27.09.2024

BITS F464: Machine Learning (1st Sem 2024-25)

LINEAR DISCRIMINANT ANALYSIS

Chittaranjan Hota, Sr. Professor
Dept. of Computer Sc. and Information Systems
hota@hyderabad.bits-pilani.ac.in

Linear Discriminant Functions: Applications

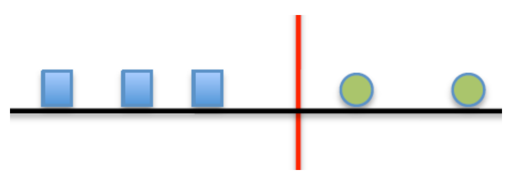
- Fisher's Linear Discriminant Analysis for reducing the number of features required for Face Recognition.
- Classifying patient's disease state as Mild, Moderate or Severe.
- Identifying the type of customers who might buy a particular product.

Linear Discriminant Functions

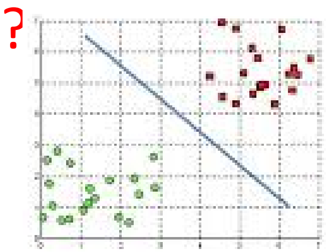
- Used to discriminate between two or more classes based on a set of predictor variables.

- Learns the mapping between feature vector and class labels.
- Does it not create a decision boundary?

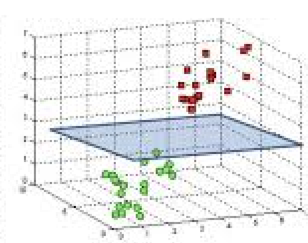
[Hyperplanes]()



(1-D, a point/ threshold)



(2-D, a line)



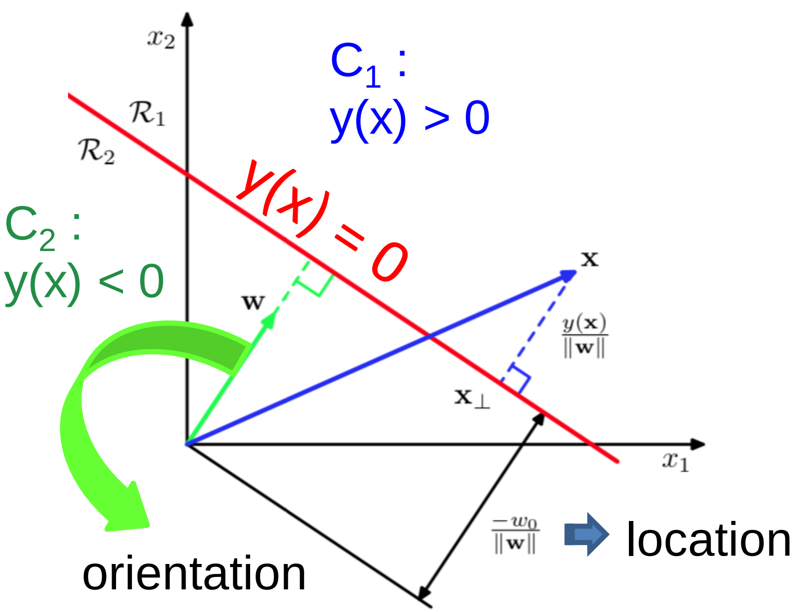
(3-D, a plane)

Logistic Regression may be unstable for well separated classes and few examples. Why?

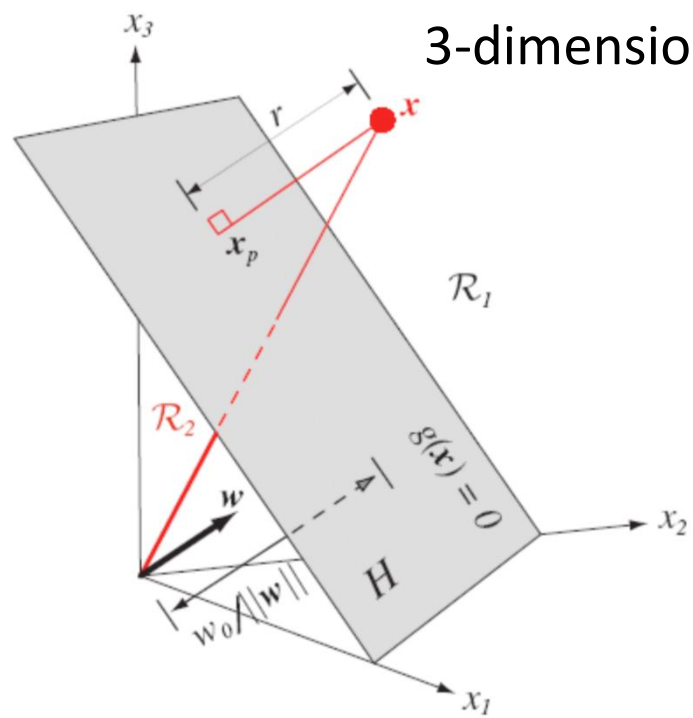
Two-class Linear Discriminant Functions (K=2)

- $y(x) = w^T x + w_0 = \sum_{i=1}^{d} w_i x_i + w_0$

Where, w^T is the weight vector and w_0 is the bias. The negative of bias (i.e. $-w_0$) sometimes is called as threshold.

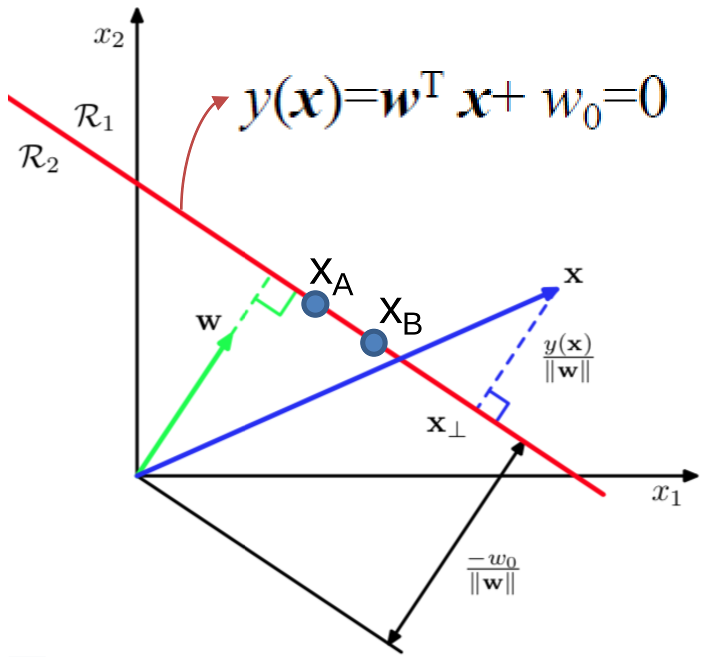


3-dimension



(geometry of LDF in 2-dimension)

