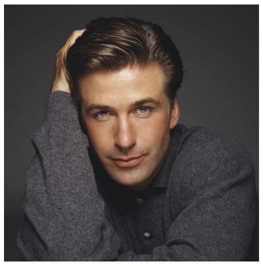
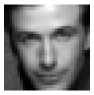
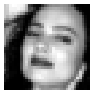
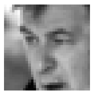
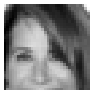
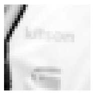
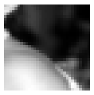
CSC411: Assignment 1

Due on Monday, January 29, 2018

Quan Zhou

January 29, 2018

Table 1: 4 Examples of the images in the dataset and cropped out faces

	uncropped	processed
example 1		
example 2		
example 3		
example 4		
example 5		
example 6		

Part 1

Describe the dataset of faces. In particular, provide at least three examples of the images in the dataset, as well as at least three examples of cropped out faces. Comment on the quality of the annotation of the dataset: are the bounding boxes accurate? Can the cropped-out faces be aligned with each other?

The dataset that we are using for this assignment is a group of grayscale images of actors' and actresses' faces. The original images includes a subset of male actors and a subset of female actors were all downloaded from FaceScrub database. The images of the faces were cropped out, resized to 32x32 and then converted to grayscale.

The quality of the annotation of the dataset is pretty good. The bounding boxes are accurate most of the time, as example 1-2 in table 1. The cropped out faces can be aligned with each other in general cases. However, depending on the angles and the lighting of the original images, some parts of the faces might be cut off slightly in cropped out images. See example 3-4 in table 1. In some rare cases, the bounding box are completely off the face. In example 5-6, a part of background and a part of shoulder were cropped out. These kind of images can not be aligned with each other.

Part 2

Separate the dataset into three non-overlapping parts: the training set (100 UPDATE: 70 face images per actor), the validation set (10 face images per actor), and the test set (10 face images per actor). For the report, describe the algorithm that you used to do that (any algorithm is fine, but you should use code to split up your dataset). The training set will contain faces whose labels you assume you know. The test set and the validation set will contain faces whose labels you pretend to not know and will attempt to determine using the data in the training set.

First of all, I went through entire dataset (”cropped/”) and take all images with the same name as the input put into a list called options. In order to make sure we will have three non-overlapping set for each actor/actress, I need $(70 + 10 + 10 = 90\ )$ images for each actor/actress. Random module was used to shuffle the option list. A training set with 70 images, a validation set with 10 images and a test set with 10 images will be returned after all.

Part 3

Use Linear Regression in order to build a classifier to distinguish pictures of Alec Baldwin form pictures of Steve Carell. In your report, specify which cost function you minimized. Report the values of the cost function on the training and the validation sets. Report the performance of the classifier (i.e., the percentage of images that were correctly classified) on the training and the validation sets. In your report, include the code of the function that you used to compute the output of the classifier. In your report, describe what you had to do in order to get the system to work. For example, the system would not work if the parameter alpha is too large. Describe what happens if alpha is too large, and how you figure out what to set alpha to. Describe the other choices that you made in order to make the algorithm work.

In order to build a classifier that distinguish pictures of Alec Baldwin form pictures of Steve Carell, Linear Regression was used. Gradient Descent from the lecture was used to minimized the cost function in order to find parameter theta.

The value of the cost function on the training sets is 1.29761628609. The value of the cost function on validation sets is 1.36655879443. The performance of the classifier on the training sets is 100.0%. The performance of the classifier on the validation sets is 95.0%.

The system does not work if the parameter alpha is too large. I changes alpha by 5E-1 at a time and i figured out that 5E-6 will probably be the best number. 5E-5 won't work and 5E-8's result is not so good. In the algorithm of splitting the sets, random.seed(n) input n also affects the performance of the classifier on both training sets and validation sets. The more iterations it does, the better perform the system give out.

