2024 ER

① ⓐ Feature Selection: Choosing subset of the original features without transformations. (KW, ROC-AUC, ..)

Feature Reduction: Transforming original features into smaller set of new variables. Here, the physical meaning of the features is lost. (PCA, LDA, ..)

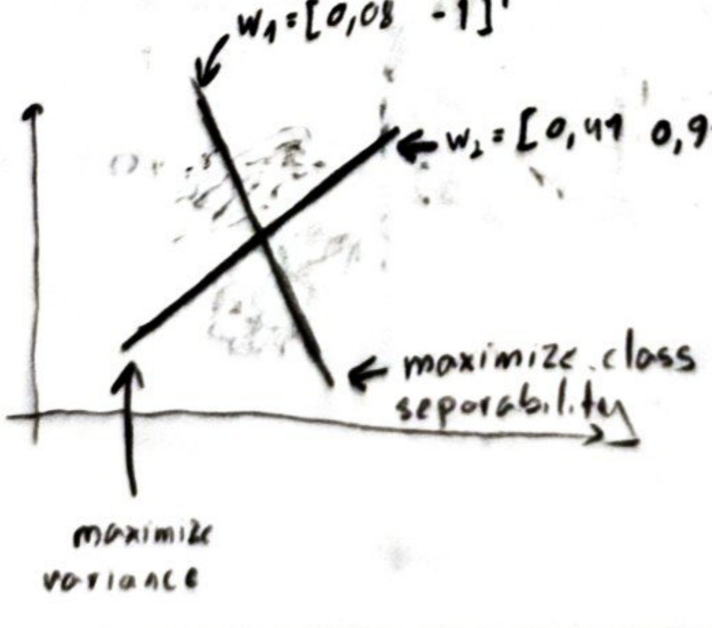
ⓑ Disadvantages:

- Reduces data to a maximum of min(C-1, D) where C is the number of classes and D is the number of original features.
- Assumes equal covariance matrices.
- Performs poorly if the class means are identical. J(w) becomes 0.
- Can't deal well w/ non-linearly separable data.

Advantages:

- Reduces dimensionality in the most optimal way to maximize the separation between classes

ⓒ



$w_1 \rightarrow$ LDA

$w_2 \rightarrow$ PCA

② ⓐ

Euclidean MDC: Ignores class covariance. Assumes that the covariance matrix is just a scalar of $\mathbb{1}$ (spherical data clouds)

Mahalanobis MDC: Accounts for the covariance matrices. Becomes a quadratic classifier if the classes have different cov matrices (but it is a linear classifier if they share a pooled cov matrix).

ⓑ $x = [1 \quad 1]^T$

$m_1 = \begin{bmatrix} 4 \\ 4 \end{bmatrix}, m_2 = \begin{bmatrix} -4 \\ 4 \end{bmatrix}, m_3 = \begin{bmatrix} 0 \\ -4 \end{bmatrix}$ $C_p = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$

$C^{-1} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$

Mahalanobis MDC discriminant function:

$C = 2I$

$$g_k(x) = m_k^T C^{-1} x - 0.5 m_k^T C^{-1} m_k =$$

$$= 2m_k^T x - m_k^T m_k$$

ω_1:

$m_1^T x = (4 \quad 4)\begin{pmatrix} 1 \\ 1 \end{pmatrix} = 4 + 4 = 8$

$m_1^T m_1 = (4 \quad 4)\begin{pmatrix} 4 \\ 4 \end{pmatrix} = 16 + 16 = 32$

$g_1(x) = 8 - 32 = -16$

ω_2:

$m_2^T x = (-4 \quad 4)\begin{pmatrix} 1 \\ 1 \end{pmatrix} = -4 + 4 = 0$

$m_2^T m_2 = (-4 \quad 4)\begin{pmatrix} -4 \\ 4 \end{pmatrix} = 16 + 16 = 32$

$g_2(x) = -32$

ω_3:

$m_3^T x = (0 \quad -4)\begin{pmatrix} 1 \\ 1 \end{pmatrix} = -4$

$m_3^T m_3 = (0 \quad -4)\begin{pmatrix} 0 \\ -4 \end{pmatrix} = 16$

$g_3(x) = -4 - 16 = -20$

$g_1(x) > g_3(x) > g_2(x)$

$\Rightarrow x$ belongs to ω_1

(c)

The decision functions are:

$g_1(x) = 2(4x_1 + 4x_2) - 32 = 8x_1 + 8x_2 - 32$

$g_2(x) = -8x_1 + 8x_2 - 32$

$g_3(x) = -8x_2 - 16$

Decision hyperplanes:

→ ω_1 / ω_2:

$g_1(x) = g_2(x) \Leftrightarrow 8x_1 + 8x_2 - 32 = -8x_1 + 8x_2 - 32 \Leftrightarrow$