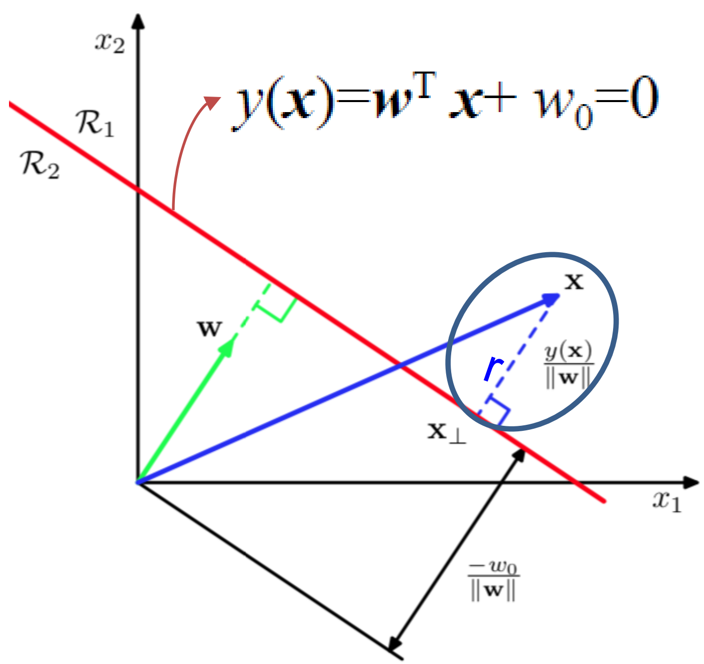
Distance of Origin to Decision Surface (Bias: w_0)



- Let x_A and x_B be points that lie on the decision surface.
- $y(x_A) = y(x_B) = 0$
- $w^T x_A + w_0 = w^T x_B + w_0 = 0$
- $w^T (x_A - x_B) = 0$
- $x_A - x_B$ is an arbitrary vector parallel to the line.
- Hence, <u>w is orthogonal</u> to every vector lying on the decision surface. orientation
- If x is a point on the decision surface, $y(x) = 0 \rightarrow w^T x = -w_0$
- So, the normal distance from origin to decision surface:

$$\frac{w^T x}{\|w\|} = -\frac{w_0}{\boxed{\|w\|}} \Rightarrow \sqrt{w_1^2 + .. + w_{M-1}^2}$$

Distance of a point ‘x’ to the Decision surface (r)



Let ‘x’ be an arbitrary point and $\mathbf{x}_\perp$ be it’s orthogonal projection on the decision surface.

$$\mathbf{x} = \mathbf{x}_\perp + r\frac{\mathbf{w}}{\|\mathbf{w}\|}$$ by vector addition

Second term is a normalized vector to the decision surface, which is collinear with ‘w’.

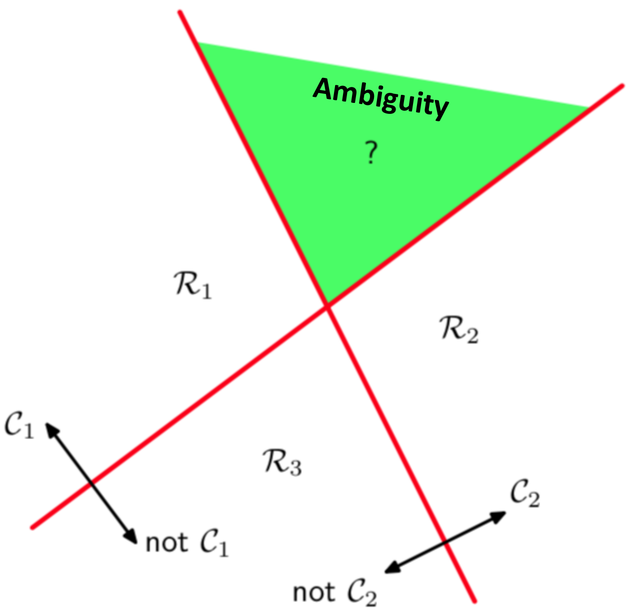
As $\frac{\mathrm{w}}{\|\mathrm{w}\|} = 1$, we need to scale it by r.

As $y(\mathbf{x}_\perp) = 0$ and $w^T w = ||w||^2$ ➡ $y(x) = w^T x + w_0 = w^T (\mathbf{x}_\perp + r\frac{\mathbf{w}}{\|\mathbf{w}\|}) + w_0$

➡ $\underline{w^T \mathbf{x}_\perp + w_0} + r\frac{w^T w}{\|\mathbf{w}\|} = 0 + r\frac{||w||^2}{\|\mathbf{w}\|} = r.||w||$ ➡ $r = y(x) / ||w||$

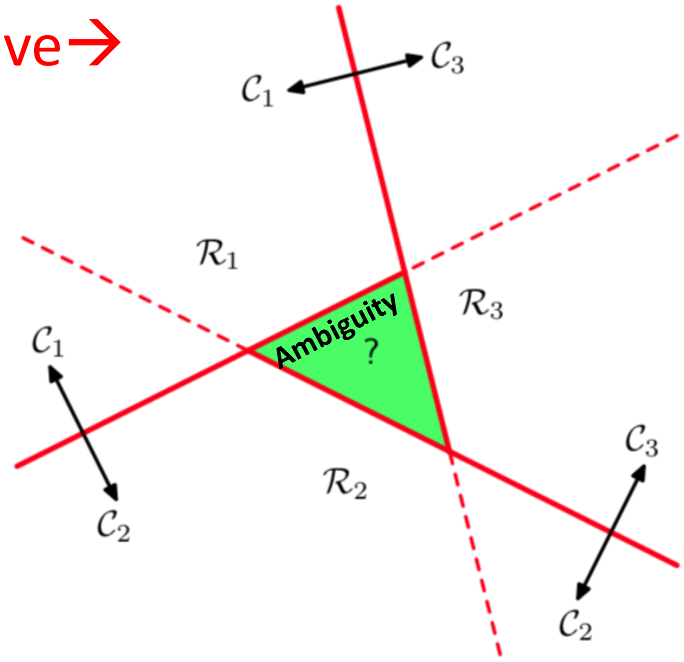
Multi-class Linear Discriminant Functions (K>2)

Approach 1: By combining a number of two-class discriminant functions.



K = ?

Alternative→



K = ?

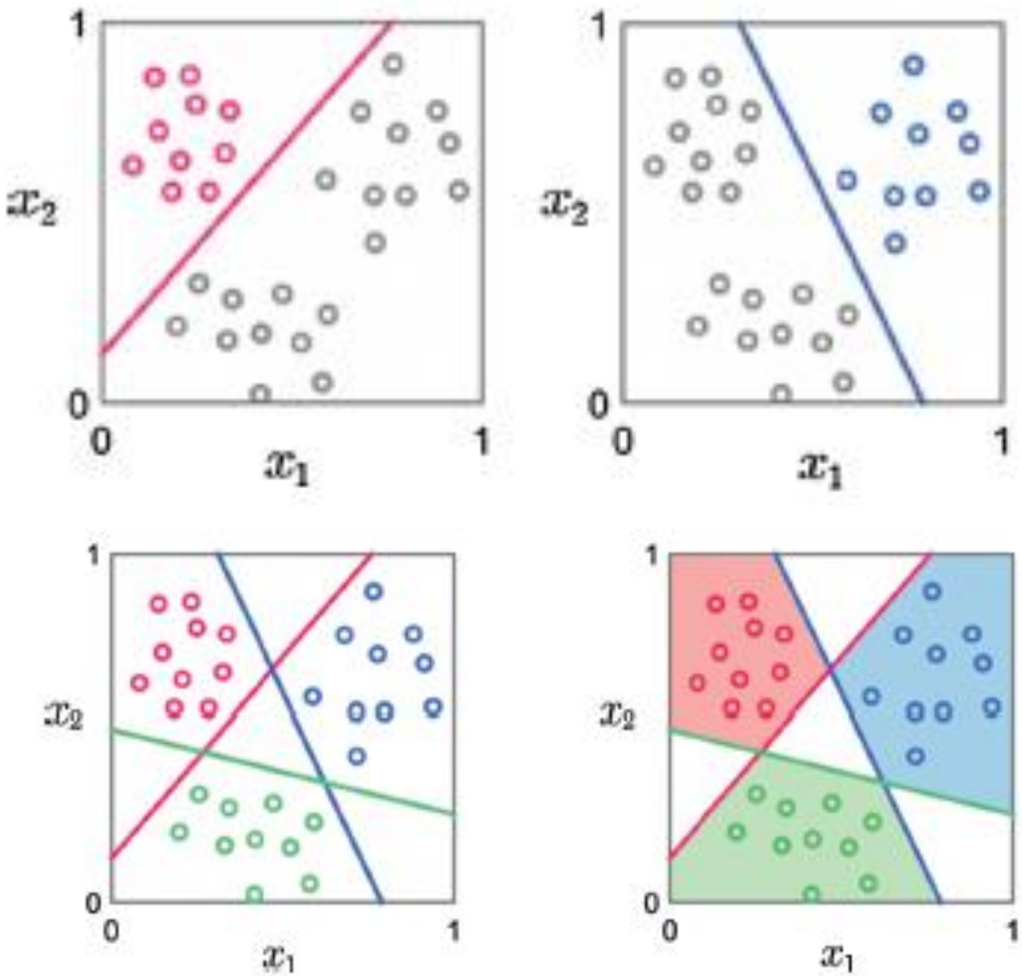
(K-1 classifiers with each one separating points in a particular class C_k from points not in that class)