The code of the function that you used to compute the output of the classifier as below:

Listing 1: grad_descent and cost function

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


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


def grad_descent(f, df, x, y, init_t, alpha):
    EPS = 1e-5
    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).reshape(1025, 1)

        if iter % 500 == 0:
            print "Iter: ", iter
            print "x = (%.2f, %.2f, %.2f), f(x) = %.2f" % (t[0], t[1], t[2], f(x, y, t))
            print "Gradient: ", df(x, y, t), "\n"
        iter += 1
    return t
```

Listing 2: part3

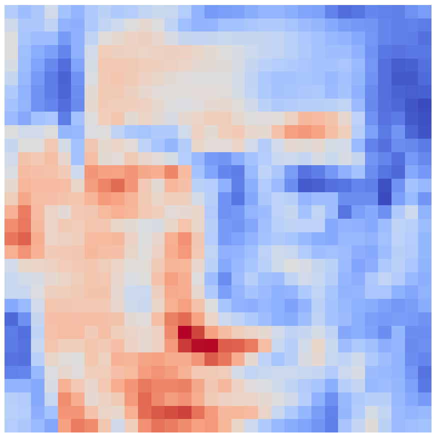
```
def part3():
    training_set, validation_set, test_set = {}, {}, {}
    # actors = ["baldwin", "carell"]
    # ==Part2(name)==
    training_set["baldwin"], validation_set["baldwin"], test_set["baldwin"] = part2("baldwin")
    training_set["carell"], validation_set["carell"], test_set["carell"] = part2("carell")

    #get_imgs(input, set_size)
    b_tr, b_v, bt = get_imgs(training_set["baldwin"], 70), get_imgs(validation_set["baldwin"], 10), get_imgs(test_set["baldwin"],
    c_tr, c_v, ct = get_imgs(training_set["carell"], 70), get_imgs(validation_set["carell"], 10), get_imgs(test_set["carell"], 10

    training_smg, validation_smg, test_smg = imgs_sets(b_tr, b_v, bt, c_tr, c_v, ct)

    training_y, validation_y, test_y = imgs_sets_y(70), imgs_sets_y(10), imgs_sets_y(10)

    theta = np.random.rand(1025, 1) * (1E-5)
    t = grad_descent(f, df, training_smg.T, training_y, theta, 5E-6)

    # this is for part 4 ===
    imsave('part3.jpg', reshape(t[1:], [32, 32]), cmap=cm.coolwarm)

    print "values of the cost function on the training: ", f(training_smg.T, training_y, t)
    print "values of the cost function on the validation: ", f(validation_smg.T, validation_y, t)

    training_correct = training_y[0]
    h = dot(t.T, vstack((ones((1, training_smg.T.shape[1])), training_smg.T)))
    training_prediction = []
    train_len = len(training_correct)

    for i in range(train_len):
        if h[:, i] < 0.5:
            training_prediction.append(0)
        else:
            training_prediction.append(1)
    print("performance  on training:", percentage_cal(training_correct, training_prediction, 140))

    validation_correct = validation_y[0]
    h = dot(t.T, vstack((ones((1, validation_smg.T.shape[1])), validation_smg.T)))
    validation_prediction = []
    valid_len = len(validation_correct)

