 5 neuron hidden layer and 2 outputs.

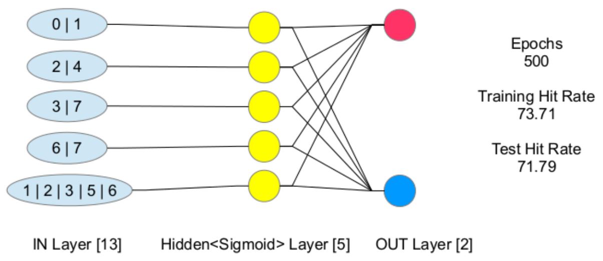


Figure 11: *Custom Multilayer Neural Network proposed.*

The inputs were chosen as pair of metrics with high correlation, and were feed to one hidden neuron. 4 pair were extracted from the correlation matrix, and the remain values were fed to the 5th hidden neuron. Then, the hidden neurons were connected to the outputs. A model of this structure can be seen in Figure 11 and Snippet 5.

7.2 Performance

The performance of the custom network was the same as for the previous network with the optimal parameters; i.e. the accuracy rate was around 72%. The time to train the network was incremented, as well as the memory consumption. Hence, the custom network was thrown away by lack of improvement over the basic network.

8 Conclusions

A system for recognition of gender using facial key points was implemented using a multilayer neural network. The results obtained showed that the system perform good, but not ideal as wanted.

The database was not suitable for the proposed task. Only 23 individuals were presented in the dataset, every one with around 55 different face captures. The individuals present different face emotions, which produce a difficulty when defining metrics. The size of the faces were not standard, so normalization was required. To really design a system for gender recognition, the face capture would need to be done with controlled conditions (same size, centred and with a relaxed expression).

To cope with some of the previous problems, computer vision techniques could be implemented. Transformation over the images could be done to centre the faces and normalize the distances. More features could be extracted using some techniques, which could lead to better metrics to define male face proportions and female face proportions.

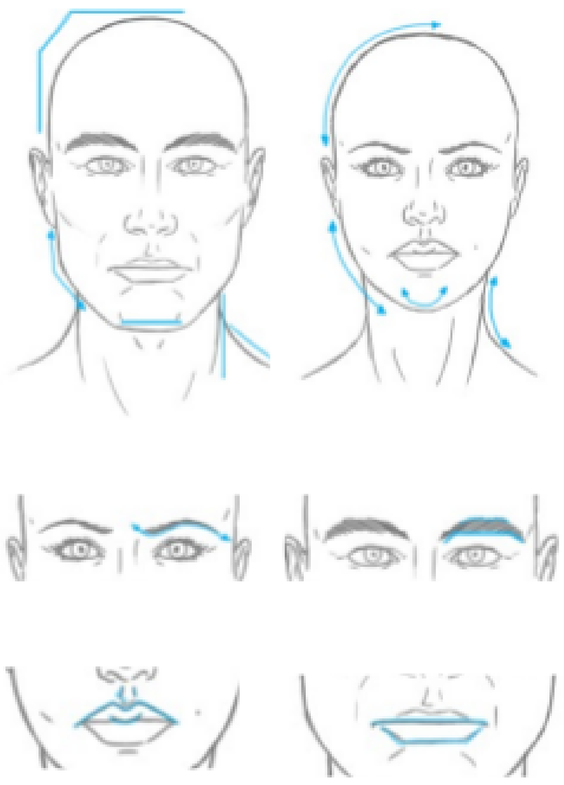


Figure 12: *Most defining face proportions.*

In some papers and drawing theory, the most defining features that difference a male face from a female face are the shape of the face, the thickness of the eyebrows and the mouth area. Given that most of the key points for this metrics were not presented in the database used, this theory could not be tested.

At the end, the design process of the network is very important. Given that no guidelines were found, this stage was done empirically. There should be relations between parameters in the network, such as number of layers, number of neurons and connections between neurons, and the distribution, domain and data in the set.

The fundamentals of how to use neural networks to solve machine learning problems were covered. The implementation of modules from the PyBrain library were covered as well. The overall performance of the system was good enough for a small project like this.

9 References

[1] BioID AG (2014). BioID face database. Retrieved from http://www.bioid.com

[2] Peters, Nastasia (2013). The Differences Between Male and Female Portraits. Design and Illustration. Retrieved from http://design.tutsplus.com/articles/the-differences-between-male-and-female-portraits–vector-14954

[3] Hammer, Alexandra (2012-2013). My Facial Feminisation Thesis, Part 5: Relative Proportions. Virtualffs. Retrieved from http://www.virtualffs.co.uk/My_Facial_Feminisation_Thesis_Part_5_Relative_Proportions.html

[4] Riedmiller, Martin. Machine Learning: Multi Layer Perceptrons. Retrieved from http://ml.informatik.uni-freiburg.de/_media/documents/teaching/ss09/ml/mlps.pdf

[5] Rashid, Sheikh F (2011). Document and Content Analysis. Lecture 08 - Classification with Neural Networks (MLP, autoMLP). Retrieved from https://sites.google.com/a/iupr.com/dia-course/lectures/lecture08-classification-with-neural-networks

