CSC411 Project1 Report

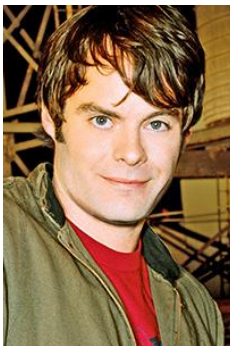
Yitian Ding

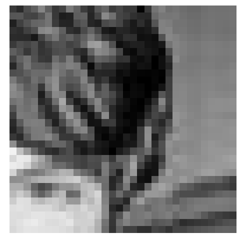
Febraury 2017

1 Part 1

Most of the bounding boxes are accurate, but some are not(like the example hader115 below) and the cropped-out faces are aligned with each other.

Examples are below:





Example of Hader115

Example of Chenoweth1

Example of vartan1

2 Part 2

First, I use a method os.makedirs() to create three folders including training, validation and test. Then, I the method shutil.copy() to copy images from my uncropped folder into three folders I created in the first step. This is my code:

```
def part2_separate(names):
    if not os.path.exists("validation_set"):
        os.makedirs("validation_set")
    else:
        shutil.rmtree("validation_set")
        os.makedirs("validation_set")
    if not os.path.exists("training_set"):
        os.makedirs("training_set")
    else:
        shutil.rmtree("training_set")
        os.makedirs("training_set")
    if not os.path.exists("test_set"):
        os.makedirs("test_set")
    else:
        shutil.rmtree("test_set")
        os.makedirs("test_set")

    image_files=os.listdir("cropped/")

    for n in names:
        name=n.split()[1].lower()
        count = 0
        for img in image_files:
            if img.startswith(name):
                if count<100:
                    shutil.copy("cropped/"+img,"training_set/")
                elif count<110:
                    shutil.copy("cropped/"+img,"validation_set/")
                elif count<120:
                    shutil.copy("cropped/"+img,"test_set/")
                count += 1
```

3 Part 3

Cost function I minimized:

$$J(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(y - X\theta)_i^2$$

This is my code for f and df:

```
def f(x, y, theta):
    return sum( (y - dot(x,theta)) ** 2)/(x.shape[0]*2)

def df(x, y, theta):
    return -dot(x.T,(y-dot(theta.T,x.T)))/(x.shape[0])
```

The value of the cost function on the training set: 100 and 100
The value of the cost function on the validation set: 10 and 7
The cost of the training set: 0.00139735275566
The cost of the validation set: 0.0550477179273

The system must work well with a proper α, which cannot be too big or too small. In order to get a α which can meet our requirement, we have to try mutiple values of α until our requirement is met. That is, θ will not move too much each time and also will not move too little each time.

Below is the gradient descent function I use:

```
def grad_descent(f, df, x, y, init_t, alpha):
    EPS = 1e-8
    prev_t = init_t -10*EPS
    t = init_t.copy()
    max_iter = 30000
    iter  = 0
    while norm(t - prev_t) >  EPS and iter < max_iter:
        prev_t = t.copy()
        t -= alpha*df(x, y, t)
        iter += 1
    return t
```

And this is my function for part 3:

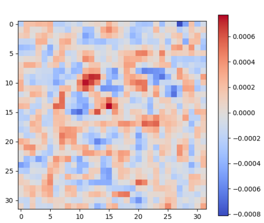
```
def part3(name1,name2,filename):
    x,y=get_data(name1,name2,filename,100)
    theta0=array([0.]*1025)
    return x,grad_descent(f, df, x, y, theta0, 1e-7),y
```

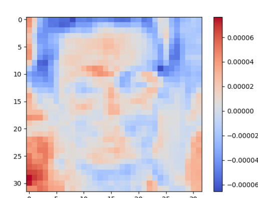
And this is the helper function which I use to get data in the function part3:

```
def get_data(names1,names2,filename,size):
    alltraining=os.listdir(filename)
    data=[]
    y=[]
    for name1 in names1:
        counter=0
        for  img in alltraining:
            if counter<size and img.startswith(name1):
                person1_data=imread(filename+"/"+img)
                data.append([1]+array(person1_data).flatten().tolist())
                y+=[1]
                counter+=1
    for name2 in names2:
        counter=0
        for img in alltraining:
            if counter<size and img.startswith(name2):
                person2_data=imread(filename+"/"+img)
                data.append([1]+array(person2_data).flatten().tolist())
                y+=[0]
                counter+=1
    all_data=array(data)
    return all_data,array(y)
```

4 Part 4

A 32×32 image of θ is presented:





The left figure is from full training and the right figure is from two image per person.