    for i in range(valid_len):
        if h[:, i] < 0.5:
            validation_prediction.append(0)
        else:
            validation_prediction.append(1)
    print("performance on validation:", percentage_cal(validation_correct, validation_prediction, 20))
```

Part 4

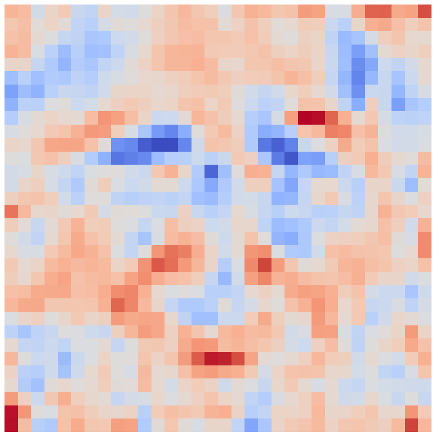
In Part 4(a), you need to display whatever image Gradient Descent produced. In this part, you should experiment in order to produce both kinds of visualizations. Report on how to obtain both a visualization that contains a face and a visualization that does not, using the full training set.

I replaced the training sets size with 2 and 132 to compute the result images as follows:

This is computed with a 2-image-training set.



This is computed with a 132-image-training set, which mean the full dataset

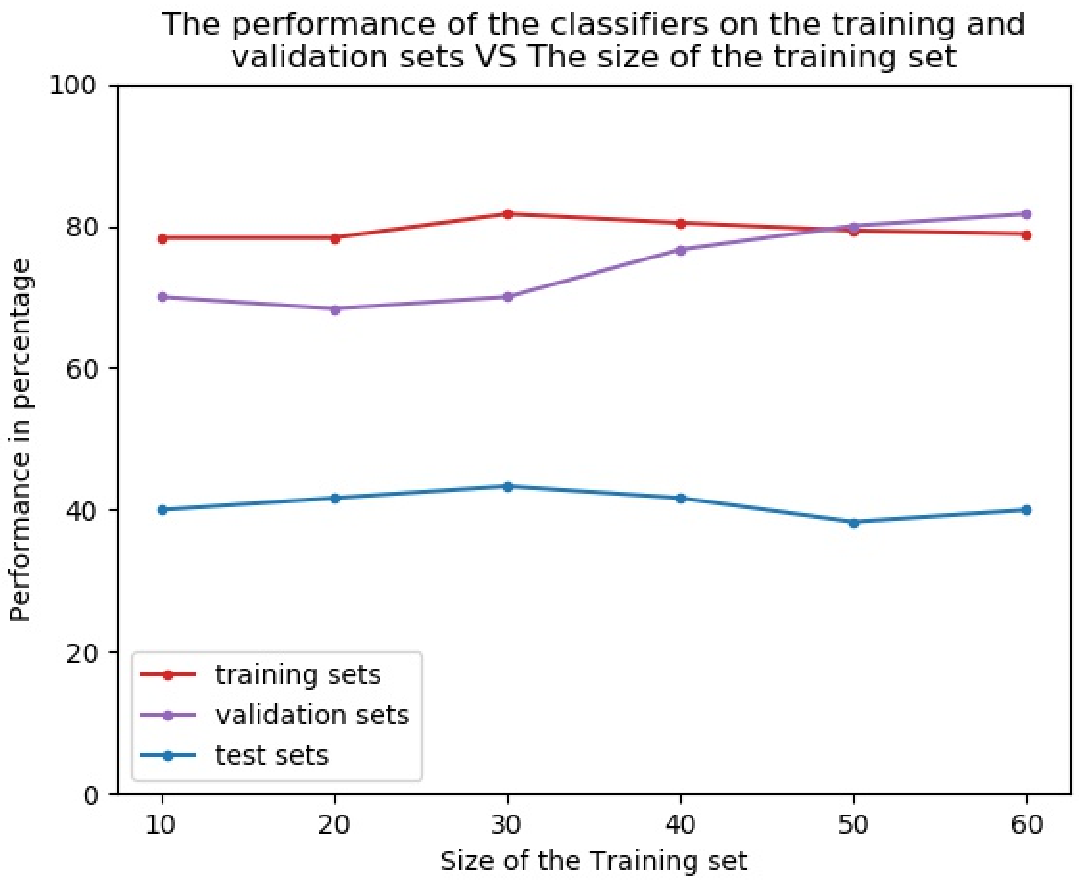


The code in part 3 was used. Details see faces.py

Part 5

In this part, you will demonstrate overfitting. Build classifiers that classify the actors as male or female using the training set with the actors from act =['Lorraine Bracco', 'Peri Gilpin', 'Angie Harmon', 'Alec Baldwin', 'Bill Hader', 'Steve Carell'] and using training sets of various sizes. Plot the performance of the classifiers on the training and validation sets vs the size of the training set.

I used similar idea of finding performance results in part3. However, 6 people per list is computed. Therefore, I coded a new function part2-lst whcih basically does the same thing as part2(), but instead of taking only one name, it takes a list of names. Returns a training set, a validation set and a test set for 6 people as a group.



Part 6

6(a): Compute $\frac{\partial J}{\partial \theta_{pq}}$

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

$$\frac{\partial J(\theta)}{\partial \theta_{pq}} = \frac{\partial}{\partial \theta_{pq}} \sum_i (\sum_j (\theta^T x^{(i)} - y^{(i)})_j^2)$$

$$\frac{\partial J(\theta)}{\partial \theta_{pq}} = \sum_i (\sum_j \frac{\partial}{\partial \theta_{pq}} (\theta^T x^{(i)} - y^{(i)})_j^2)$$

$$\frac{\partial J(\theta)}{\partial \theta_{pq}} = 2 \sum_i \sum_j ((\theta^T x^{(i)} - y^{(i)})_q \times x_p^{(i)})_j$$

$$\frac{\partial J(\theta)}{\partial \theta_{pq}} = 2 \sum_i \sum_j ((\theta_{p,q}^T x_p^{(i)} - y_p^{(i)})_q \times x_p^{(i)})_j$$

$$\frac{\partial J(\theta)}{\partial \theta_{pq}} = 2(\theta_q^{(p)} x_q^{(p)} - y_q^{(p)}) x_q^{(p)}$$

$$\frac{\partial J(\theta)}{\partial \theta_{pq}} = 2x_q^{(p)} (\theta_q^{(p)} x_q^{(p)} - y_q^{(p)})$$

6(b)

$$\frac{\partial J(\theta)}{\partial \theta_{pq}} = \begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_{11}} & \frac{\partial J(\theta)}{\partial \theta_{12}} & ...... & \frac{\partial J(\theta)}{\partial \theta_{1k}} \\ \frac{\partial J(\theta)}{\partial \theta_{21}} & \frac{\partial J(\theta)}{\partial \theta_{22}} & ...... & \frac{\partial J(\theta)}{\partial \theta_{2k}} \\ \vdots & \vdots & ...... & \vdots \\ \vdots & \vdots & ...... & \vdots \\ \frac{\partial J(\theta)}{\partial \theta_{n1}} & \frac{\partial J(\theta)}{\partial \theta_{n2}} & ...... & \frac{\partial J(\theta)}{\partial \theta_{nk}} \end{bmatrix}$$

$$= \begin{bmatrix} 2x_1(\theta^T x - y)_1 & 2x_1(\theta^T x - y)_2 & ...... & 2x_1(\theta^T x - y)_k \\ 2x_2(\theta^T x - y)_1 & 2x_2(\theta^T x - y)_2 & ...... & 2x_2(\theta^T x - y)_k \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ 2x_n(\theta^T x - y)_1 & 2x_n(\theta^T x - y)_2 & ...... & 2x_n(\theta^T x - y)_k \end{bmatrix}$$

$$= 2 \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} \begin{bmatrix} (\theta^T x - y)_1 & (\theta^T x - y)_2 & ...... & (\theta^T x - y)_k \end{bmatrix}$$

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

6(c)

Listing 3: cost function and ts vectorized gradient function

