CSC411: Assignment 1

Due on Monday, January 29, 2018

Cindy Lian

Part 1

Description of Dataset

The dataset of faces is given as a text file with each line in the file representing one photo. Each line includes the actor's or actress' name, the URL where the photo can be downloaded from, and the bounding box of the part of the photo which should be cropped out. In the given dataset, there are 6 actors and 6 actresses - each of which has 100+ photos. The raw photos are given with sizes ranging from 3000x4500 pixels to 200x200 pixels. Some of the images given in the dataset do not work and they must be excluded either manually or through code.

Some of the photos have the actor or actress facing the camera head on, some of them have the actor or actress looking to the side, and some of the photos contain other people in addition to the intended actor. Most of the crop boundaries are good at centering the face without having additional space on the sides of the cropped image. The images in Figure 1 have been selected to properly demonstrate the quality of the crop boundaries and of the photos themselves.

(a) Alec Baldwin (b) Peri Gilpin (c) Angie Harmon

Figure 1: Three images selected to demonstrate the quality of the crop boundaries

As seen in Figure 1(a), the image and crop of Alec Baldwin is not accurate because he is not facing forward and there is a lot of background on the right side of the crop of his face. The image of Peri Gilpin in Figure 1(b) is accurate because her face is centered and facing forward. When there are a lot of images like this one, it makes it easy for the cropped-out faces to be aligned with each other. The cropped out image of Angie Harmon in Figure 1(c) is in between the previous two. Overall, the majority of the images have accurate bounding boxes, similar to the one in Figure 1(c) and the faces can be aligned with each other decently well. There are a few images with bad bounding boxes and there are a few images with no face at all.

Part 2

Separating the Dataset

The dataset was separated into the training set, validation set, and test set simply by iterating through each of the actors/actresses and allocating the images in the order that they were downloaded. The first 65 images for each actor were allocated to the training set, the next 10 were allocated to the test set, and the final 10 images were allocated to the validation set.

Listing 1: Code used to split the images into the folders

```
#sort it in the corresponding folder
if (count < 65):
    location = "Training/"
elif (count <75):
    location = "Test/"
else:
    location = "Validation/"
```

Part 3

Distinguishing Alec Baldwin from Steve Carell

Linear regression was used to build a classifier which distinguishes images of Alec Baldwin from Steve Carell. The following cost function was minimized using gradient descent:

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

For the training, Steve Carell was given the label 0 and Alec Baldwin was given the label 1. When classifying the images in the validation set, 0.5 was used as a boundary for the labels. An image with a hypothesis value less than 0.5 was classified as Steve Carell and an image with a hypothesis value greater than 0.5 was classified as Alec Baldwin.

The parameters used for gradient descent were:

- $\alpha = 0.000001$
- max_iter $= 50000$
- $\theta_0 = [0.01] * 1025$

These parameters were chosen after testing gradient descent. It was found that when alpha was too large (¿0.1), the code would cause an overflow. When alpha was too small, gradient descent would take too long to converge and after 50000 iterations, it still did not get the cost function close to its minimum. Thus, 0.000001 was chosen for alpha. The max iterations 50000 was chosen because when observing the cost function at the end of the 30000 iterations given in the gradient descent code on the CSC411 course website, it was still decreasing by a significant amount per iteration when it was terminated. The initialization of theta was chosen by trial and error. Randomizing the initial thetas and initializing thetas to all ones did not result in as good of a performance as the initialization to a value close to zero (0.01).

The cost of the training set is: 3.6987
The cost of the validation set is: 1.3314
The performance of the classifier on the training set is: 100%
The performance of the classifier on the validation set is: 90%

The following listing contains the code which was used to compute the output of the classifier for either the training set or the validation set.