one-versus-the-rest

(K (K-1)/2) classifiers with one for every possible pair of classes. Each point is classified according to a majority vote.

one-versus-one

Another Example...



$$b_c + x_p^T w_c > 0$$

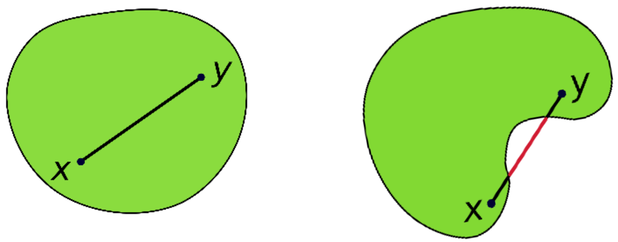
$$b_j + x_p^T w_j < 0,\ j = 1 \ldots c,\ j \mathrel{!}= c$$

$$y = \boxed{\underset{j = 1, \ldots c}{\operatorname{argmax}}}\ b_j + x^T w_j$$

global maximum

Solution: Using K-discriminant functions

- Building a single K-class discriminant comprising K-linear functions of the form:
  - $y_k(x) = w_k^T x + w_{k0}$

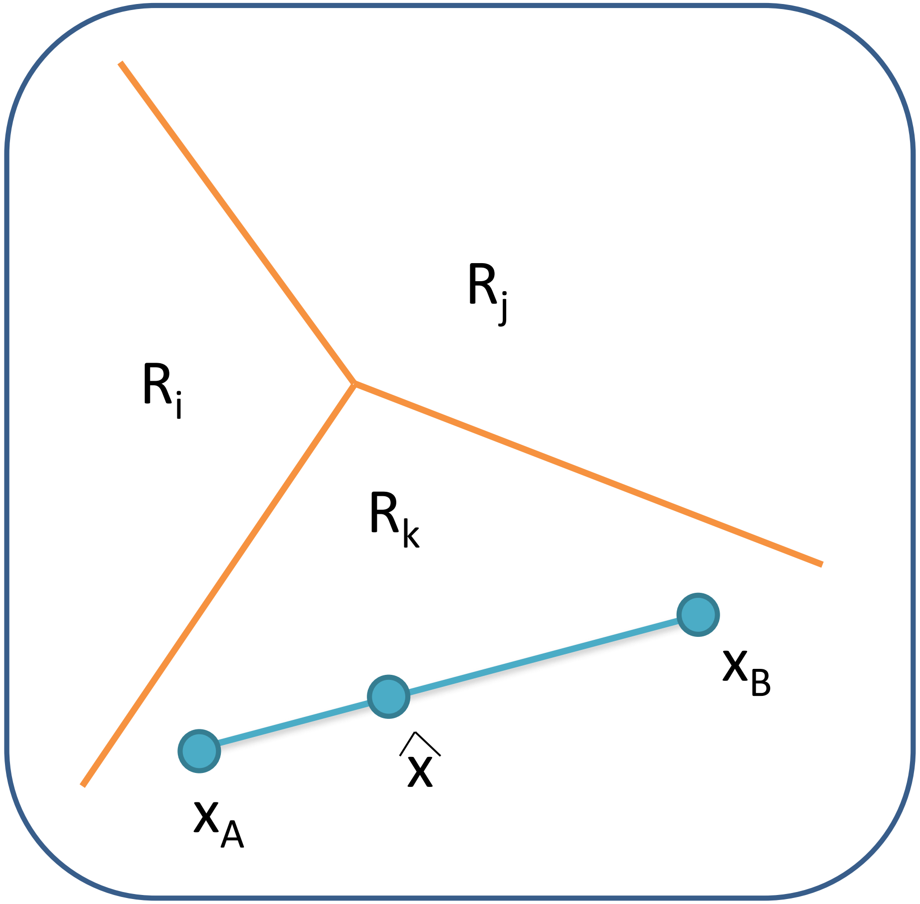


- Then, assigning a point x to class C_k if $y_k(x) > y_j(x), \forall j \neq k$.
- The decision boundary between C_k and C_j : $y_k(x) = y_j(x)$
- Defined by: $(w_k - w_j)^T x + (w_{k0} - w_{j0}) = 0 \longrightarrow$ Same as 2-class
- Hence, same geometrical properties apply.

Decision regions of such discriminants are always singly connected and convex.

Proof of Convexity Next…

Proof of Convexity of Decision Region



$$\hat{\boldsymbol{x}} = \lambda \boldsymbol{x}_A + (1-\lambda)\boldsymbol{x}_B$$

Where, $0 \leq \lambda \leq 1$

From the Linearity of Discriminant functions:

$$y_k(\hat{\mathbf{x}}) = \lambda y_k(\mathbf{x}_A) + (1-\lambda) y_k(\mathbf{x}_B)$$

As X_A and X_B lie inside R_K :

$y_k(\boldsymbol{x}_A) > y_j(\boldsymbol{x}_A)$ and $y_k(\boldsymbol{x}_B) > y_j(\boldsymbol{x}_B)$

$\forall\ j \neq k$ ❯ $y_k(\hat{\mathbf{x}}) > y_j(\hat{\mathbf{x}})$ ❯ $\hat{\boldsymbol{x}}$ Lies in R_K

$\hat{x}$ must also lie in R_K $\longrightarrow$ R_K is Singly Connected and Convex

Multi-class Classification using LDA (sklearn)

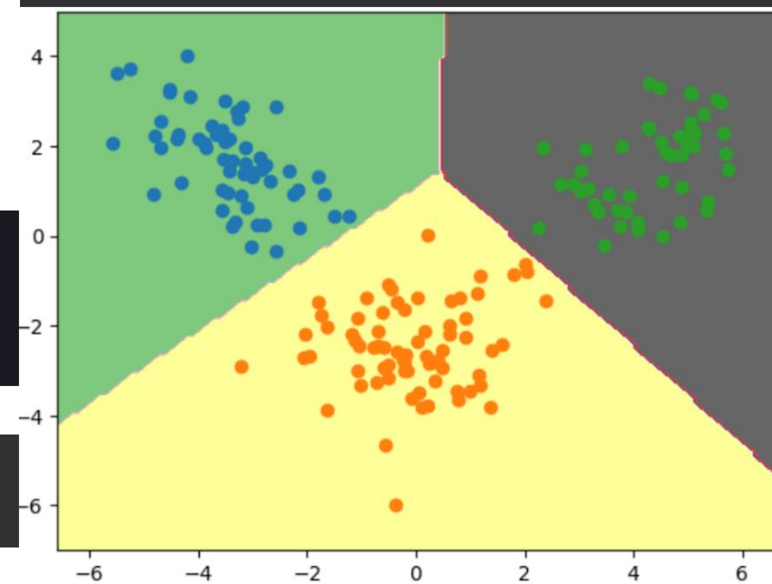
```
from sklearn.datasets import load_wine
dt = load_wine()
X = dt.data
y = dt.target
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[13,  0,  0],
       [ 0, 26,  0],
       [ 0,  1, 14]])
```

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3)

lda.fit(X_train,y_train)




```
y_pred = lda.predict(X_test)
print(accuracy_score(y_test,y_pred))
```

```
0.9814814814814815
```

```
num_records = wine_data.data.shape[0]
print(num_records)
```

Alcohol, magnesium, hue, proline, ...