$\Leftrightarrow 16x_1 = 0 \Leftrightarrow x_1 = 0$

→ ω_1 / ω_3

$g_1(x) = g_3(x) \Leftrightarrow 8x_1 + 8x_2 - 32 = -8x_2 - 16 \Leftrightarrow$

$\Leftrightarrow 8x_1 + 16x_2 + 16 = 0 \Leftrightarrow$

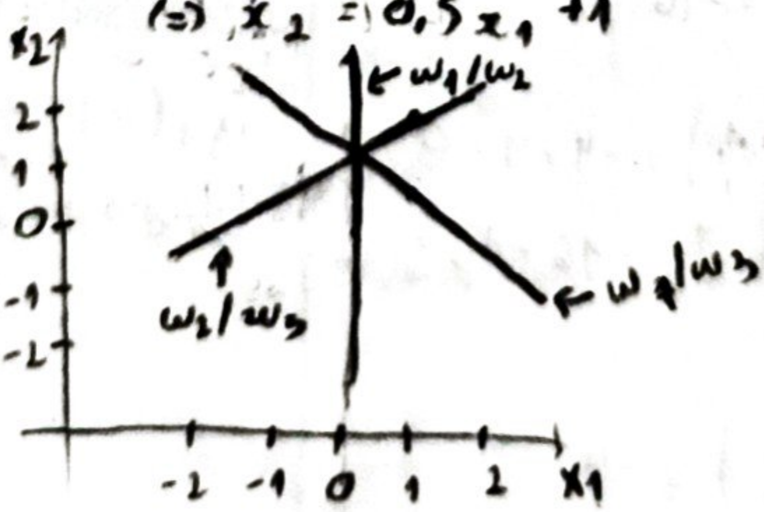
$\Leftrightarrow x_1 + 2x_2 - 2 = 0 \Leftrightarrow x_2 = -0{,}5x_1 + 1$

→ ω_2 / ω_3

$g_2(x) = g_3(x) \Leftrightarrow -8x_1 + 8x_2 - 32 = -8x_2 - 16 \Leftrightarrow$

$\Leftrightarrow -8x_1 + 16x_2 - 16 = 0 \Leftrightarrow$

$\Leftrightarrow x_2 = 0{,}5x_1 + 1$



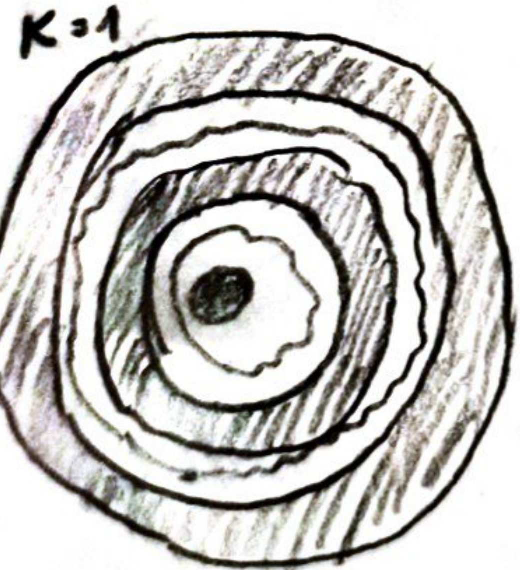
(d) No, a one-vs-all would create binary classifiers for each ω_i vs $\bigcup_{j \neq i} \omega_j$. The "all" class would have its own mean vector (mean of oposing classes) and its own cov matrix. The classifier in (b) just uses $\arg\max_k g_k(x)$ to determine the class (ω_k) of x.

(3)

(a) No, we cannot draw a straight line that perfectly separates the classes.

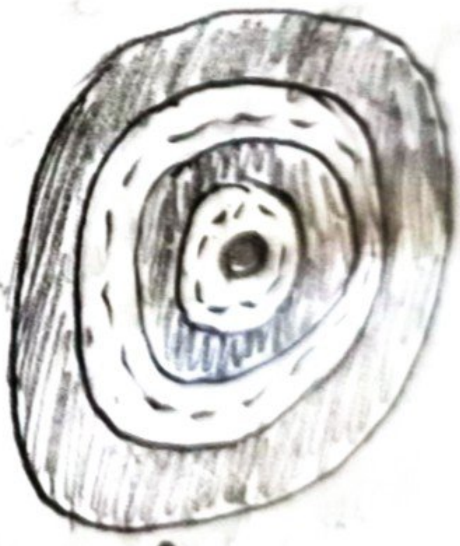
(b) Yes, it should be able to classify the data. A small K might capture too much detail and noise while a exceedingly large K might start favoring the classes with the most total samples. We should find a good in-between balance for K.

(c) K=1



Noisy

K=3



Smooth

(4)

(a)

Bagging:

- Uses ressampling to create multiple datasets.
- Trains a classifier on each set independently
- Goal: Reduce variance

Boosting

- Uses the same training set but reweights data points (increase weight of failed samples).
- Trains classifiers iteratively. At each iteration, it focuses on the shortcomings of the previous model by increasing weights of misclassified samples.
- The inflence of each weak learner in the final decision is weighted by α_m (its performance on the training set).
- Goal: Reduce bias and variance.

(b) $$H(x) = \text{sign}\left(\sum_{j=1}^{T} \alpha_j H_j(x)\right)$$

α_j: weight of the j-th weak learner.

$H_j(x)$: output of week learner j

$w_1 \equiv +1$

$w_2 \equiv -1$

(c) Get result from each weak classifier:

$H_1(x) = -1 \quad (x_2 = 1 < 6)$

$H_2(x) = +1 \quad (x_1 < 7{,}77)$

$H_3(x) = -1 \quad (x_2 < 9.00)$

Weight

$H(x) = -0{,}76 + 0{,}75 - 0{,}63 = -0{,}64$

$\text{sign}(-0{,}64) = -1 \Rightarrow x$ is classified as ω_2

(5)
```
def optimize_svm(Xtr, Ttr, Cs, n_runs):
    MeanF1 = []
    stdF1 = []
    Size = len(Xtr[0]) #num samples
    Xtrain = Xtr[0:Size*0,7]
    Ttrain = Ttr[0:Size*0,7]
    Tval = Ttr[Size*0,7:]
    Xval = Xtr[Size*0,7:]
    for c_val in Cs:
        f1s = []
        for _ in range(n_runs):
            svm = SVC(C=c_val):
            svm.fit(Xtrain, Ttrain)
            predval = svm.predict(Xval)
            f1s.append(f1_score(Tval, predval))
        MeanF1.append(mean(f1s))
        StdF1.append(std(f1s))
    return (MeanF1, StdF1)
```

Maybe add standard scaler