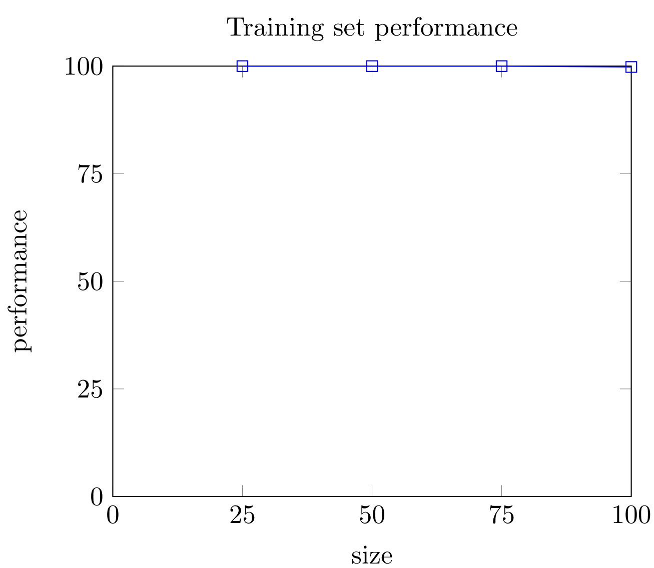
5 Part 5

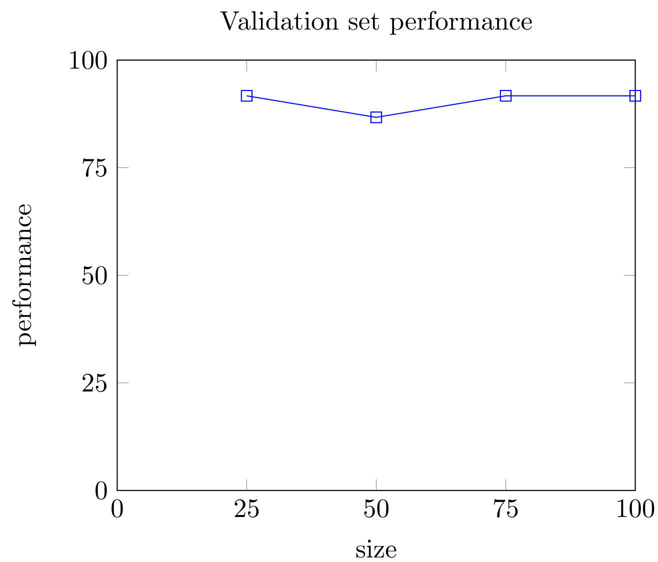
The performance is below:
100 iamges for each person:
Training set: 299 and 300, 99.8%
Validation set: 28 and 27, 91.7%
act_test: 238 and 263, 83.5%

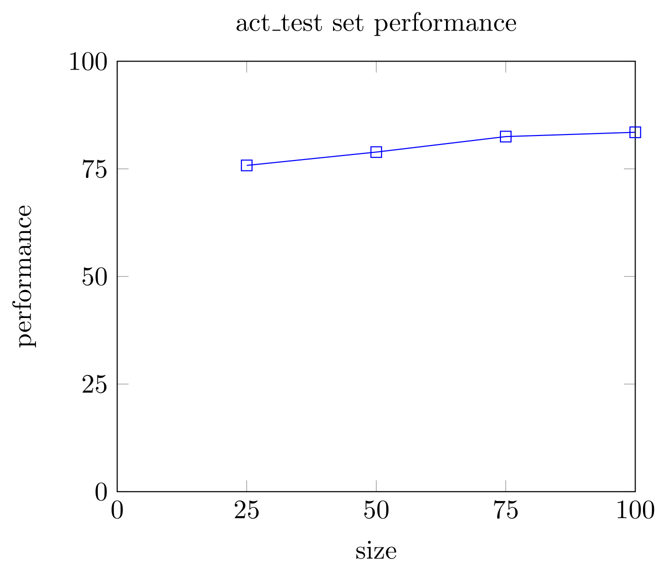
75 iamges for each person:
Training set: 225 and 225, 100%
Validation set: 28 and 27, 91.7%
act_test: 228 and 267, 82.5%