Least Squares for Classification

- Straightforward way to adapt regression techniques for classification tasks.

- How do we compute y(x), and w_0, w_1, w_2, …w_d ?
- Each class C_k is described by its own linear model:
  - $y_k(x) = w_k^T x + w_{k0}$ (where x and w have D dimensions each)
- We can group these together using a vector notation:

$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^{\mathrm{T}} \boxed{\widetilde{\mathbf{x}}}$$

Augmented input vector $(1, \mathbf{x}^{\mathrm{T}})^{\mathrm{T}}$

A Parameter matrix whose k^{th} column is a D+1-dimensional vector: $\widetilde{\mathbf{w}}_k = (w_{k0}, \mathbf{w}_k^{\mathrm{T}})^{\mathrm{T}}$

A new input x is then assigned to a class for which the output is largest. $y_k = \widetilde{\mathbf{w}}_k^{\mathrm{T}} \widetilde{\mathbf{x}}$ Get $\tilde{\mathbf{w}}$ by minimizing the Sum-of-squares.

An example classification using LDF

- Suppose we have a dataset of two classes: **Class A** and **Class B**. Each data point has two features, x_1 and x_2. Our task is to classify new data points into either Class A or Class B using a linear discriminant function.
- Class A (positive class): X_A = {(2,3), (3,3), (4,5), (5,6)} & **Class B** (negative class): X_B = {(1,1),(2,2),(3,1),(4,2)}
- Step 1: Define the Linear discriminant function:

$g(x) = w_1x_1 + w_2x_2 + w_0$, Decision rune is: If $g(x) > 0$, classify as Class A, else Class B.

- Step 2: Train the classifier:

Suppose, after training we get the LDF as: $g(x) = 2x_1 + 3x_2 - 15$

Least squares

- Step 3: Classify new points:

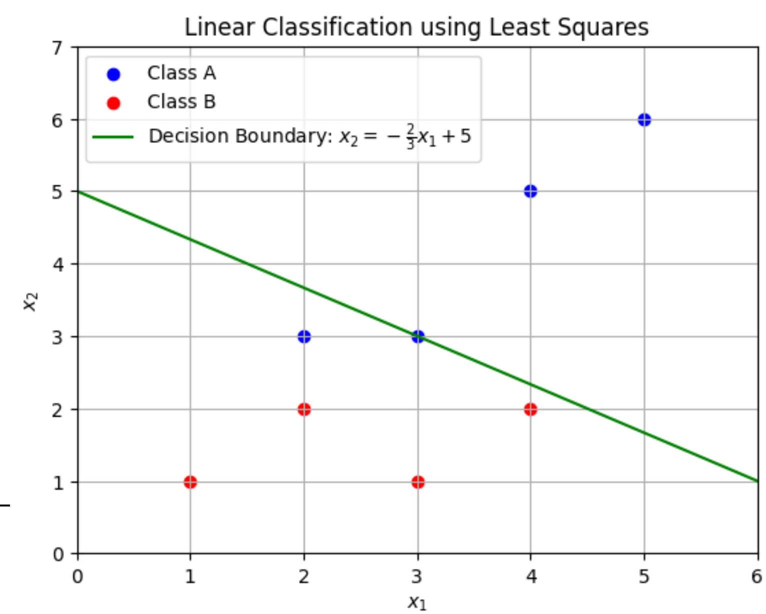
$x = (3,4) \Rightarrow g(3,4) = 2X3 + 3X4 -15 = 3$

$\Rightarrow$ As, $g(3,4) = 3 > 0$, we classify it as Class A.

What class a point (2,1) will belong to?

- Decision boundary:

$g(x) = 0 \Rightarrow x_2 = (-2/3).x_1 + 5$



Minimizing sum-of-squares error func

Let there be a training dataset $\{x_n, t_n\}$ where n = 1, …N

Define a matrix T whose n^{th} row is the vector t_n^T and matrix $\widetilde{\mathbf{X}}$ whose n^{th} row is: $\widetilde{\mathbf{x}}_n^{\mathrm{T}}$.

Then, the sum-of-squares error function can be written as:

$$E_D(\widetilde{\mathbf{W}}) = \frac{1}{2}\mathrm{Tr}\left\{\boxed{(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})^{\mathrm{T}}(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})}\right\}$$

Multiplying a matrix with its's transpose results in a square matrix.

Taking a Trace of this square matrix (sum of the elements on the main diagonal)

LSE Computation: An Example

Let, $X = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$, $T = \begin{bmatrix} 5 \\ 11 \\ 17 \end{bmatrix}$

Let, initial value of W:

$$W = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

$$\Rightarrow XW = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \times \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 1.5 \\ 3.5 \\ 5.5 \end{bmatrix}$$

Now, Error between predicted and target:

$$XW - T = \begin{bmatrix} 1.5 \\ 3.5 \\ 5.5 \end{bmatrix} - \begin{bmatrix} 5 \\ 11 \\ 17 \end{bmatrix} = \begin{bmatrix} -3.5 \\ -7.5 \\ -11.5 \end{bmatrix}$$

Squaring the error vector:

$$(XW - T)^T \cdot (XW - T) = \begin{bmatrix} -3.5 & -7.5 & -11.5 \end{bmatrix} \times \begin{bmatrix} -3.5 \\ -7.5 \\ -11.5 \end{bmatrix}$$

$$= (3.5)^2 + (7.5)^2 + (11.5)^2 = 158.75$$

Now, take Trace of the resulting square matrix:

$$Tr\left((XW - T)^T \cdot (XW - T)\right) = Tr\left(\begin{bmatrix} 158.75 \end{bmatrix}\right) = \underline{158.75}$$

$$\Rightarrow \text{Least Square Error (LSE)} = \frac{1}{2} \times 158.75 = \underline{\underline{79.375}}$$

Goal: Minimize LSE by adjusting W.