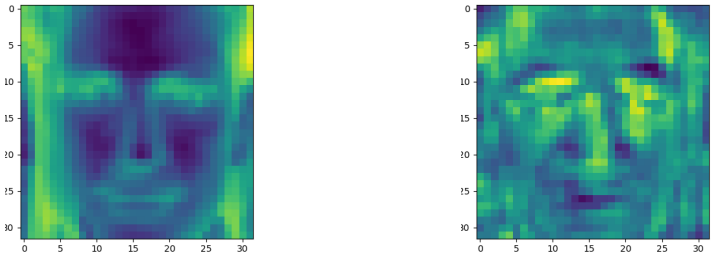
Listing 2: Code used to compute the output of the classifier

```
def test(theta, folder):
    '''Tests the trained theta against a set of images. Returns the performance of the
    classifier against the chosen set of images.

    Arguments:
    theta --- The theta trained used gradient descent
    folder --- The set of images which should be used (ie "Training" or "Validation")

    Return:
    performance --- Ratio of correctly classified images to total images
    '''
    correct = 0 #counter storing the number of images classfied correctly
    (images,y) = readImages (folder,100)
    images = vstack( (ones((1, images.shape[1])), images))

    for i in range (0,len(y)):
        if (y[i] == 0):
            print ("The image is of Steve Carell. Classified as: ")
        else:
            print ("The image is of Alec Baldwin. Classified as: ")

        classification = dot(theta, images.T[i])

        #0.5 is used as the boundary for the labels. If the correct label is 0
        #and the image was classified as <0.5, then it was classfied correctly.
        #If the correct label is 1 and the image was classified as >0.5, then it
        #was classified correctly.
        if (classification < 0.5):
            if (y[i] == 0):
                correct = correct + 1
            print ("Carell " + str(classification) + "\n")
        else:
            if (y[i] == 1):
                correct = correct + 1
            print ("Baldwin " + str(classification) + "\n")

    performance = float(float(correct)/len(y))*100
    print ("The classifier is correct " + str(performance) + "% of the time.")
    return performance
```

Part 4

Visualizing Theta

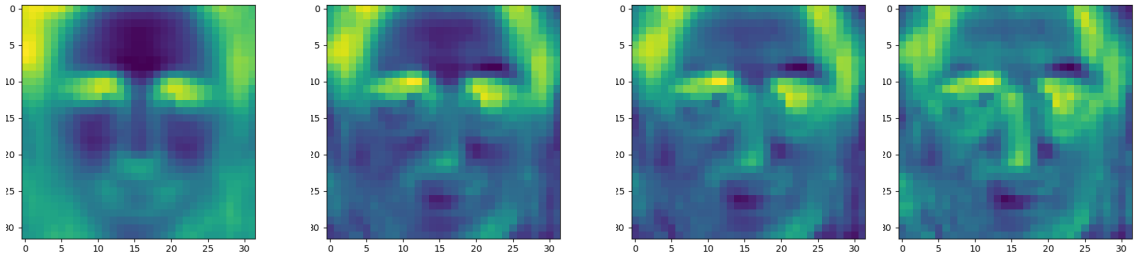
4(a) The following figures are visualizations of theta when trained on two images per actors versus a full training set.



(a) Two images (b) Full training set

Figure 2: Theta represented as an image with different training set sizes

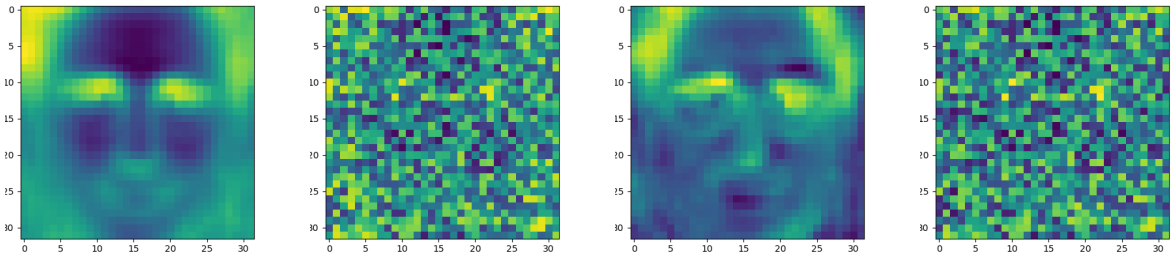
4(b) The following figures are visualizations of theta after having been trained using gradient descent under different parameters, but all trained using the full training set. In Figures 3(a) to 3(d), gradient descent was terminated at different steps. Theta was initialized to 0.01 for all images in Figure 3.



