Pattern Recognition with Applications to Biomedical Images
Independent Study in Mathematics – CSUN Spring 2006
Module 5 – The Fisher approach P. Perona & K. F. Stevenson

January 22, 2007

1. Understanding Fisher:

   (a) Understand the notes on Fisher. Realize that the generalized eigenvalue problem may be solved using the pseudoinverse. Numerically this is a bad idea because it may amplify noise (if B has really small nonzero eigenvalues). Test the Matlab function 'eigs' and realize that it does solve the generalize eigenvalue problem.

   (b) Implement Fisher in Matlab in the special case when each category has the same number N of training examples, and the training examples are sorted by category. This will make the code somewhat simpler to realize. OK if the code is not efficient.

   (c) Compare own code with Markus Weber's code I will send you on a simple synthetic example.

2. Playing with Fisher (You can skip b and c):

   (a) Use Weber's code to calculate the first 2 Fisher dimension for the 4 categories used so far. Do so using the training set and a great number of features (those used in the shotgun approach). Visualize the training data plotted w.r. to the first 2 Fisher dimensions.

   (b) pick a classifier by hand. Measure its performance and compare to performance with (i) hand-chosen features, (ii) shotgun approach. Which one worked best?

   (c) Adjust parameters so that (i) the number of training examples for the Fisher dimensions is smaller than the number of features used, (ii) the number of training examples is larger. Compare the training and test error in both cases. Discuss.

   (d) Make a classifier out of Fisher for the four classes of data:

i. First take the classes of data in pairs (C_i, C_j) for $i < j$ do Fisher on *all* the features (that you want) to obtain the first Fisher dimension, $[a_{ij}]$ (this is your matrix that projects onto the Fisher line).

ii. Now choose a decision boundary b_{ij} (i.e. a point on your Fisher line) for (C_i, C_j) so that $[a_{ij}] - b_{ij}$ is positive on class C_i and negative on class C_j. Let N_{ij} be the number of points in the union of the test images in classes C_i and C_j. Then using $[a_{ij}]$ on the images J_k in these classes, we obtain N_{ij} points $a_1 = [a_{ij}]J_1, ..., a_{N_{ij}} = [a_{ij}]J_{N_{ij}}$. Now choose $N_{ij} - 1$ points $b_1, ..., b_{N_{ij}-1}$ on the Fisher line so that $a_l < b_l < a_{l+1}$ (it does not matter which b_l you choose, so you could take the midpoint if you want). Now for each b_l let E_{il} be the number of C_i training images that would be miss-classified as C_j images if you took b_l to be your boundary. Define E_{jl} similarly and take $E_l = E_{il} + E_{jl}$. Take b_{ij} to be the b_l with the minimum number of errors.

iii. Now you need to put these boundaries together to make a classifier. You can do this in two ways. Try both and compare results

A. Build a tree: Given an image I. Compute $\delta([a_{12}]I - b_{12})$ (zero if negative, 1 if positive). If 0 then repeat with $\delta([a_{13}]I - b_{13})$. If 1 then repeat with $\delta([a_{23}]I - b_{23})$. Loop some more to get a decision tree.

B. Another way to do this is to compute for each image J_k in training set the $\frac{n(n-1)}{2}$-vector $\vec{v}(J_k)$ (here $n = 4$) with entries $\delta([a_{ij}]J_k - b_{ij})$ for $i < j$. Now consider the set V of all $\frac{n(n-1)}{2}$ vectors $\vec{v}$ with 0 or 1 entries. As each image in the test set will give us one such matrix, we need to decide a class assignment for for each $\vec{v} \in V$. Can you use the data from your training set (i.e. the vectors $\vec{v}(J_k)$) to figure out this assignment?

(e) Now with your Fisher classifier, re-run it on the training and test data provided in 4-Categories-Large.zip.

(f) Finally re-run you Classifier (the one where you drew the lines by hand) on the training and test data provided in 4-Categories-Large.zip. Compare this with the Fisher on the same set.