50 iamges for each person:
Training set: 150 and 150, 100%
Validation set: 28 and 24, 86.7%
act_test: 232 and 241, 78.9%

25 iamges for each person:
Training set: 75 and 75, 100%
Validation set: 29 and 26, 91.7%
act_test: 236 and 219, 75.8%




Validation set performance
100
75
50
25
0
performance
0
25
50
75
100
size


act_test set performance
100
75
50
25
0
performance
0
25
50
75
100
size

6 Part 6

6.1 a)

Let $Z(\theta) = \sum_j (\theta^T \vec{x} - \vec{y})_j^2$ be the cost of $n \times 1$ vector X which the size of the training set is L.

$$\frac{\alpha Z}{\alpha \theta_{pq}} = \frac{\alpha Z}{\alpha \theta^T_{qp}} = \frac{\alpha Z}{\alpha \theta^T x_q} \cdot \frac{\alpha \theta^T x_q}{\alpha \theta^T_{qp}}$$

$$\frac{\alpha Z}{\alpha \theta^T x_q} = \frac{\alpha [(\theta^T x_1 - y_1)^2 + \cdots + (\theta^T x_q - y_q)^2 + \cdots + (\theta^T x_k - Y_k)^2]}{\alpha \theta^T x_q}$$

$$= \frac{\alpha [(\theta^T x_q - Y_q)^2]}{\alpha \theta^T x_q}$$

$$= 2(\theta^T x_q - Y_q)$$

$$\frac{\alpha \theta^T x_q}{\alpha \theta^T_{qp}} = \frac{\alpha (\theta^T_{q1} \cdot x_1 + \cdots + \theta^T_{qp} \cdot x_p + \cdots + \theta^T_{qn} \cdot x_n)}{\alpha \theta^T_{qp}}$$

$$= x_p$$

$$\frac{\alpha Z}{\alpha \theta_{rq}} = \frac{\alpha Z}{\alpha \theta^T x_q} \cdot \frac{\alpha \theta^T x_q}{\alpha \theta^T_{qp}} = 2(\theta^T x_q - Y_q) \cdot x_p$$

$$\frac{\alpha J}{\alpha \theta_{pq}} = \frac{\alpha J}{\alpha \theta_{qp}} = \sum \frac{\alpha Z}{\alpha \theta^T_{qp}} = \sum_i 2(\theta^T x^{(i)}{}_q - y^{(i)}{}_q) \cdot x_p^{(i)}$$

6.2 b)

$$\frac{\partial J(\theta)}{\partial \theta}$$

$$= \frac{\partial(\theta^T X - Y)^T(\theta^T X - Y)}{\partial \theta}$$

$$= \frac{\partial(X^T \theta \theta^T X - Y^T \theta^T X - X^T + Y^T Y)^T(\theta^T X - Y)}{\partial \theta}$$

$$= \frac{\partial(X^T \theta \theta^T X - 2Y^T \theta^T X + Y^T Y}{\partial \theta}$$

$$= \frac{\partial(X^T (\theta)^2 X - 2Y^T \theta^T X + Y^T Y}{\partial \theta}$$

$$= 2XX^T\theta - 2XY^T$$

$$= 2X(X^T\theta - Y^T)$$

$$= 2X(\theta^T X - Y)^T$$

Let n be the number of pixels+1, m be the number of training datas, and k be the number of lables, then the dimension of X is (n,m), the dimension of θ^T is (k,n), and the dimension of Y is (k,m).