(a) 100 iterations (b) 500 iterations (c) 1000 iterations (d) 5000 iterations

Figure 3: Theta represented as an image with different stopping points for gradient descent

In Figures 4(a) to 4(d), theta was initialized using different strategies. In 4(a) and 4(c), theta was initialized to 0.01. In 4(b) and 4(d), theta was initialized to random values in between 0 and 1. Both 100 iterations and 1000 iterations of gradient descent are shown. After 1000 iterations of gradient descent when theta has been initialized randomly, there is not any kind of pattern in the image, whereas all the images with a non-random initialization have a discernible shape of face.



(a) $\theta_0 = 0.01$, itr=100 (b) $\theta_0 = rnd$, itr=100 (c) $\theta_0 = 0.01$, itr=1000 (d) $\theta_0 = rnd$, itr=1000

Figure 4: Theta represented as an image with different initializations of theta

Part 5

Demonstrating Overfitting

For this problem, theta was trained on actors in "act" to classify images as containing someone who is male or female. Gradient descent was run with the constant parameters of $\alpha = 0.000001$, max_iter $= 30000$, and $\theta_0 = [0.01] * 1025$. Various training set sizes were used from 5 images per actor to 65 images per actor, incrementing by 5 images. Figure 5 contains a plot with the performances of the classifiers at training set size.

The green line represents the performance of the classifier on the training set, and the blue and orange lines represent the performance of the classifier on the validation sets of act and act2 respectively. The performance of the classifier on the training set is almost perfect, with the exception of the smallest training set size of 5 images per actor which is slightly lower.The performance of the classifier on validation sets of act and act2 both increase as the size of the training set increases, with the performance on act2 being higher overall than the performance on act.

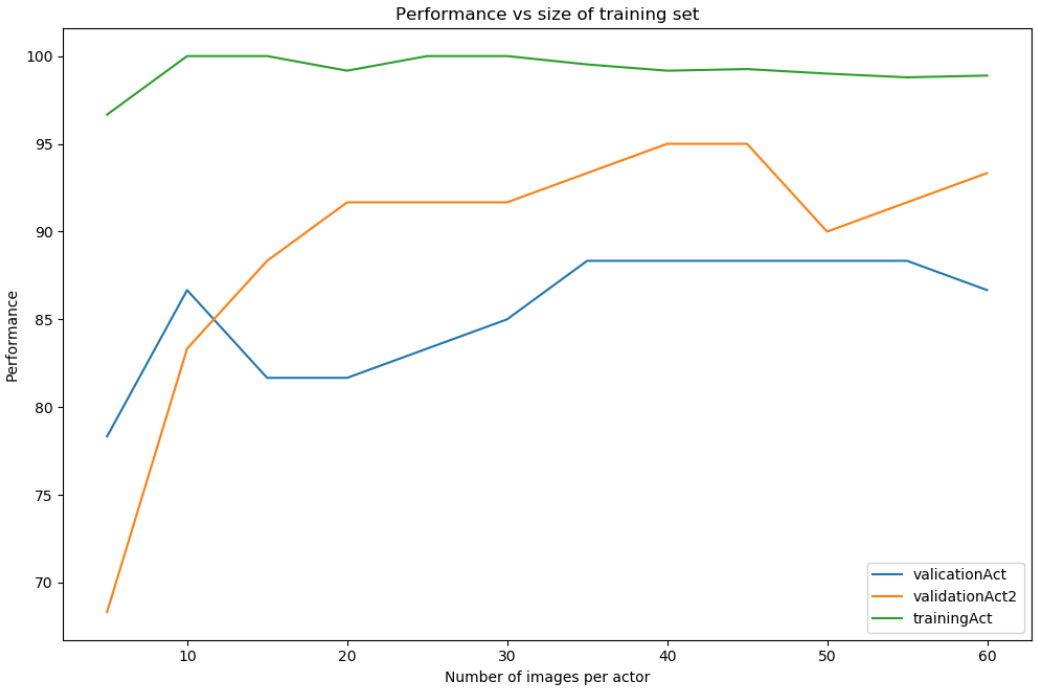


Figure 5: Plot of performances at different training set sizes per actor

Part 6

Classification with Multiple Classes

6(a)

