MIT CSAIL

6.869 Advances in Computer Vision

Spring 2011

Problem Set 11: Recognition and Boosting

Posted: Wednesday, April 27, 2011	**Due:** Wednesday, May 4, 2011

You should submit a hard copy of your work in class, and upload your code (and all files needed to run it, images, etc) to stellar.
Your report should include images and plots showing your results, as well as pieces of your code that you find relevant.

Problem 11.1 *GentleBoost*

In this problem you will derive the update steps for a variant of boosting called *GentleBoost* that we discussed in class. Assume that there are $x_1, \cdots, x_N$ training examples with corresponding labels $y_1, \cdots, y_N$. Note that $y_i = \{-1, +1\}$.

As in other boosting algorithms, a strong classifier is produced by iteratively adding weak classifiers. In this problem, you will derive one particular boosting algorithm, the GentleBoost algorithm.

The goal of the GentleBoost algorithm is to learn a classifier of the form

$$h[x] = sign(F[x]) = sign\left(\sum_{m=1}^{M} f_m(x)\right) \tag{1}$$

where the functions are weak-learners. In the above equation, M represents the number of weak learners in the classifier. The weak learners are chosen by minimizing the exponential loss, J, of the training set

$$J = \sum_{n=1}^{N} e^{-y_n F(x_n)} \tag{2}$$

(a) Explain why this is a reasonable criterion to optimize.

We will consider weak-learners of the form

$$f(x) = a(r_i(x) > \theta) \tag{3}$$

where $(r_i(x) > \theta)$ evaluates to either $+1$ or -1. These are sometimes known as *regression stumps*. The function $r_i(x)$ denotes the absolute value of the response of some filter to the observation x. The variable a can be thought of as a weight, given this particular weak learner.

(b) Given a set of responses over the training set, $r_i(x_1), \cdots, r_i(x_N)$, the GentleBoost algorithm chooses the coefficient a by minimizing a quadratic approximation to the exponential loss. Assume that we just completed the $k-1$ iteration, and so we have picked up $k-1$ weak learners. Consider the k iteration. The function $F[x]$ denotes the response of the previous weak learners chosen

$$F[x] = \sum_{m=1}^{k-1} f_m(x) \tag{4}$$

Let $J(a) = \sum_{n=1}^{M} e^{-y_n(F[x_n]+a(r_i(x_n)>\theta))}$. Write the second-order Taylor approximation of $J(a)$, expanded about $a = 0$. Assume that $r_i(\cdot)$ and θ have already been chosen and that you are simply choosing a.

(c) Derive the expression for the value of a that minimizes this approximation to the exponential loss.

Problem 11.2 *Eigenfaces for recognition*

In this problem we explore the use of eigenfaces for person recognition. Download the MATLAB data file `faces.mat`, which contains the cropped, normalized 51×43 images of 34 individuals. Each individual has been photographed with two facial expressions: "neutral" and "smiling". The pixels of each person's image have been stored as columns of the `neutralFaces` and `smileFaces` matrices.

(a) To (approximately) account for illumination variations, normalize each of the face vectors so that it has zero mean and unit variance.

(b) Assume that we have a database of neutral face images, and want to determine the identity of the smiling individuals. For each of the 34 smiling faces, measure the norm of the difference between their normalized appearance vector and each of the 34 neutral faces. Classify each smiling face according to its nearest neighbor. What percentage of the smiling faces are correctly recognized?

(c) Using the MATLAB `svd` command, determine the principle components of the set of normalized neutral faces (do not include the smiling faces). Be sure to subtract the mean face before performing your PCA. Plot the mean face and the first three principle components (the "eigenfaces").

Hint: To avoid excessive memory usage when calculating SVD, use MATLAB's economy size option: `svd(mat, 0)`.

(d) Determine the number of principle components required to model 90% of the total variance in the neutral face set. Project each of the neutral and smiling faces onto the corresponding eigenfaces. Use the coefficients of these projections to classify each smiling face. Compute the percentage of correctly recognized faces, and compare to part (b).

(e) Repeat part (d) using the number of principal components required to model 80%, 70%, 60%, and 50% of the total neutral face variance. Plot the corresponding recognition rates.