```
def f_6a(x, y, theta):
    x = vstack((np.ones((1, x.shape[1])), x))
    return sum((y - dot(transpose(x), theta.T)) ** 2)

# according to 6(a)
def df_6a(x, y, theta):
    x = vstack((np.ones((1, x.shape[1])), x))
    return 2 * dot(x, (dot(theta.T,x) - y.T).T)
```

6(d) adjusting h value by trying different numbers

Listing 4: computing several components of the gradient using finite-difference approximations

```
# 6d()
def grad_descent_6d(f, df, x, y, init_t, alpha, max_iter = 10000):
    EPS = 1e-5
    prev_t = init_t - 10 * EPS
    t = init_t.copy()

    max_iter = 100
    iter = 0
    while norm(t - prev_t) > EPS and iter < max_iter:
        prev_t = t.copy()
        t -= alpha*df_6a(x, y, t)
        if iter % 500 == 0:
            print "Iter: ", iter
        iter += 1
    return t


def part6():
    training_size = 50
    act_names = ['bracco', 'gilpin', 'harmon', 'baldwin', 'hader', 'carell']
    training_set, validation_set, test_set = part2_lst(act_names, training_size)

    training_stack = np.empty((0, 1024))
    for i in range(6):
        act_train = get_imgs(training_set[act_names[i]], training_size)
        training_stack = np.vstack((training_stack, act_train))

    m_training = np.zeros((training_size * 6, 6))
    training_counter, training_idx, = 0, 0
    for i in range(training_size * 6):
        training_cp = 0
        for j in range(6):
            if j == 6:
                loop_index = 0
            else :
                loop_index = j
            if loop_index == training_idx and training_cp == 0:
                m_training[i][j], training_cp = 1, 1
                training_counter += 1
                if training_counter == training_size:
                    training_idx += 1
                    training_counter = 0
    training_ylabel = m_training

    theta = np.zeros((1025, 6))
    training_se, label_se = training_stack[1], training_ylabel[1]
    t = grad_descent_6d(f, df_6a, training_stack.T, training_ylabel, theta, 10E-6)

