CSC411 Project 1

Xiaoxue Xing

February 6, 2017

1 Part1

The database of the faces contains many actors and actresses faces, total image is more than 2000. Each person has around 200 pictures and the picture is about daily life, so ts a goof database to do face recognition. Some picture cant open or is an picture without people. This may because the database is old. But we can delete the broken image and ignore the wrong image since the number is too less. Most picture only contains the actor himselves, however, some picture contains the two people. But the cropped pic works fine, it still that actress. For example ferrera 175 contains two people, but after crop it only shows ferreras face. Except some wrong images, most bounding boxes are correct. However, some faces it only crop half. For example baldwin24, this one is a full face photo, but after crop, it only has half face. Sometimes, due to the actor post, the image only has the side face of the actor. This will make the recognition harder. Only a few image has completely wrong bounding boxes, for example bracco95. Before crop, this is a normal image contains bracco, after crop, it is not a face at all. But the rate for fail to crop the face is really low, about 6 pictures compared with total download about 1000 pictures. So this database is ideally for face recognition test.

2 Part2

First, I read all the file in the cropped folder and test if contains the name of the actor I need. If it contains the name, I will add it to the list. After find all image of one specific actor, I will random chose 120 images in the list, and save it to the folder training(100), test(10) and validation(10). Then do the same thing for another actor.

3 Part3

To compute the classifier. I assume the hader has a result of one and carell has a result of 0. First I construct the matrix x,y. thata * x = y . x is all the input of the image in training set , which is a 200*1024 matrix, y is the answer of the image, if the image is hader, then y equals to 1, if the image is carell, the image equals to 0. The computer will think the image is hader if the theta times image is greater than 0.5. Othwrwise, it is carell.

first I construct the matrix x,y. thata * x = y . x is all the input of the image in training set , which is a 200*1025 matrix, y is the answer of the image, if the image is hader, then y equals to 1, if the image is carell, the image equals to 0. I minimized the cost function Then I use the graddescent to reduce to smallest cost to find that theta. The code is as follows construction function is as follows. minimized the cost function.

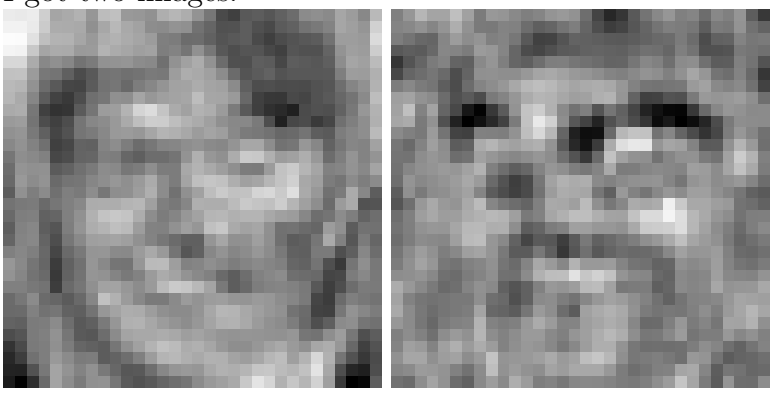
$$\sum_{i=1}^{m}((\theta(x(i))y(i))^2)$$

The value of the cost function for validation set is 1.3739085664044663, while for the test set is 2.9273159641867927.
To make the function run, we need to set theta and alpha really small. I set theta as 0 and alpha as 0000010. The performance on validation set is 0.9 and test set is 0.85. When the alpha is really large, the graddescent cant run.
To compute the theta, we minimize the cost function using the code:

```
def grad_descent(f, df, x, y, init_t, alpha):
    EPS = 1e-10
    prev_t = init_t-10*EPS
    t = init_t.copy()
    max_iter = 30000
    ite  = 0
    while norm(t - prev_t) >  EPS and ite < max_iter:
        prev_t = t.copy()
        t -= alpha*df(x, y, t)
        if ite % 5000 == 0:
            print "Iter", ite
            print "Gradient: ", df(x, y, t), "\n"
        ite += 1
    return t
```

4 Part4

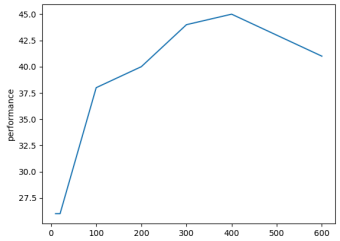
I got two images.



