Assignment 4 of Math6321

Due: Feb 25, 2015, in class

From this assignment, you will use the logistic regression model to perform face detection in images. We will use a subset of CBCL Face Database from MIT. The set contains 2429 faces and 4548 non-faces. Use the first 400 faces and 600 non-faces from the set to train the model, and use the rest of the images as the test set. You are welcome to use more data for the training. Each image is of size 19×19, and is reshaped as a 1×361 vector. There are two matricies in the Matlab data file, one for face, one for non-face. Each data is a n by 361 matrix, where n is the number of images in that set and each row represents one image.

Assume D is the data matrix, to view the ith image, you may do the following:

```
a=D(i,:);
a=reshape(a,19,19);
imshow(a)
```

For each training image x, we assign a label y, $y = 1$ if x is a face image and $y = -1$ if x is not a face image. The logistic regression model is

$$P(y = \pm 1|x, w) = \sigma(yw^T x) = \frac{1}{1 + exp(-yw^T x)}. \tag{1}$$

Assume $p(w)$ is Gaussian, with mean 0 and variance $\lambda^{-1} I$, where I is the identity matrix.

Given a data set $(\mathbf{X}, \mathbf{y})$, where $X = (x_1, x_2, ..., x_n)$ represent the n training images, $y = (y_1, y_2, ..., y_n)$ represent the labels of the training images, we would like to find a parameter vector w which maximizes the likelihood $P(\mathbf{X}|\mathbf{y}, \mathbf{w})$. It is equivalently to maximizes

$$l(w) = \sum_{i=1}^{n} \ln(1 + \exp(-y_i w^T x_i)) + \frac{\lambda}{2} w^T w. \tag{2}$$

You may use $\lambda = 0.01$. You are welcome to try other values of λ.

1. Find the gradient of l with respect to w. Write a Matlab function "myfunc.m" which calculates both $l(w)$ and its gradient. In addition to w, you may also need put x and y as the variables of the function. The function will look like $[f, g] = myfunc(w, x, y)$, where f returns the function value, g returns the gradient vector. There is a separate PDF document about how to pass extra variables in Matlab, you may find that useful.

2. Find the optimal w using the Matlab function "fminunc" starting with $w = \vec{0}$. Note that w is a vector of length 361, the same length as each image vector x. Plot the optimal w in a figure.

   To call the function "fminunc", you may do the following:

   options = optimoptions('fminunc','GradObj','on','MaxIter',1000,'Display','iter');

   w0=zeros(361,1);

   func=@(w)myfunc(w,x,y);

   w1=fminunc(func,w0,options);

   The default method is the trust-region Newton-CG method. Another option is the BFGS method. To use the BFGS method, you may define the options as follows:

   options=optimset('LargeScale','off', 'HessUpdate','bfgs','gradobj','on','MaxIter',1000,'display','iter');

   Try both methods.

3. Test your model on the test set and report the accuracy, i.e. the percentage of faces that are detected correctly and the percentage of non-faces that are detected correctly. For a given test image x, let $y = 1$, use Equation (1) to calculate the probability $p(y = 1|w, x)$ of that x is a face image. If $p > 0.5$, declare the image as a face, otherwise, declare it as a non-face.

4. Comment on the performance of the two algorithms and your results.