ASSIGNMENT ON FLD

DUE BY MIDNIGHT DECEMBER 4, 2017

Implement and experiment with FLD

The purpose of this exercise is to implement the Fisher Linear Discriminant (FLD) and obtain a deeper understanding of this simple classification tool. You will also practice how to draw an ROC curve from empirically obtained data and how to compute a scalar performance characteristic, the total error probability under equal priors, P_{E}. The FLD is covered in the pdf slides 'FLD.pdf' available from Blackboard.

Task I

Write a Matlab routine for the FLD analysis called 'FLD.m' that has the following on the input and output:

Input: Two $d \times N$ arrays $\mathbf{X}$ and $\mathbf{Y}$ containing d-dimensional cover and stego features as columns. To be more precise, the first column of $\mathbf{X}$ is the feature of the first cover image, and the first column of the array $\mathbf{Y}$ is the feature of the first stego image, etc.

Output: The generalized eigenvector $\mathbf{v} \in \mathbb{R}^d$, a plot of the ROC curve (the probability of detection, P_{D}, as a function of the probability of false alarm, P_{FA}), and the minimal total error probability under equal priors:

$$P_{\mathrm{E}} = \min_{P_{\mathrm{FA}}} \frac{1}{2}(P_{\mathrm{FA}} + P_{\mathrm{MD}}) = \min_{P_{\mathrm{FA}}} \frac{1}{2}(P_{\mathrm{FA}} + 1 - P_{\mathrm{D}}).$$

Task II

Test your routine on artificial data by generating both input feature arrays as independent Gaussian vectors (in Matlab notation):

$$\begin{aligned} \mathbf{X} &= \mathrm{randn}(d, N); \\ \mathbf{Y} &= a + \mathrm{randn}(d, N); \end{aligned}$$

for $d = 10$, $N = 1000$, and $a = 0, 0.25, 0.5, 0.75, 1, 1.5$. For each a, you only need to generate one new array $\mathbf{Y}$ but keep the same $\mathbf{X}$ (it does not change with a).

Run your FLD routine for each value of a and store the error probability $P_{\mathrm{E}}(a)$. Plot $P_{\mathrm{E}}(a)$ as a function of a and include it in your homework document. You should see a decreasing function – as a increases, the separation between both classes increases and the detection error of FLD on your training data should decrease. Plot all six ROCs in one figure and include it in your homeowrk document.

What I need from you:

(1) A Matlab file FLD.m.

(2) A PDF (MS-Word) document with one figure showing $P_{\mathrm{E}}(a)$ and one figure showing six ROCs (one for each value of a). You can put all ROCs into a single figure – they should not overlap too much.

As a sanity check, the ROC for $a = 0$ should be pretty much following the diagonal, while for $a = 1.5$, it should be almost the ideal ROC bending close to the point $(P_{\mathrm{FA}}, P_{\mathrm{D}}) = (0, 1)$.

To make the implementation easier for you, I provide the following scheme for the FLD routine:

```
function [$\mathbf{v}$,$P_{\mathrm{E}}$] = FLD($\mathbf{X}, \mathbf{Y}$)
% $\mathbf{X}$ ... a matrix of cover features as columns
% $\mathbf{Y}$ ... a matrix stego features as columns
% $\mathbf{v}$ ... generalized eigenvector
% $P_{\mathrm{E}}$ ... total minimal detection error under equal priors
$d$ = number of rows of $\mathbf{X}$ ($\mathbf{Y}$) (feature dimensionality)
$N$ = number of columns in $\mathbf{X}$ ($\mathbf{Y}$) (the number of features)
$\mu_x$ = mean of each feature (row) of $\mathbf{X}$
$\mu_y$ = mean of each feature (row) of $\mathbf{Y}$
$\mathbf{M}_x$ = matrix $\mathbf{X}$ normalized so that each row of $\mathbf{X}$ has a zero mean
$\mathbf{M}_y$ = matrix $\mathbf{Y}$ normalized so that each row of $\mathbf{Y}$ has a zero mean
Compute the matrix $\mathbf{S}_w$
Compute the inverse of $\mathbf{S}_w$, $\mathbf{S}_w^{-1}$.
Compute $\mathbf{v} = \mathbf{S}_w^{-1} * (\mu_x - \mu_y)$, provided the means are column vectors.
```

Compute the projections $\mathbf{p}_x$ and $\mathbf{p}_y$ of the cover and stego features on $\mathbf{v}$, respectively (dot products of $\mathbf{v}$ and the corresponding feature matrix columns).

Adjust the projections so that you always have the cover class "on the left" and stego class "on the right":

```
    if mean($\mathbf{p}_x$) > mean($\mathbf{p}_y$), $\mathbf{p}_x = -\mathbf{p}_x$, $\mathbf{p}_y = -\mathbf{p}_y$;
```

Plot ROC. While this can done in many possible ways, I recommend the following approach.

- Merge $\mathbf{p}_x$ and $\mathbf{p}_y$ into one row vector or length $2N$, $\mathbf{p}_x$ first, $\mathbf{p}_y$ next; call it $\mathbf{p}$. Define a vector $\mathbf{I}$ of length $2N$, the first N elements are zeros, the last N are ones. Run [~,rank] = sort($\mathbf{p}$). The vector $\mathbf{I}$(rank) will contain the class labels (0 or 1) across all projections sorted by size from the smallest to the largest. Use the vector $\mathbf{I}$(rank) to compute the ROC. You can use a 'for cycle' here. Start with $P_{\mathrm{FA}}(1) = 1 = P_{\mathrm{D}}(1)$ (upper right corner of the ROC corresponding to a threshold at $-\infty$). Then, go through the for cycle from $i = 1$ to $2N$, and each time update both error probabilities. If $\mathbf{I}(\mathrm{rank}(i)) = 0$, it means you added a cover image left of your thresholds, which means, you lowered your P_{FA} by $1/N$ and P_{D} stays unchanged. On the contrary, if $\mathbf{I}(\mathrm{rank}(i)) = 1$, your FAs do not change but you just lowered P_{D} by $1/N$. Keep updating the values P_{FA} and P_{D} until you go through all $2N$ values. At the end, you should have $P_{\mathrm{FA}}(2N + 1) = P_{\mathrm{D}}(2N + 1) = 0$, which corresponds to the lower left corner of the ROC.

Compute P_{E} by evaluating the minimal value of $(P_{\mathrm{FA}}(i) + 1 - P_{\mathrm{D}}(i))/2$ over i.