CSC411: Project #1

Due on Monday, January 29, 2018

Xiangyu Kong

Problem 1

Dataset description

The dataset consists of 1691 gray-scale 32 × 32-pixel of images of 12 different actor/actress's faces. The angle, lighting and expressions for the faces are all different (as shown in Figure 1a and Figure 1b). Some faces aren't fully shown (as shown in Figure 1c). After cropping their faces out, examining the dataset, most of the faces are correctly cropped given the coordinates (Figure 1d), but there are still incorrect coordinates that leads to incorrect faces (Figure 1e). Also, some cropped faces are not aligned together. e.g. Some are center aligned and some are not (Figure 1d and Figure 1f).

(a) front

(b) side

(c) blocked face

(d) correctly cropped

(e) incorrect coordinates for face

(f) face not aligned to the center

Figure 1

Problem 2

Separating the sets.
The algorithm for separating the sets of data is to first shuffle the data as j list, then through list slicing, first pick 10 images as the test set, then pick 10 images as the validation set, and finally the rest are used as the training set.

Problem 3

Linear classifier to distinguish between Alec Baldwin from Steve Carell
Using the divided data sets from part 2, the program trains the linear classifier by gradient descent and then apply the classifier to the validation set and the test set.

The functions to compute the output are as follows (all print statements are removed for simplicity):

Listing 1: Gradient Descent

```
def grad_descent(loss, dlossdx, x, y, init_theta, alpha):
    eps = 1e-5
    prev_theta = init_theta - 10 * eps
    theta = init_theta.copy()
    max_iter = 100000
    i = 0
    while norm(theta - prev_theta) > eps and i < max_iter:
        prev_theta = theta.copy()
        theta -= alpha * dlossdx(x, y, theta)
        i += 1
    return theta
```

Listing 2: Predict

```
def predict(im, theta):
    data = imread("./cropped/" + im) / 225.
    data = reshape(data, 1024)
    data = np.insert(data, 0, 1)
    prediction = np.dot(data, theta)
    return prediction
```

Listing 3: Test

```
theta = grad_descent(loss, dlossdx, x, y, theta, 0.005)
for im in validation_set:
    prediction = predict(im, theta)
    if im in actor1_validation_set and norm(prediction) > 0.5:
        correct_count += 1
    elif im in actor2_validation_set and norm(prediction) <= 0.5:
        correct_count += 1
```

In order for the system to work, I had to keep track of each vector/matrix's shape and when to use their transpose in order for the multiplication to work. Also, I had to tune the alpha value and the iteration value. This is because if the alpha value is too small, the gradient descent will be too slow so more iterations are required, and if the alpha value is too large, numpy will give *nan* on the trained gradients.

Problem 4

Plotting thetas
The produced thetas give figures Figure 2 for using a full dataset to train the theta and Figure 3 for using only two images to train the theta:

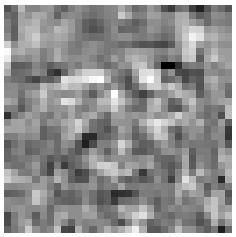


Figure 2: Full dataset

Figure 3: two data

Problem 5

Overfitting

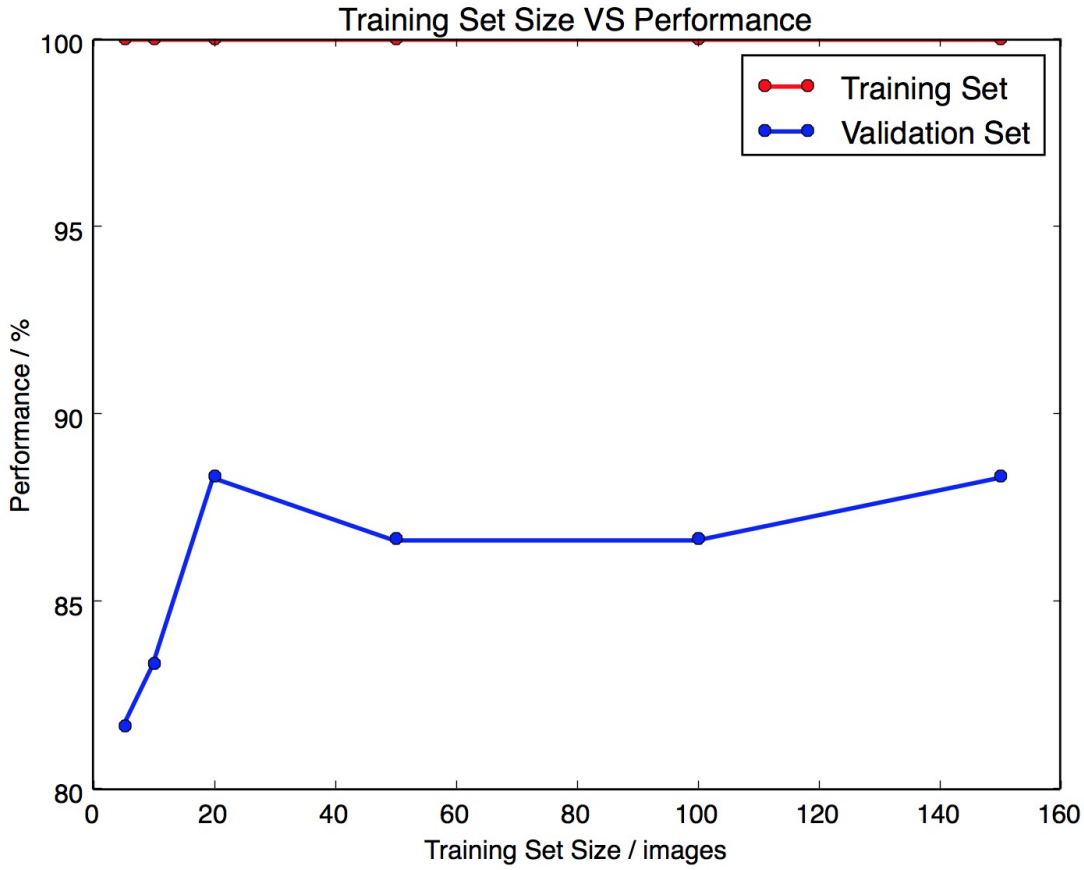


Figure 4: Overfitting

The graph above shows that after the training performance reaches the peak, which is around 88%, the performance for larger training set decreases and this is because at the peak, the set of data best describes the difference between j male's face and j female's face. After this size, the data starts focusing more on each individual's facial features instead of the features for male and female's faces.

Problem 6

Multiclass linear classification

1.

$$\theta^\top x^{(i)} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} & \dots & \theta_{n1} \\ \theta_{21} & \theta_{22} & \theta_{23} & \dots & \theta_{n2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \theta_{1k} & \theta_{2k} & \theta_{3k} & \dots & \theta_{nk} \end{bmatrix} \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{j=1}^{n} \theta_{j1} x_j^{(i)} \\ \sum_{j=1}^{n} \theta_{j2} x_j^{(i)} \\ \vdots \\ \sum_{j=1}^{n} \theta_{jk} x_j^{(i)} \end{bmatrix}$$

Then:

$$(\theta^\top x^{(i)} - y^{(i)})^2 = (\begin{bmatrix} \sum_{j=1}^{n} \theta_{j1} x_j^{(i)} \\ \sum_{j=1}^{n} \theta_{j2} x_j^{(i)} \\ \vdots \\ \sum_{j=1}^{n} \theta_{jk} x_j^{(i)} \end{bmatrix} - \begin{bmatrix} y_1^{(i)} \\ y_2^{(i)} \\ \vdots \\ y_k^{(i)} \end{bmatrix})^2$$

$$= \begin{bmatrix} (\sum_{j=1}^{n} \theta_{j1} x_j^{(i)} - y_1^{(i)})^2 \\ (\sum_{j=1}^{n} \theta_{j2} x_j^{(i)} - y_2^{(i)})^2 \\ \vdots \\ (\sum_{j=1}^{n} \theta_{jk} x_j^{(i)} - y_k^{(i)})^2 \end{bmatrix}$$

Then

$$\frac{\partial J}{\partial \theta_{pq}} = \sum_i 2(\sum_{j=1}^{n} \theta_{aq} x_a^{(i)} - y_q^{(i)}) x_p^{(i)}$$

2. Let m be the number of training examples.

X is j $n \times m$ matrix $\begin{bmatrix} x_1^{(1)} & \dots & x_1^{(m)} \\ x_2^{(2)} & \dots & x_2^{(m)} \\ \vdots & \ddots & \vdots \\ x_n^{(1)} & \dots & x_n^{(m)} \end{bmatrix}$

Y is j $k \times m$ matrix $\begin{bmatrix} y_1^{(1)} & \dots & y_1^{(m)} \\ y_2^{(2)} & \dots & y_2^{(m)} \\ \vdots & \ddots & \vdots \\ y_n^{(1)} & \dots & y_n^{(m)} \end{bmatrix}$

Then $\nabla J(\theta) = 2X(\theta^\top X - Y)^\top$.

Then for each row q for $(\theta^\top X - Y) = \begin{bmatrix} \sum_{j=1}^{n} \theta_{aq} x_a^{(1)} - y_q^{(1)} & \sum_{j=1}^{n} \theta_{aq} x_a^{(2)} - y_q^{(2)} & \dots & \sum_{j=1}^{n} \theta_{aq} x_a^{(i)} - y_q^{(m)} \end{bmatrix}$.

Each row p for $X = \begin{bmatrix} X_p^{(1)} & X_p^{(2)} & \dots & X_p^{(m)} \end{bmatrix}$

Then the pqth entry for $2X(\theta^\top X - Y)^\top$ is $2 \sum_{i=1}^{m} (\sum_{j=1}^{n} \theta_{jq} X_j^{(i)} - Y_q^{(i)}) X_p^{(i)} = \frac{\partial J}{\partial \theta_{pq}}$

3. Code for cost function and its gradient function in 6.1:

Listing 4: Test

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

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

Problem 7

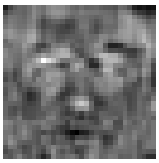
Running multiclass linear classification The performance on Training set is $570/621 = 91.7\%$ and on Validation set is $50/60 = 83.3\%$

α was chosen to be 0.0000001, which is much smaller than when we trained binary linear classification. This is because when larger alpha is used, the trained theta will contain *nan*.

The output is obtained by choosing the argmax of the prediction array. This is because the higher the predicted value is, the more confident the algorithm is. Then we compare the index of the chosen label with the correct label, and finally we can see if the prediction is correct or not.

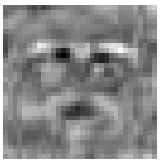
Problem 8

Plotting thetas
The trained thetas are as follows:



(a) baldwin

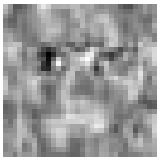
(b) bracco



(c) carell

(d) gilpin

(e) hader



(f) harmon

Figure 5