Minimizing Sum-of-Squares

$$E(w) = \frac{1}{2}\mathrm{tr}\left((Xw-t)^T(Xw-t)\right) \qquad E(w) = \frac{1}{2}\mathrm{tr}\left(w^TX^TXw - w^TX^Tt - t^TXw + t^Tt\right)$$

Since $\mathrm{tr}(w^TX^Tt)$ and $\mathrm{tr}(t^TXw)$ are scalars, and $\mathrm{tr}(AB) = \mathrm{tr}(BA)$, we can combine these two terms:

$$E(w) = \frac{1}{2}\mathrm{tr}\left(w^TX^TXw\right) - \mathrm{tr}\left(t^TXw\right) + \frac{1}{2}\mathrm{tr}\left(t^Tt\right)$$

$$\frac{\partial}{\partial w}\mathrm{tr}(t^TXw) = X^Tt$$

$$\frac{\partial}{\partial w}\left(\frac{1}{2}\mathrm{tr}\left(w^TX^TXw\right)\right) = X^TXw$$

0

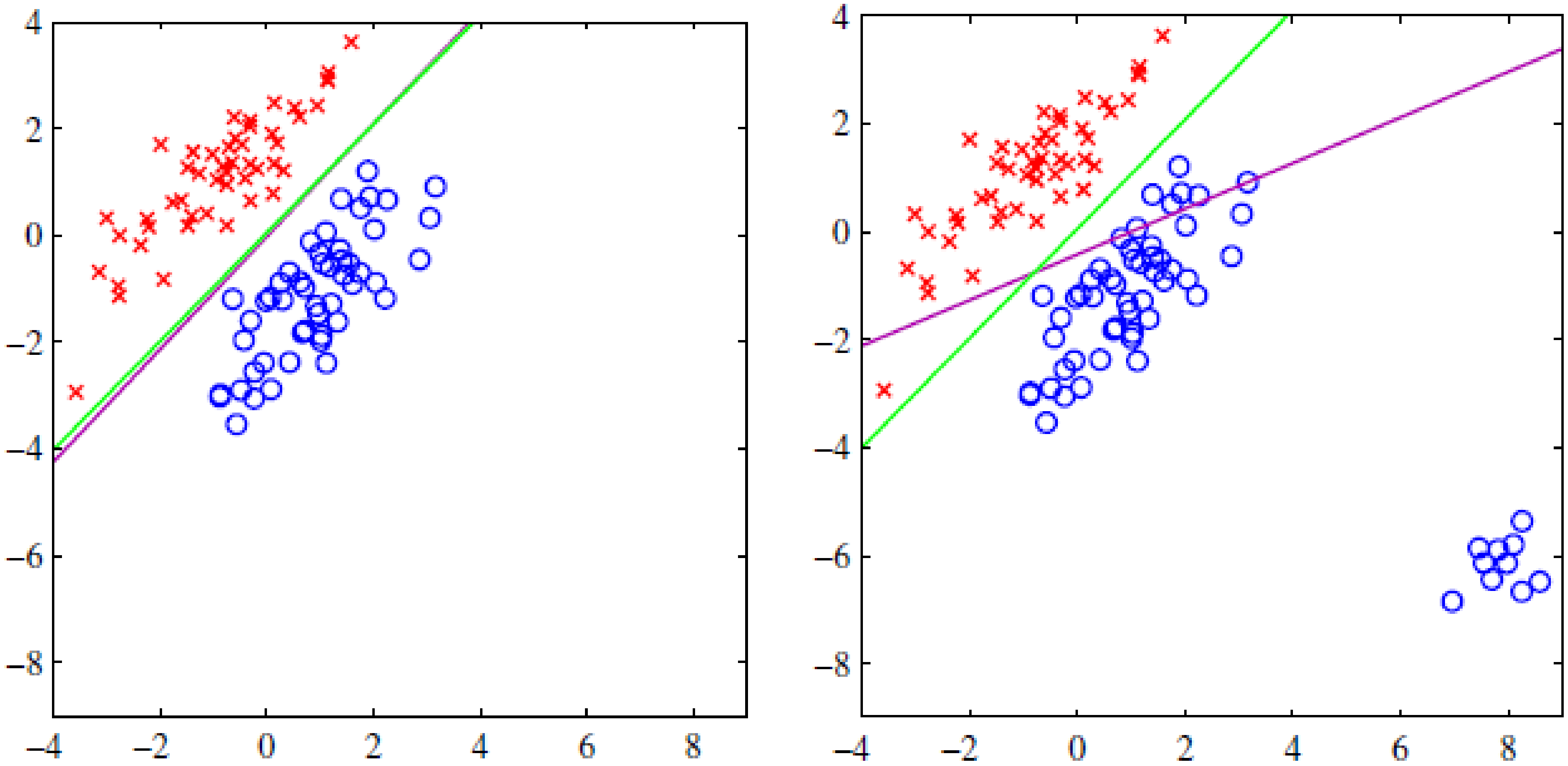
To minimize the error, set the derivative equal to zero:

$$\frac{\partial E(w)}{\partial w} = X^TXw - X^Tt \quad = 0$$

$$w = (X^TX)^{-1}X^Tt$$

pseudoinverse solution

Least-squares: highly sensitive to outliers

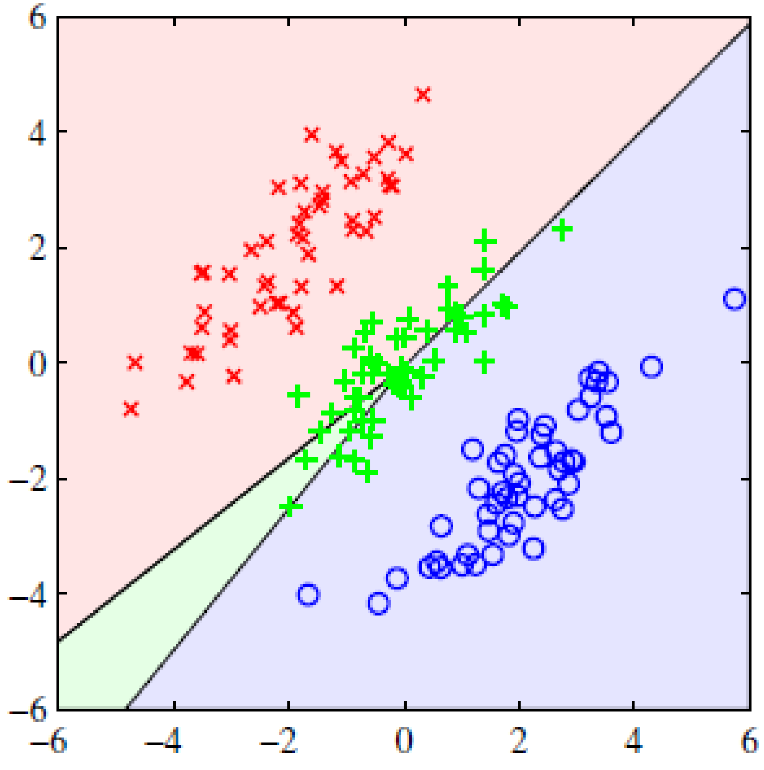


Magenta: Least squares, Green: Logistic regression

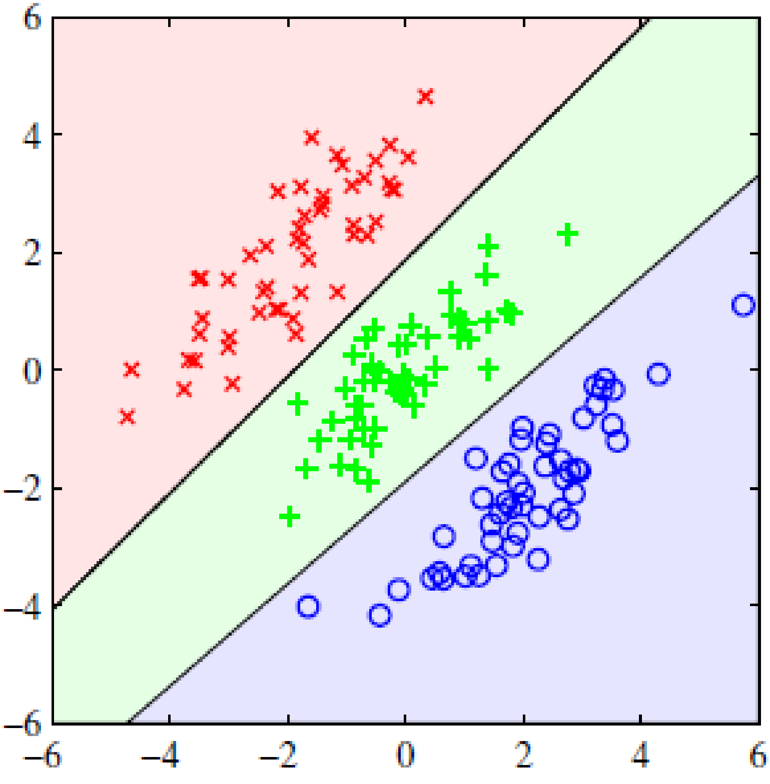
Least squares: more severe problems

Certain datasets: Unsuitable

3-classes, 2-D space, synthetic data



(Least-squares classification)