The cost function is:

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

Find:

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

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

$$= 2 \sum_i (\theta^T x^{(i)} - y^{(i)})_q x_p^{(i)}$$

6(b)

Let m be the number of training examples.
Let θ be a p by q matrix, x be an p by 1 matrix, and y be a q by 1 matrix.

$$\frac{\partial J}{\partial \theta} = \begin{bmatrix} \frac{\partial J}{\partial \theta_{11}} & \cdots & \frac{\partial J}{\partial \theta_{1q}} \\ \vdots & \ddots & \vdots \\ \frac{\partial J}{\partial \theta_{1p}} & \cdots & \frac{\partial J}{\partial \theta_{pq}} \end{bmatrix} = \begin{bmatrix} 2\sum_i^m (\theta^T x^{(i)} - y^{(i)})_1 x_1^{(i)} & \cdots & 2\sum_i^m (\theta^T x^{(i)} - y^{(i)})_q x_1^{(i)} \\ \vdots & \ddots & \vdots \\ 2\sum_i^m (\theta^T x^{(i)} - y^{(i)})_1 x_p^{(i)} & \cdots & 2\sum_i^m (\theta^T x^{(i)} - y^{(i)})_q x_p^{(i)} \end{bmatrix}$$

$$= 2 \begin{bmatrix} x_1^{(1)} & \cdots & x_1^{(q)} \\ \vdots & \ddots & \vdots \\ x_p^{(1)} & \cdots & x_p^{(q)} \end{bmatrix} \left\{ \begin{bmatrix} x_1^{(1)} & \cdots & x_p^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(q)} & \cdots & x_p^{(q)} \end{bmatrix} \begin{bmatrix} \theta_1^T & \cdots & \theta_q^T \\ \vdots & \ddots & \vdots \\ \theta_1^T & \cdots & \theta_q^T \end{bmatrix} - \begin{bmatrix} y_1^{(1)} & \cdots & y_q^{(1)} \\ \vdots & \ddots & \vdots \\ y_1^{(m)} & \cdots & y_q^{(m)} \end{bmatrix} \right\}$$

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

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

6(c) The following is the code of the cost function and its vectorized gradient function.

Listing 3: Code of the cost function and gradient function

```
def fPart6 (X, Y, theta):
    '''Vectorized cost function'''
    return sum( (Y - matmul(theta.T,X)) ** 2)

def dfPart6 (X, Y, theta):
    '''Vectorized cost function gradient'''
    return 2*matmul(X,(matmul(theta.T, X)-Y).T)
```

6(d) The following listing includes the code to calculate a finite difference approximation. The value of h was selected to be 0.01, as it produced the lowest error between the finite difference and the gradient. The two values were compared simply by indexing the gradient with the coordinate that was selected to calculate the finite difference, taking the value calculated through finite differences, taking the difference of the two, and dividing by the indexed gradient.

Listing 4: Code to calculate a finite difference at coordinate pq

```
def df_approxPt6d (X,Y,p,q,theta, h):
    cost = fPart6(X,Y,theta)
    theta[p][q] += h

    return (fPart6(X,Y,theta)-cost)/h

def part6d (p,q):
    alpha = 0.000001
    max_iter = 100000
    theta0 = array([0.01]*1025*6).reshape([1025,6]) #initialized to all low values

    (X,Y) = readImages7("Training")
    X = vstack( (ones((1, X.shape[1])), X))

    theta = dfPart6(X,Y,theta0)
    theta_approx = df_approxPt6d(X,Y,p,q,theta0,0.01)
    diff = (theta[p][q]-theta_approx)/theta[p][q]
    print("Approx: " + str(theta_approx))
    print("Exact: "+ str(theta[p][q]))
    print ("% Difference = " + str(diff*100))
```

Below is a summary of the coordinates that were tested to compare the finite differences and the gradient. It can be seen that the percent difference between the finite differences and gradients are all less than 1.