The left one trained with 4 images and right one trained with 200 images.

5 Part5

The performance is gradually going up when the training set is becoming larger. And then, at the training set is 400,the performance is the best. The best preformance is Then the performance is going down while the training set is going up. This is the overfitting.



6 Part6

6.1 a

see picture

6.2 b

part6(b) X is a 1025*600 vector, theta is 1025*6, y is 6*600 600 is the training set. theta*x-y is a matrix od 6*600 X*(theta*x-y) is a matrix of 2015*6

6(a) $X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ $y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$ $\theta = \begin{bmatrix} \theta_{11} & \cdots & \theta_{1k} \\ \theta_{21} & \cdots & \theta_{2k} \\ \vdots & & \vdots \\ \theta_{n1} & \cdots & \theta_{nk} \end{bmatrix}$

$J(\theta) = \sum_i \sum_j [(\theta^T x^{(i)} - y^{(i)})_j^2]$ Since $R^2 = R^T R$

$= \sum_i \sum_j [\sum_{t=1}^{k} (\sum_{s=1}^{n} \theta_{st} x_s^{(i)} y_t^{(i)})^2]_j$

$\frac{\partial J(\theta)}{\partial \theta_{pq}} = \sum_i \sum_j [\sum_{t=1}^{k} 2(\sum_{s=1}^{n} \theta_{st} x_s^{(i)} - y_t^{(i)}) \frac{\partial (\sum \theta_{st} x_s^{(i)} - y_t^{(i)})}{\partial \theta_{pq}}]_j$

$= \sum_i \sum_j [2(\sum_{s=1}^{n} \theta_{sq} x_s^{(i)} - y_q^{(i)}) x_p^{(i)}]_j$

$= 2 \sum_i \sum_j [(\sum_{s=1}^{n} \theta_{sq} x_s^{(i)} - y_q^{(i)}) x_p^{(i)}]_j$

$= 2 \sum_i \sum_j [(\theta^T x^{(i)} - y^{(i)})_q \cdot x_p^{(i)}]_j$

$= 2 [\theta^T \sum_i \sum_j x_j^{(i)} - \sum_i \sum_j y_j^{(i)}]_q \cdot \sum_i \sum_j (x_p^{(i)})_j$

Set $X = \sum_i \sum_j x_j^{(i)}$ as a $n \times 1$ vector

and $Y = \sum_i \sum_j y_j^{(i)}$ as a $k \times 1$ vector

$\frac{\partial J(\theta)}{\partial \theta_{pq}} = 2[\theta^T X - Y]_q X_p$

6.3 c

The code for f, df and graddescent:

```
def f(x, y, theta):
    x = np.transpose(x)
    x = vstack( (ones((1, x.shape[1])), x))
    return sum( (y - dot(x.T,theta.T)) ** 2)

def df(x, y, theta):
    x = np.transpose(x)
    x = vstack( (ones((1, x.shape[1])), x))
    return 2*(dot(x,(dot(theta,x)).T-y)).T
```

```
def grad_descent(f, df, x, y, init_t, alpha):
    EPS = 1e-10
    prev_t = init_t-10*EPS
    t = init_t.copy()
    max_iter = 30000
    ite  = 0
    while norm(t - prev_t) >  EPS and ite < max_iter:
        prev_t = t.copy()
        t = t - alpha*df(x, y, t)
        if ite % 5000 == 0:
            print "Iter", ite
            print "Gradient: ", df(x, y, t), "\n"
        ite += 1
    return t
```

6.4 d

```
>>> theta = np.zeros(shape=(6,1025))
>>> theta1 = theta
>>> theta1[0,0] = 0.0001
>>> f(lst[0], lst[1], theta1)
599.980006
>>> thet2 = theta
>>> thet2[0,0] = -0.0001
>>> f(lst[0], lst[1], thet2)
600.02000599999997
>>> (599.980006-600.02000599999997)/0.0002
-199.9999999998181
>>> df(lst[0], lst[1], theta)
array([[-200.12      ,  -56.42345534,  -53.07434446, ...,  -57.34941148,
         -54.00717518,  -53.63270208],
       [-200.        ,  -58.44794275,  -55.18566275, ...,  -56.11290353,
         -54.79593804,  -51.75448941],
       [-200.        , -103.4697051 , -102.9762902 , ...,  -89.3063302 ,
         -92.61338431,  -94.27704784],
       [-200.        ,  -71.24603059,  -71.41059373, ...,  -78.84176784,
         -79.66883137,  -82.27714353],
       [-200.        ,  -64.88990745,  -58.36168471, ...,  -86.00991608,
         -85.19114196,  -85.21801569],
       [-200.        ,  -74.90360784,  -70.86934824, ...,  -88.72047059,
         -88.8227749 ,  -85.58513098]])
```

-199.9999999998181 is really near -200.12 , therefore, the df works fine.