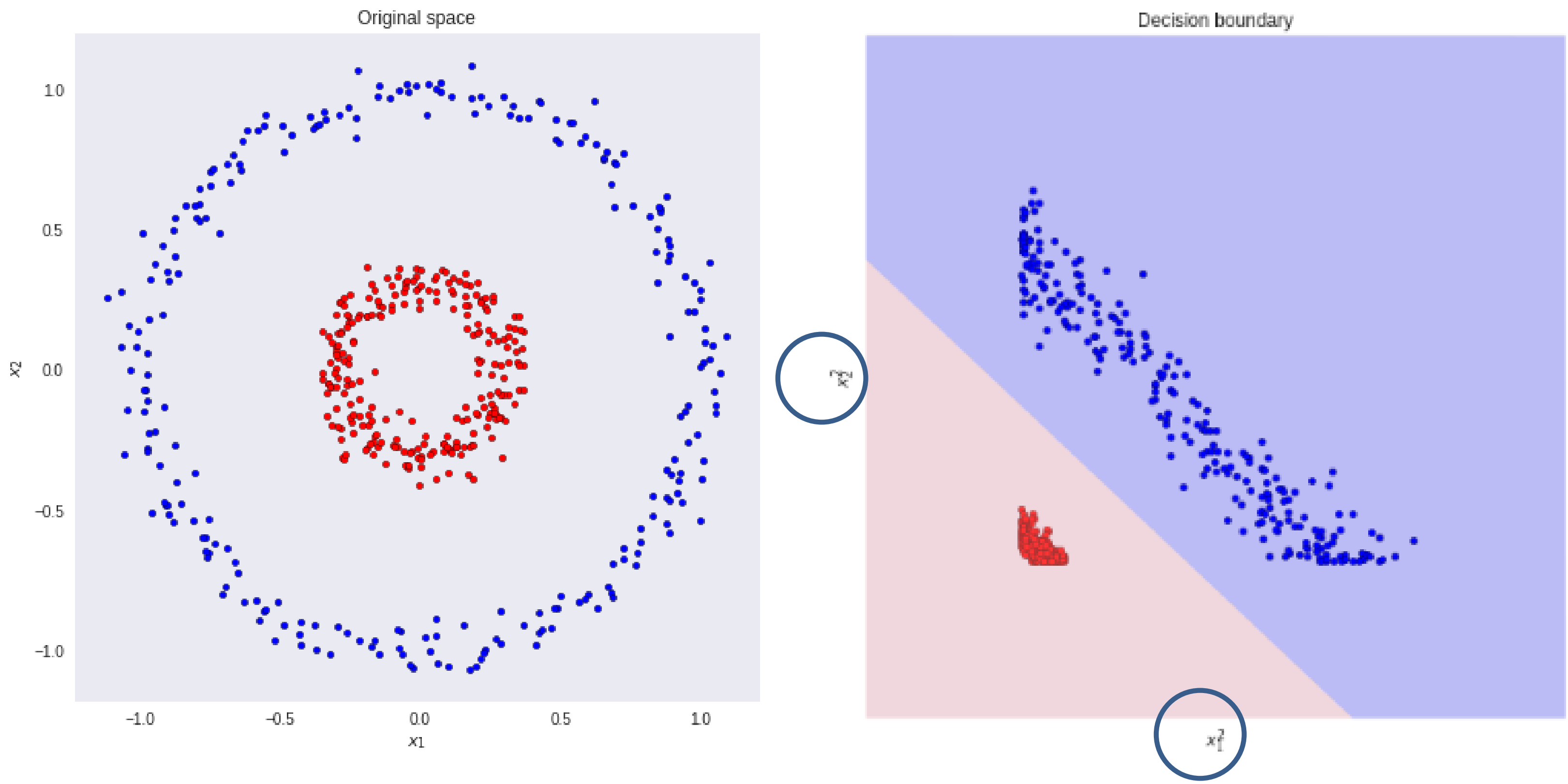


(Logistic Regression classification)

The region of input space assigned to the green class is too small and so most of the points from this class are misclassified.

Fisher’s Linear Discriminant: Motivation

- Why do we need it?



Question: How difficult are these transformations to figure out?

Image source: https://sthalles.github.io/

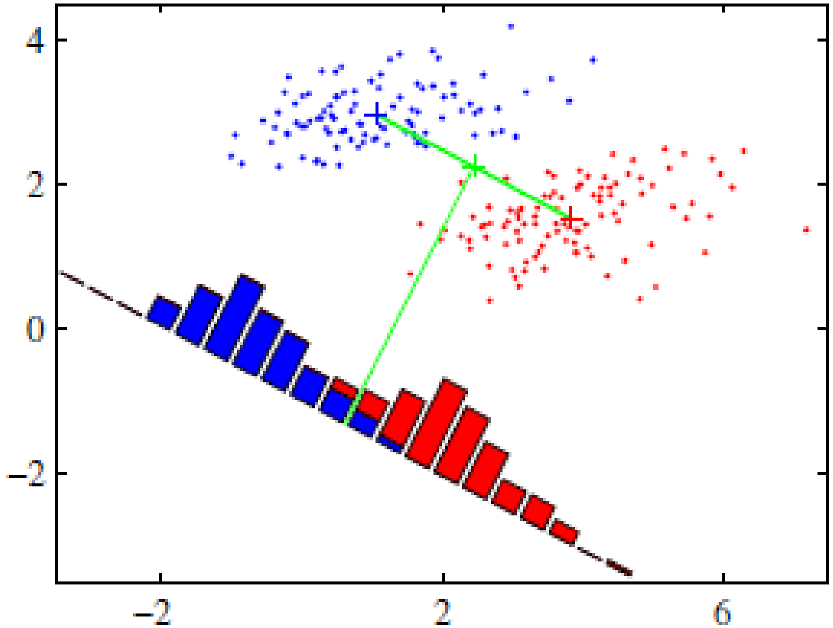
Fisher's Linear Discriminant


Ronald A. Fisher

- View classification in terms of dimensionality reduction
  - Project D-dimensional input vector x into one dimension using: $y = w^T x$
  - Place threshold on y to classify $y \geq -w_0$ as class C_1 else class C_2
  - We get a standard linear classifier
- Classes well-separated in D-dimension space may strongly overlap in 1-dimension
  - Adjust component of the weight vector w
  - Select projection to maximize class-separation

