ER 2023

1

a) ROC curves are related to the discriminative power of a feature by plotting sensitivity against specificity. A good feature separates classes well alone. We can calculate the area under the curve for each feature's ROC curve and select the best ones

b) That feature is a perfect classifier and there is a threshold that perfectly separates the classes.

c) Features 2 and 3. Since the AUCs are similar, we should select the least redundant feature (f_2 and f_3 are the least correlated)

2

a) Some examples:

→ Reduce dimensionality with LDA and then use MDC for classification.

→ Train binary classifiers and decompose the multiclass problem as one of the following problems:

→ One-Against-All :
train one classifier per class (that class against every other class). Ideally, only 1 $d_i(x)$ should be positive while all other $d_j(x)$, $j \neq 1$ should be negative, then choose class ω_i.

→ One-Against-One:
train $K(K-1)/2$ binary classifiers, each $d_{ij}(x)$ is trained to discriminate between ω_i and ω_j ($d_{ij}(x) > 0$ if x belongs to ω_i). The final result comes from majority voting.

→ Error-Correcting Output Codes (ECOC) is the general framework (OAA and OAO) are instances of it). Here we have a "coding matrix" that determines which classes each binary classifier discriminates, The final decision comes from comparing the vector of binary classifier outputs with the code words for each class (e.g. with Hamming distance metric) to find closest match.

(b) $D=3$ (3 rows in each mean vector)

(i) $K=3$ (3 mean vectors)

(ii) 1- Get projected mean for each class

$$\mu'_K = W^T u_K$$

$$\mu'_1 = \begin{pmatrix} 0,26 & -0,77 & -0,58 \\ -0,75 & 0,65 & 0,12 \end{pmatrix} \begin{pmatrix} +0,93 \\ -0,96 \\ -0,89 \end{pmatrix} \approx \begin{pmatrix} 1,01 \\ -0,03 \end{pmatrix}$$

$$\mu'_2 = W^T \begin{pmatrix} -0,26 \\ -0,19 \\ -0,27 \end{pmatrix} \approx \begin{pmatrix} 0,24 \\ 0,04 \end{pmatrix}$$

$$\mu'_3 \approx \begin{pmatrix} -1,24 \\ -0,01 \end{pmatrix}$$

2- Project sample

$$x' = W^T x \Leftrightarrow W \begin{pmatrix} 2 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} -2,76 \\ 0,16 \end{pmatrix}$$

3- Compute scores

$$d^2 = \left((x_1 - u_{K_1})^2 + (x_2 - u_{K_2})^2\right)^2$$

$$\omega_1 : \left((-2,76 - 1,01)^2 + (0,16 + 0,03)^2\right)^2 \approx 14,2$$

ω_2 : Do the same for the others and select

ω_3 : the best score

(iii) •

MDC decision boundary:

$$(m_1 - m_2)^T x - 0{,}5\left(\|m_1\|^2 - \|m_2\|^2\right) = 0$$

in LDA space:

$$x \to W^T x$$

$$m_k \to u'_k$$

$$(u'_1 - u'_2)^T (W^T x) - 0{,}5\left(\|u'_1\|^2 - \|u_2\|^2\right) = 0$$

$$(AB)^T = B^T A^T$$

$$\underbrace{(W(u'_1 - u'_2))^T}_{\text{weights vector}} x \underbrace{- 0{,}5\left(\|u'_1\|^2 - \|u'_2\|^2\right)}_{\text{bias}} = 0$$

$$w = W(u'_1 - u'_2) = \begin{pmatrix} 0{,}26 & -0{,}75 \\ -0{,}77 & 0{,}65 \\ -0{,}58 & 0{,}12 \end{pmatrix} \begin{pmatrix} 0{,}77 \\ -0{,}07 \end{pmatrix} = \begin{pmatrix} 0{,}25 \\ -0{,}64 \\ -0{,}46 \end{pmatrix}$$

$$b = -0{,}5\left(1{,}01^2 - 0{,}24^2\right) \approx -0{,}48$$

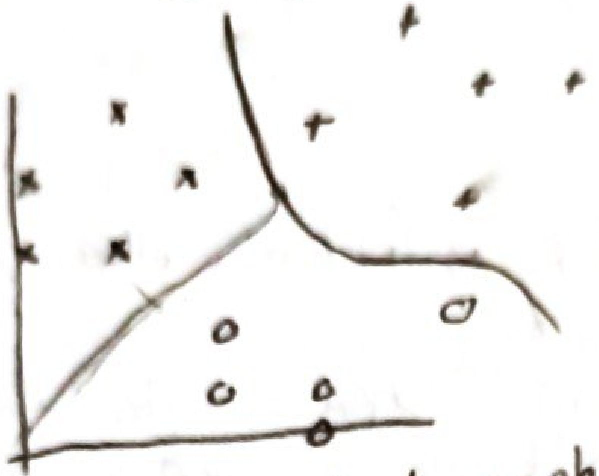
$$w^T x - b = 0{,}25 x_1 - 0{,}64 x_2 - 0{,}46 x_3 - 0{,}48$$

③

ⓐ It is lazy learner because there is no explicit training phase, we just need to store the labelled data

ⓑ The training algorithm does not exist

ⓒ



mais ou menos...