Coordinates	Finite Difference	Gradient	% Difference
(5,2)	1183.446	1182.993	0.038%
(234,4)	3007.661	3005.666	0.066%
(600,5)	3415.365	3412.867	0.073%
(750,3)	2808.514	2806.759	0.062%
(1000,1)	1979.3714659	1978.33044306	0.05%

Part 7

Six actor gradient descent

The this problem, gradient descent trained on the images of the 6 actors in 'act'. The labels of each of the six actors are as follows:

```
[1,0,0,0,0,0] = 'bracco'
[0,1,0,0,0,0] = 'gilpin'
[0,0,1,0,0,0] = 'harmon'
[0,0,0,1,0,0] = 'baldwin'
[0,0,0,0,1,0] = 'hader'
[0,0,0,0,0,1] = 'carell'
```

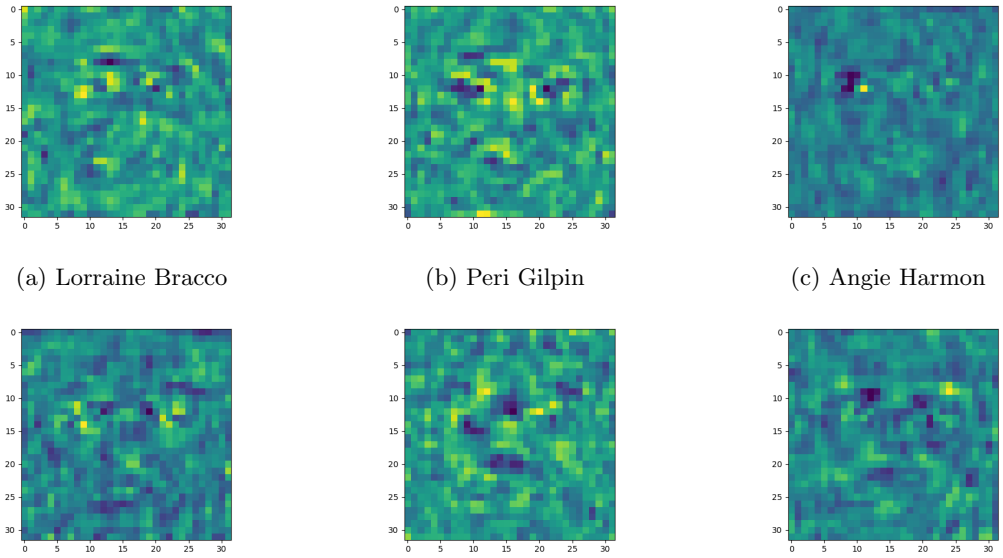
The parameters for this multiclass classification were chosen similarly to as in Part 3 of the assignment. α was chosen to be small enough so as to not overflow the program during multiplication, but not small enough to not converge the function to a minimum quick enough. In then end, the same $\alpha = 0.000001$ was chosen. Theta was initialized in the same strategy as previously, as it was shown that random initializations of theta converge very slowly. The maximum number of iterations was chosen to be 100000 so that gradient descent had enough time to find the true minimum of the cost function.

After training theta with these parameters, the accuracy of the classifier on the training set was found to be 99.7% and the accuracy of the classifier on the validation set was 80.0%.

Part 8

Visualizating theta for multiple classes

In Figure 6, the theta for each of the actors is represented as a 32x32 pixel image.



(a) Lorraine Bracco (b) Peri Gilpin (c) Angie Harmon

(d) Alec Baldwin (e) Bill Hader (f) Steve Carell

Figure 6: 6 images each representing the theta of their respective actors