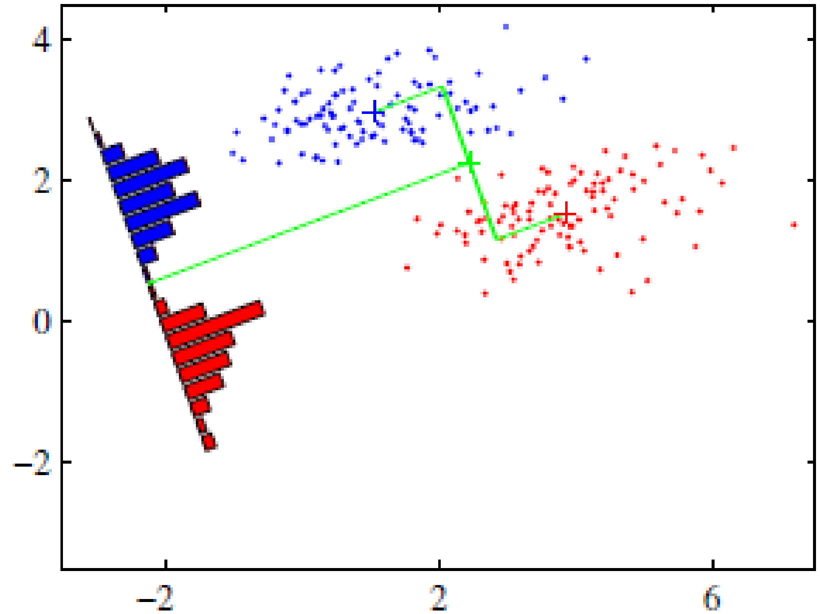
FLD seeks to maximize the ratio of between-class variance to within-class variance, thus maximizing class discrimination.

An illustration of Fisher's LDF



(Projection onto the line joining the class means)

What is the degree of class overlap?



(Projection based on Fisher's Linear discriminant function)

Is the class separation improved?

Maximizing Mean Separation

- Let us consider a two-class problem with N_1 points of C_1 class and N_2 points of C_2 class
- Mean vectors: $m_1 = \frac{1}{N_1} \sum_{n \varepsilon C_1} x_n \qquad m_2 = \frac{1}{N_1} \sum_{n \varepsilon C_2} x_n$
- Choose w to best separate class means:
- Maximize $m_2 - m_1 = w^T(m_2 - m_1)$, where $m_k = w^T m_k$ is the mean of the projected data from class C_k
- Can be made arbitrarily large by increasing the magnitude of w:
  - We could have w to be of unit length i.e. $\sum_i w_i^2 = 1.$
  - Using a Lagrange multiplier, maximize

$\mathbf{w} \propto (\mathbf{m}_2 - \mathbf{m}_1).$ There is still a problem with this approach…

Illustration of the problem

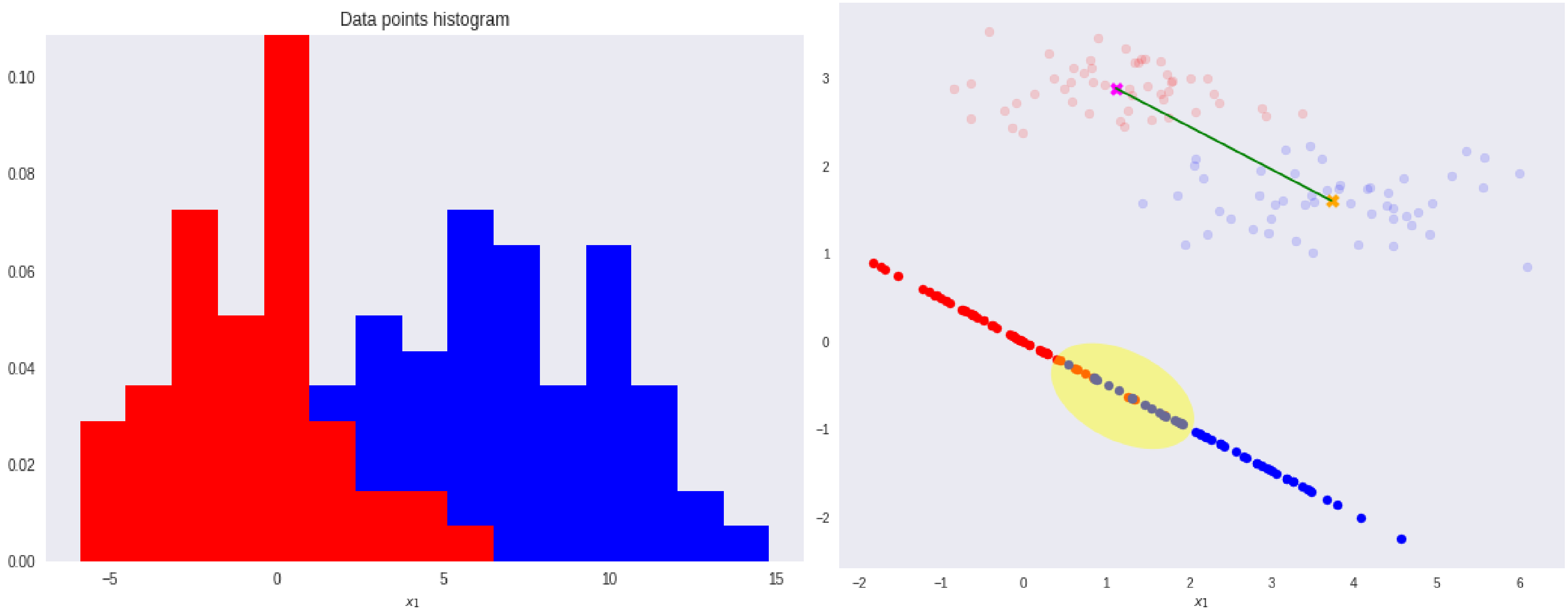


Image source: https://sthalles.github.io/

After re-projection, the data exhibit some sort of class overlapping - shown by the yellow ellipse on the plot.

This difficulty arises from the strongly non-diagonal co-variances of the class distributions.

Minimizing Variance and Optimizing

- Project D-dimensional input vector x into one dimension using: $y_n = w^T x_n$
- The within-class variance of the transformed data from class C_k is given by:

$$s_k^2 = \sum_{n \in C_k} (y_n - m_k)^2$$

- Total within-class variance for the whole dataset is: $s_1^2 + s_2^2$
- Fisher's criterion: $J(\boldsymbol{w}) = (m_2 - m_1)^2 / s_1^2 + s_2^2$

Rewriting (to make the dependence on w explicit: $J(\boldsymbol{w}) = \boldsymbol{w}^T S_B \boldsymbol{w} \,/\, \boldsymbol{w}^T S_W \boldsymbol{w}$

Where, $S_B = (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T$ is the between-class covariance matrix &

$S_W = \Sigma_{n \in C1}(\boldsymbol{x}_n - \boldsymbol{m}_1)(\boldsymbol{x}_n - \boldsymbol{m}_1)^T + \Sigma_{n \in C2}(\boldsymbol{x}_n - \boldsymbol{m}_2)(\boldsymbol{x}_n - \boldsymbol{m}_2)^T$ the within-class covariance matrix.