Note: in the actual graph from the exam, (4,4) belongs to w_1 (misclassification)

ⓓ (According to the actual exam's figure)

K=1 s_1 s_2 s_3 66% accuracy

predict: 3 2 3

✗ ✓ ✓

K=3 s_1 s_2 s_3 66% accuracy

1 1 3

✓ ✗ ✓

④

ⓐ Sep. margin γ is given by $\gamma = \frac{2}{||w||}$

$$\gamma = \frac{2}{||w_3||} = 2/\sqrt{1^2 + 1,3^2} \approx 1,22$$

ⓑ

One-vs-All:

3 classes

Classifier i is trained w/ i as $(+1)$ and others as (-1)

$$M = \begin{pmatrix} +1 & -1 & -1 \\ -1 & +1 & -1 \\ -1 & -1 & +1 \end{pmatrix}$$

For example, when using SVM 3, w_3 is $(+1)$ while w_1 and w_2 are (-1)

rows ➔ classes

cols ➔ classifiers

One-vs-One:

we need $\frac{K(K-1)}{2} = \frac{3(2)}{2} = 3$ classifiers ✓

Ignored classes are marked as 0

$$M = \begin{pmatrix} +1 & +1 & 0 \\ -1 & 0 & +1 \\ 0 & -1 & -1 \end{pmatrix}$$

(c) A sample is a support vector if its exactly on the margin boundary of the cannonical hyperplane for at least one classifier (if $|w^T x + b| = 1$

Test SVM1:

$$g_1(x) = w_1^T x + b =$$

$$= \begin{pmatrix} -3 \\ 1 \end{pmatrix} (3 \ \ 1) + 7 = -9 + 1 + 7 = -1$$

$|-1| = 1 \Rightarrow x$ is support vector for SVM1
(on the negative side)

x is a support vector. now let's check its class

$g_1(x) = -1 \rightarrow \text{Sign} = -1$

$g_2(x) = 0{,}4 \cdot 3 - 1{,}6 + 1{,}4 = 1 \rightarrow \text{Sign} = +1$

$g_3(x) = 3 + 1{,}3 - 8{,}3 = -4 \rightarrow \text{Sign} = -1$

So, we have $(-1 \ \ +1 \ \ -1)$, which is the same as row 2 from the OVA matrix
$\Rightarrow x$ belongs to ω_2

(Hamming distance = 0)

④

$g_1(x) = -3 \rightarrow \text{Sign} = -1$

$g_2(x) = -0,2 \rightarrow \text{Sign} = -1$

$g_3(x) = -1,2 \rightarrow \text{Sign} = -1$

Yes, this is a special case, the Hamming distance is the same between (-1 -1 -1) and all of the rows in the OVA matrix.

⇒ Ambiguous region

⑤

```
import numpy as np
def bayes_training(Xtr, Ttr)
    X_c1 = Xtr[:, Ttr == 1]   } Get samples from
    X_c2 = Xtr[:, Ttr == 2]   } each class

    total = Xtr.shape[1]                } Get priors:
    prior1 = X_c1.shape[1] / total      } P(1) and P(1)
    prior2 = X_c2.shape[1] / total      }
                                  cols
    mean1 = np.mean(X_c1, axis=1, keepdims=true) } Get class
    mean2 = np.mean(X_c2, axis=1, keepdims=true) } centroids

    cov1 = np.cov(X_c1, bias=True)  } Get cov
    cov2 = np.cov(X_c2, bias=True)  } matrices
                                      (bias=True normalizes by N)

    model = {
        "mu1": mean1,
        "mu2": mean2,
        "cov1": cov1,
        "cov2": cov2,
        "p1": prior1,
        "p2": prior2 } → return model
```

```
def gaussian_pdf(X, mu, cov):        multivariate
    D = X.shape[0]                       gaussian
    X_centered = X - mu
    det_cov = np.linalg.det(cov)
    inv_cov = np.linalg.inv(cov)
    ---
    const = 1/((2*np.pi)**(D/2)*(det_cov**0.5))
```

$$\# \frac{1}{2\pi^{(D/2)}\sqrt{det(cov)}}$$

```
    ---
    exp = -0.5 * np.sum(np.dot(X_centered.T, inv_cov) *
          X_centered.T, axis=1)
    return const * np.exp(exp)

def bayes_testing(Xte, Tte, model):
    likelihood1 = gaussian_pdf(Xte, model['mu1'], model['cov1'])
    likelihood2 = gaussian_pdf(Xte, model['mu2'], model['cov2'])
```

$P(x|w)$

```
    ---
    post1 = likelihood1 * model['p1']
    post2 = likelihood2 * model['p2']
```

Get posteriors ($\alpha\ p(x|w_i)\, p(w_i)$)

```
    ---
    out = np.where(post1 >= post2, 1, 2)
```

if post1 ≥ post2 assign 1, else 2

```
    return out
```

maybe also return performance metrics