    # along 5 coordinates
    for i in range(5):
        h_value = 1e-7
        theta = t.copy()
        mtx_l, mtx_r = np.random.randint(0, 1025), np.random.randint(0, 5)
        theta[mtx_l, mtx_r] = t[mtx_l, mtx_r] + h_value
        x = training_se.T.reshape(1024, 1)
        y = label_se.reshape(1,6)
        print "=== [", mtx_l, ",", mtx_r, "] ==="
        print "finite difference:", (f_6a(x, y, (theta).T) - f_6a(x, y, t.T))/float(h_value)
        print "gradient output:", df_6a(x, y, t)[i, j]
```

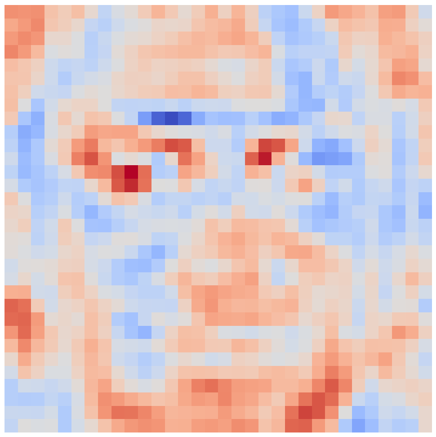
```

=== [ 659 , 0 ] ===
finite difference: -1.01145457343
gradient output: 0.462168983767
=== [ 1019 , 1 ] ===
finite difference: 0.188179660743
gradient output: 0.418670726471
=== [ 827 , 0 ] ===
finite difference: -0.923021390431
gradient output: 0.371547614401
=== [ 1020 , 0 ] ===
finite difference: -0.707465506178
gradient output: 0.400546452598
=== [ 110 , 3 ] ===
finite difference: 0.288692850736
gradient output: 0.382422178725
```

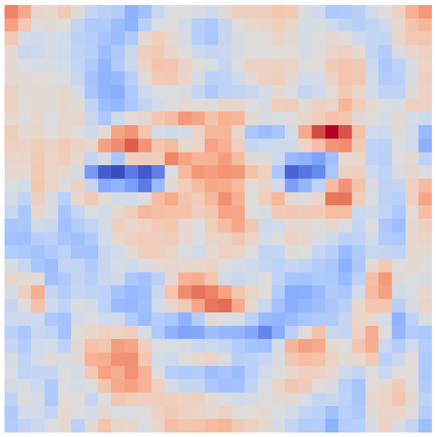
Part 7

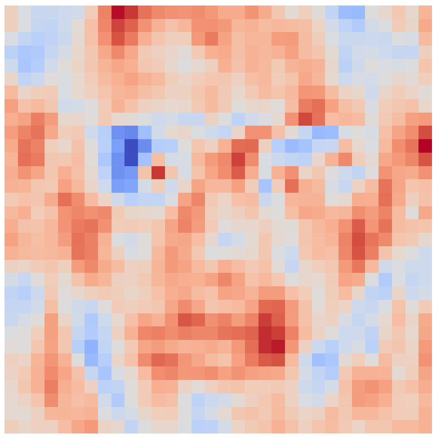
Performance on training: 92.619047619
Performance on validation: 75.0
Alpha: 10E-7

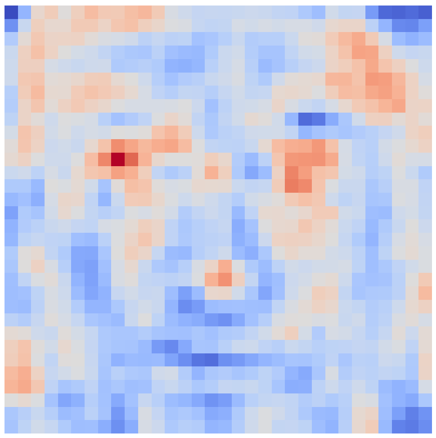
I choose use the alpha a bit smaller than the one (10E-6) I used earlier.
10E-6 is too large and the system wont work.
10E-8 is giving performance not as good as 10E-7.

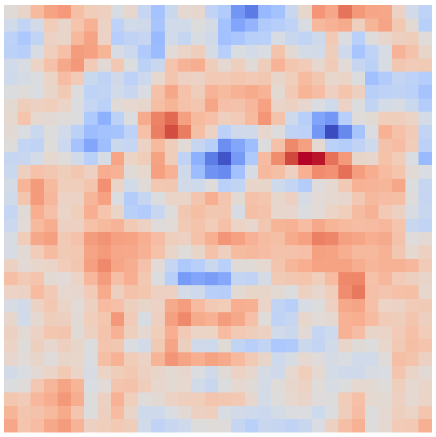
Part 8

Lorraine Bracco ⇒ 

Peri Gilpin ⇒ 

Angie Harmon ⇒ 

Alec Baldwin ⇒ 

Bill Hader ⇒ 

Steve Carell ⇒