Differentiating with respect to w, J(w) is maximized when: $(\boldsymbol{w}^T S_B \boldsymbol{w}) S_W \boldsymbol{w} = (\boldsymbol{w}^T S_W \boldsymbol{w}) S_B \boldsymbol{w}$

Dropping scalar factors, and noting S_B is in the same direction as $m_2 - m_1$ and multiplying both the sides by S_W^{-1} : $\boxed{w \;\alpha\; S_W^{-1}(m_2 - m_1)}$ Fisher's LD

Optimization of J(w)

$$\frac{dJ(w)}{dw} = \frac{\frac{d}{dw}N(w)D(w) - N(w)\frac{d}{dw}D(w)}{(D(w))^2}$$

Quotient rule

$$\frac{d}{dw}N(w) = \frac{d}{dw}(w^T S_B w) = 2S_B w$$

$$\frac{d}{dw}D(w) = \frac{d}{dw}(w^T S_W w) = 2S_W w$$

$$\frac{dJ(w)}{dw} = \frac{2S_B w(w^T S_W w) - 2S_W w(w^T S_B w)}{(w^T S_W w)^2}$$

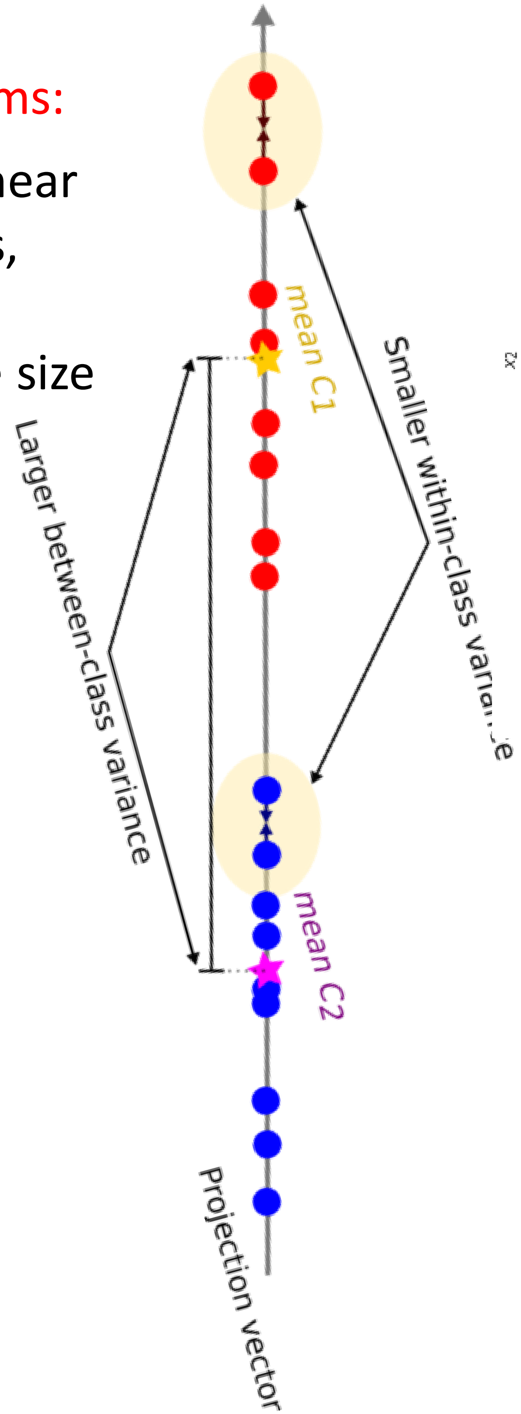
To maximize J(w): $S_B w(w^T S_W w) - S_W w(w^T S_B w) = 0$

$S_W^{-1} S_B w = \lambda w$ This is an **eigenvalue problem**, where λ is the eigenvalue, and w is the eigenvector.

Illustration with Fisher’s LDF

Problems:

Non-linear models, Small sample size



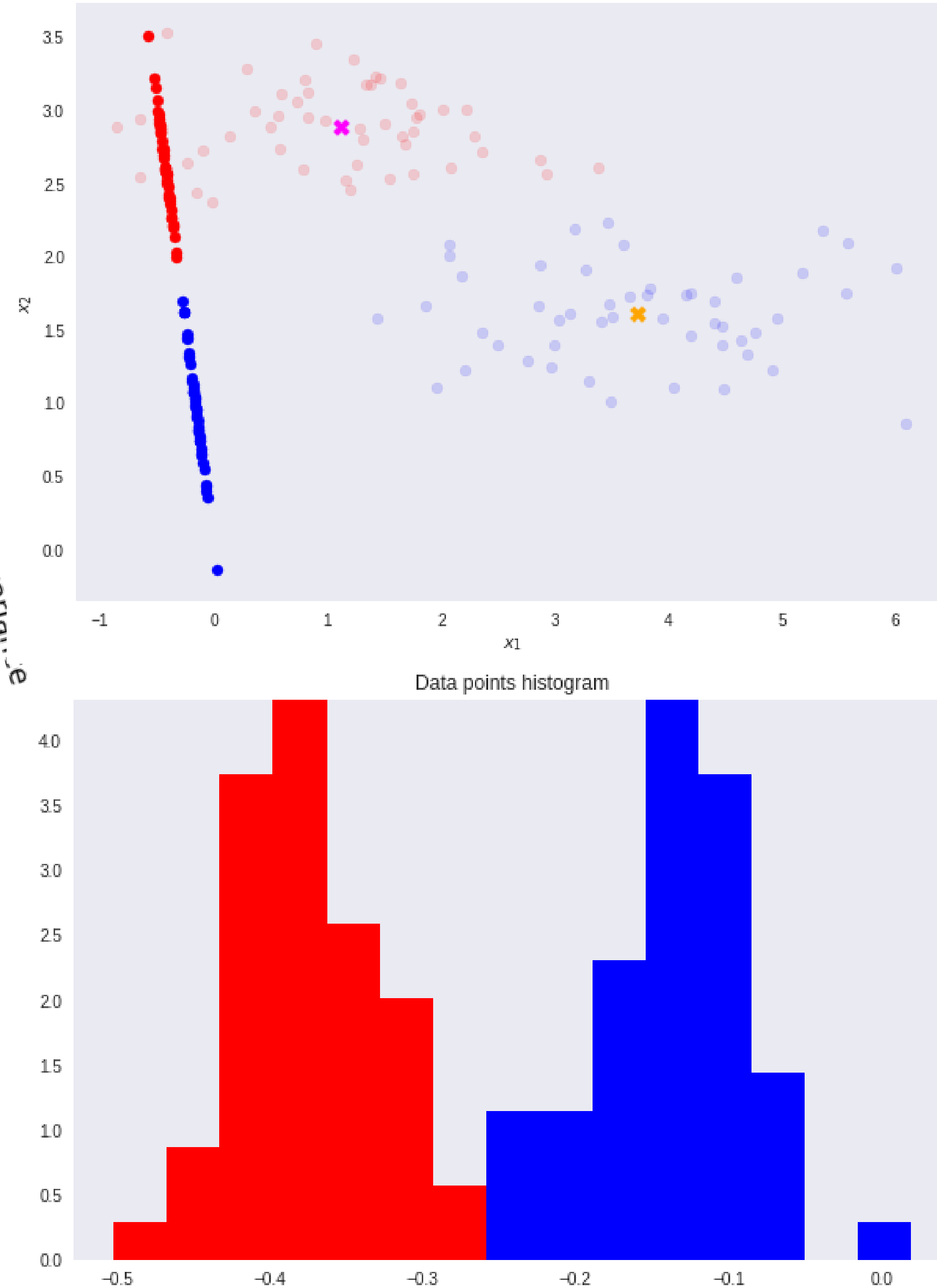


Image source: https://sthalles.github.io/

Fisher's Linear Discriminant Functions

L D A O N I R I S D A T A S E T

```
Original Feature Space:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   target
0       0
1       0
2       0
3       0
4       0

Reduced Feature Space using Fisher's LDA:
        LD1       LD2  target
0  8.061800 -0.300421       0
1  7.128688  0.786660       0
2  7.489828  0.265384       0
3  6.813201  0.670631       0
4  8.132309 -0.514463       0
```

Assignment 3

Ref: https://developer.ibm.com/tutorials/awb-implementing-linear-discriminant-analysis-python/

Thank you!