Appendix A: Code

data_handler.py

```python
# Description:
# Generator of the gender labels based on the classification made
# manually of the data set. The file 'gender-distribution.txt' contains
# the distribution of gender, female=0 and male=1. The function will
# generate a file 'gender-labels.txt' with the correspondent labels.
def label_creator():
    gender_indices = np.genfromtxt("gender-distribution.txt", dtype=int, delimiter="
    label = []
    for gender,amount in gender_indices:
        for i in xrange(amount):
            label.append(gender)
    label = np.array(label)
    num_samples = label.shape[0]
    label = label.reshape(num_samples,1)
    print "Number of Samples: %.0f\n\n" % (num_samples)
    np.savetxt('gender-labels.txt', label, delimiter=",", fmt="%.0f")
    return label
```

Snippet 1. Label generator.

data_handler.py

```python
# Description:
# This function separates the data into two new subsets, one
# for females ('female_set.txt') and other for males
# ('male_set.txt'). This will be used for histogram analysis
# and for training and test sets separation.
def separate_gender(name_file):
    data = np.genfromtxt(name_file, delimiter=",")
    label = np.genfromtxt("gender-labels.txt")
    male = []
    female = []
    for i in xrange(data.shape[0]):
        if label[i] == 0:
            female.append(data[i])
        else:
            male.append(data[i])
    female = np.asarray(female)
    female_label = np.zeros((female.shape[0],1))
    male = np.asarray(male)
    male_label = np.ones((male.shape[0],1))

    female = np.hstack((female,female_label))
    male = np.hstack((male,male_label))

    print "Females: %s" % female.shape[0]
    print "Males: %s" % male.shape[0]

    np.savetxt('female_set.txt', female, delimiter=",", fmt="%.3f")
    np.savetxt('male_set.txt', male, delimiter=",", fmt="%.3f")
```

Snippet 2. Subset generation for female and male samples.

keypoint−gender−classification.py

```
def basic_network(num_in, num_hid, num_out):
    net = pyb.FeedForwardNetwork()
    inLayer = pyb.LinearLayer(num_in, name="Metrics")
    hiddenLayer = pyb.SigmoidLayer(num_hid, name="Hidden1")
    outLayer = pyb.LinearLayer(num_out, name="Gender")
    net.addInputModule(inLayer)
    net.addModule(hiddenLayer)
    net.addOutputModule(outLayer)

    in2hidden = pyb.FullConnection(inLayer, hiddenLayer)
    hidden2out = pyb.FullConnection(hiddenLayer, outLayer)
    net.addConnection(in2hidden)
    net.addConnection(hidden2out)
    net.sortModules()

    return net
```

Snippet 3. Built of the Multilayer Neural Network.

keypoint−gender−classification.py

```
    trainer = trainers.BackpropTrainer(net, dataset=train_d,
                                       learningrate=l_r ,momentum=mom, verbose=False

    error_array = []
    epoch_array = []
    for i in range(50):
        trainer.trainEpochs(10)
        trnresult = percentError( trainer.testOnClassData(),
                                  train_d['class'] )

        error_array.append(trnresult)
        epoch_array.append(trainer.totalepochs)
        #print "epoch: %4d" % trainer.totalepochs, \
        #  "  train error: %5.2f%%" % trnresult

    train_result,train_target = trainer.testOnClassData(dataset=train_d, return_targ

    plt.plot(epoch_array,error_array)
    plt.xlabel("Epochs")
    plt.ylabel("Error␣[%]")
    plt.title("LearningRate=%.2f,␣Momemtum=%.2f,␣WeightDecay=%.2f" % (l_r,mom,w_d))
    plt.savefig("figures/error2_3.png")
```

```python

    print trainer.testOnClassData(dataset=train_d)
    print net.activateOnDataset(train_d)
    train_hit_rate = Validator.classificationPerformance(train_result ,train_target)
    print "Training set hit rate: %.2f%%" % (train_hit_rate*100)
```

Snippet 4. Training and validation stage.

keypoint–gender–classification.py

```python
def custom_network(num_out):
    num_in = 13
    num_hid = 5
    net = pyb.FeedForwardNetwork()
    inLayer = pyb.LinearLayer(num_in, name="Metrics")
    hiddenLayer = pyb.SigmoidLayer(num_hid, name="Hidden1")
    outLayer = pyb.LinearLayer(num_out, name="Gender")
    net.addInputModule(inLayer)
    net.addModule(hiddenLayer)
    net.addOutputModule(outLayer)

    in2hiddenn1 = pyb.FullConnection(inLayer, hiddenLayer, inSliceTo=2, outSliceTo=1
    in2hiddenn2 = pyb.FullConnection(inLayer, hiddenLayer, inSliceFrom=2, inSliceTo=
    in2hiddenn3 = pyb.FullConnection(inLayer, hiddenLayer, inSliceFrom=4, inSliceTo=
    in2hiddenn4 = pyb.FullConnection(inLayer, hiddenLayer, inSliceFrom=6, inSliceTo=
    in2hiddenn5 = pyb.FullConnection(inLayer, hiddenLayer, inSliceFrom=8, outSliceFr
    hidden2out = pyb.FullConnection(hiddenLayer, outLayer)
    net.addConnection(in2hiddenn1)
    net.addConnection(in2hiddenn2)
    net.addConnection(in2hiddenn3)
    net.addConnection(in2hiddenn4)
    net.addConnection(in2hiddenn5)
    net.addConnection(hidden2out)
    net.sortModules()

    return net
```

Snippet 5. Built of the Custom Multilayer Neural Network.