6.3 c)

The code for my newf and newdf:

```
def newf(x,y,theta):
    return sum((dot(theta.T,x)-y)**2)

def newdf(x,y,theta):
    return (dot(x.T,(dot(x,theta)-y))*2)
```

6.4 d)

In this part, I check for three pairs of data.
First, I check 1st row and 2nd column:
Limit: -3.56727620776
Gradient descent: -3.56727618536
First, I check 1st row and 1st column:
Limit: -1.58994404309
Gradient descent: -1.5894405675
First, I check 0th row and 2nd column:
Limit: -3.32390110991
Gradient descent: -3.32390111448
From above we find that the results are almost the same using two different method.
The code to achieve this is below:

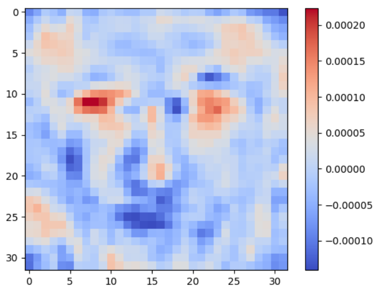
```
def limit(x,y,theta,a,b):
    h=1e-8
    list_h =[[0,0,0],[0,0,0],[0,0,0]]
    list_h[a][b]=h
    return (newf(x,y,theta+array(list_h))-newf(x,y,theta-array(list_h)))/(2*h)

def part6d(a,b):
    test_x=array(([3,2,1],[2,1,2],[0,1,1],[2,1,0]))
    test_y=array(([1,0,0],[0,1,0],[1,0,1],[0,0,1]))
    init_theta=array([[0.]*3]*3)
    theta=grad_descent(newf,newdf,test_x,test_y,init_theta,1e-7)
    print "using limit"
    print limit(test_x,test_y,theta,a,b)
    print "using gradient descent"
    print newdf(test_x,test_y,theta)[a][b]
```

7 Part 7

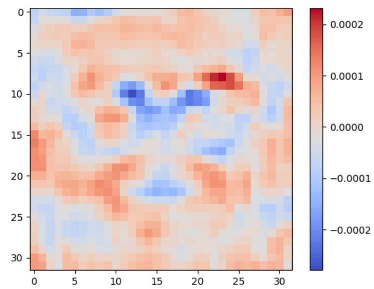
After I run gradient descent on the set of six actors act in order to perform face recognition, I got performance below:
Training set performance: [91, 93, 94, 96, 85, 88], 91.2%
Validation set performance: [10, 6, 10, 9, 8, 10], 88.3%

For gradient descent, I choose parameter f, df, x, y, init_t, and α. These parameter make sense because we have to use them: f is the cost function, df is derivative $df/d\theta$, x contains all the training data, y contains all the labels of images, and α is used to move θ properly.

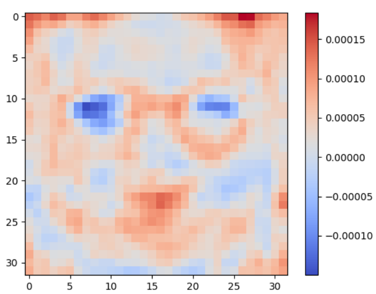
8 Part 8



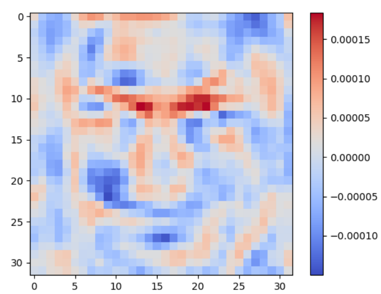
Baldwin



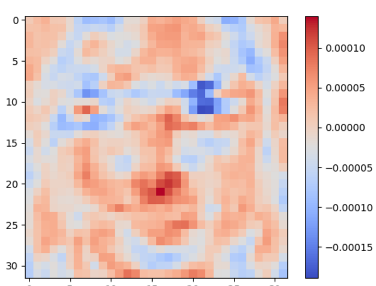
Carell



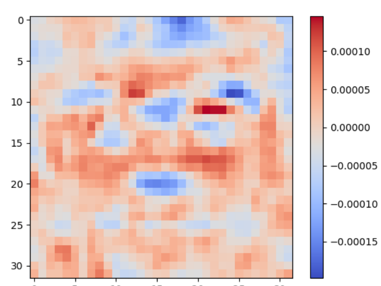
Chenoweth



Descher



Ferrera



Hader