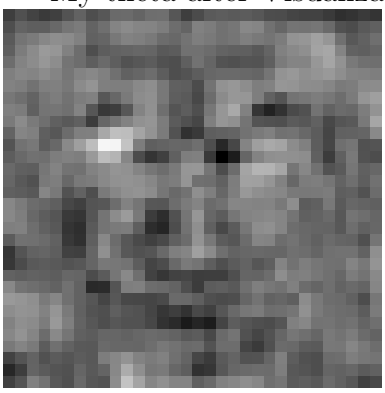
7 Part7

The performance is really good, for the validation set is 0.866666666667, while for the test set is 0.883333333333. I choose theta as 6*1025 matrix and alpha as 0.0000010. Since the smaller the alpha , the more accurate the result is. Theta should also be smaller to get a good theta.

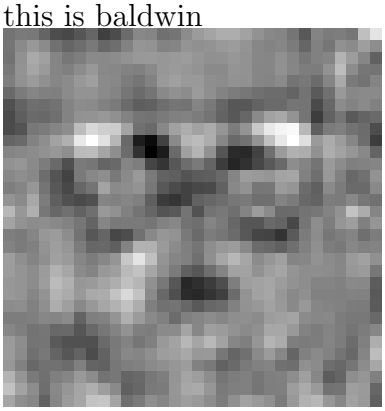
8 Part8

I think the first image is hader, second one is chenoweth. However, after I tried theta*image, it shows first one is hader, but the second image is still hader(haderhas has the highest result, but chenoweth also has a really high result).

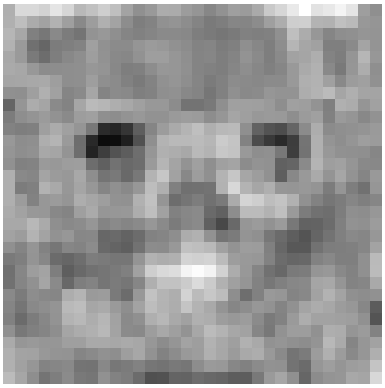
My theta after Visualization is



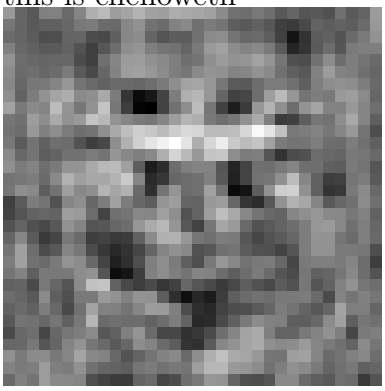
this is baldwin



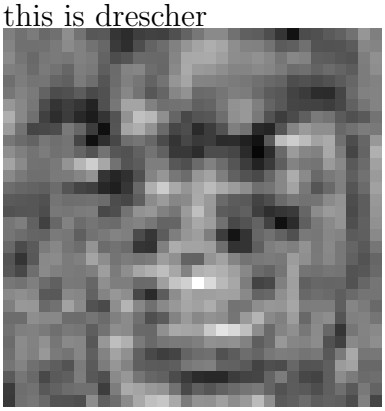
this is carell



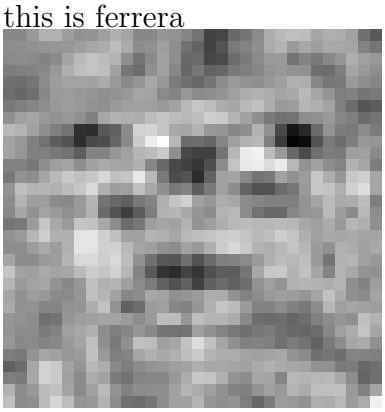
this is chenoweth



this is drescher



this is ferrera



this is hader