Linear Classification.

A Modern Theory of Detection and Estimation.
Block-2: Detection

Emilio Parrado-Hernández, emilio.parrado@uc3m.es

November 14, 2022

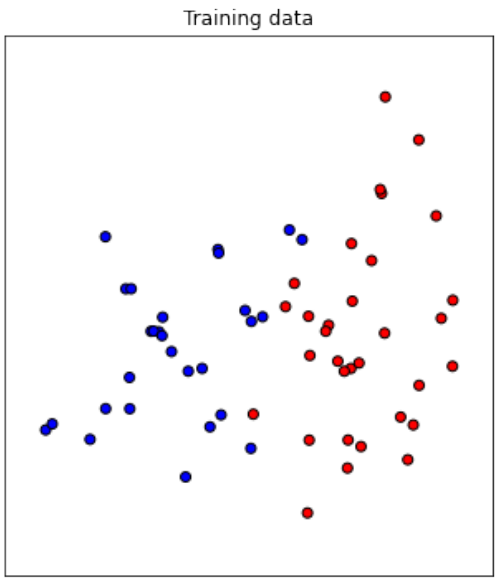
Index

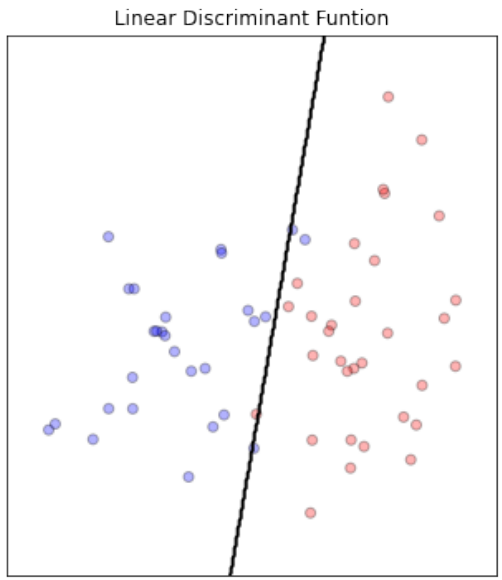
Index

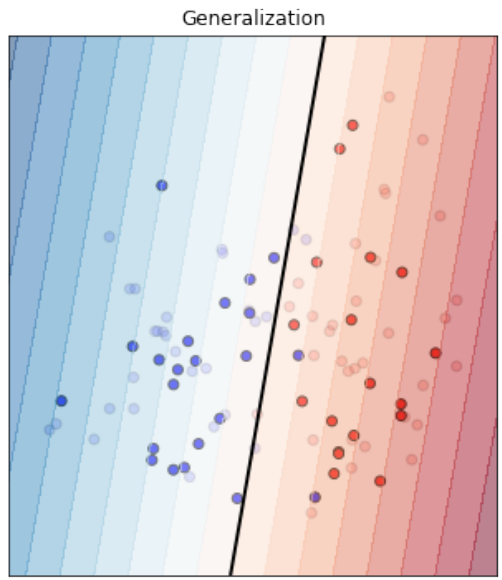
Linear Classifier

Linear discriminant function

$$g(\mathbf{x}) = w_0 + \mathbf{w}^\top \mathbf{x} \begin{array}{c} D=1 \\ \gtrless \\ D=0 \end{array} 0$$







Linear Classifier Motivations

- Problems are generally defined in terms of a **data collection**, not with the true likelihood or the true priors of the hypothesis.
- Sometimes we have the true pdfs but they become **untractable** or very **difficult to handle**
- **Advantages of Linear Classification**
  - Easy to implement
  - Fast operation
  - Optimal when the likelihoods are Gaussian with identical covariance matrices (very common situation in real life applications)

Compact notation

Define targets

$$y_i = \begin{cases} +1 & \text{if } \mathbf{x}_i \in H_1 \\ -1 & \text{if } \mathbf{x}_i \in H_0 \end{cases}$$

Linear discriminant function

$$g(\mathbf{x}) = w_0 + \mathbf{w}^\top \mathbf{x} \begin{array}{c} D=1 \\ \gtrless \\ D=0 \end{array} 0 \rightarrow \hat{y}_t = \text{sign}\left(g(\mathbf{x})\right) = \text{sign}(w_0 + \mathbf{w}^\top \mathbf{x})$$

Compact notation

$$\mathbf{w}_e = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix} \qquad \mathbf{x}_e = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}$$

Therefore

$$g(\mathbf{x}) = \text{sign}(\mathbf{w}_e^\top \mathbf{x}_e)$$

During the rest of the lecture, unless explicitely stated, $\mathbf{x}$ and $\mathbf{w}$ will refer to the compact notations.

Linear separability

Linear separability

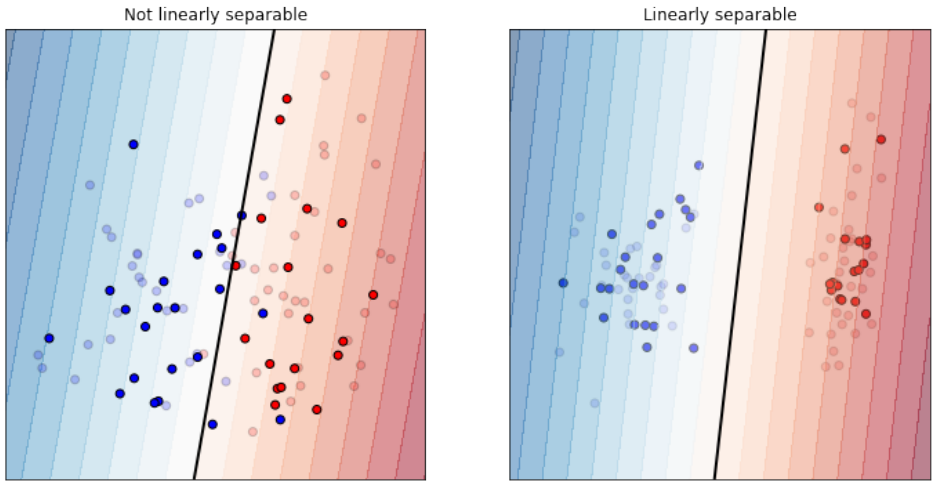
Define the score

$$yg(\mathbf{x}) = y\text{sign}(\mathbf{w}_e^\top \mathbf{x}_e)$$

If $yg(\mathbf{x}) > 0$ for all the observations then the problem is **Linearly separable**

In general we want $yg(\mathbf{x}) > 0$ for a majority of observations.

Any *coherent* set of $d + 1$ observations, where d is the number of variables (dimensionality) of these observations, is linearly separable independently of the values of the targets.



Index

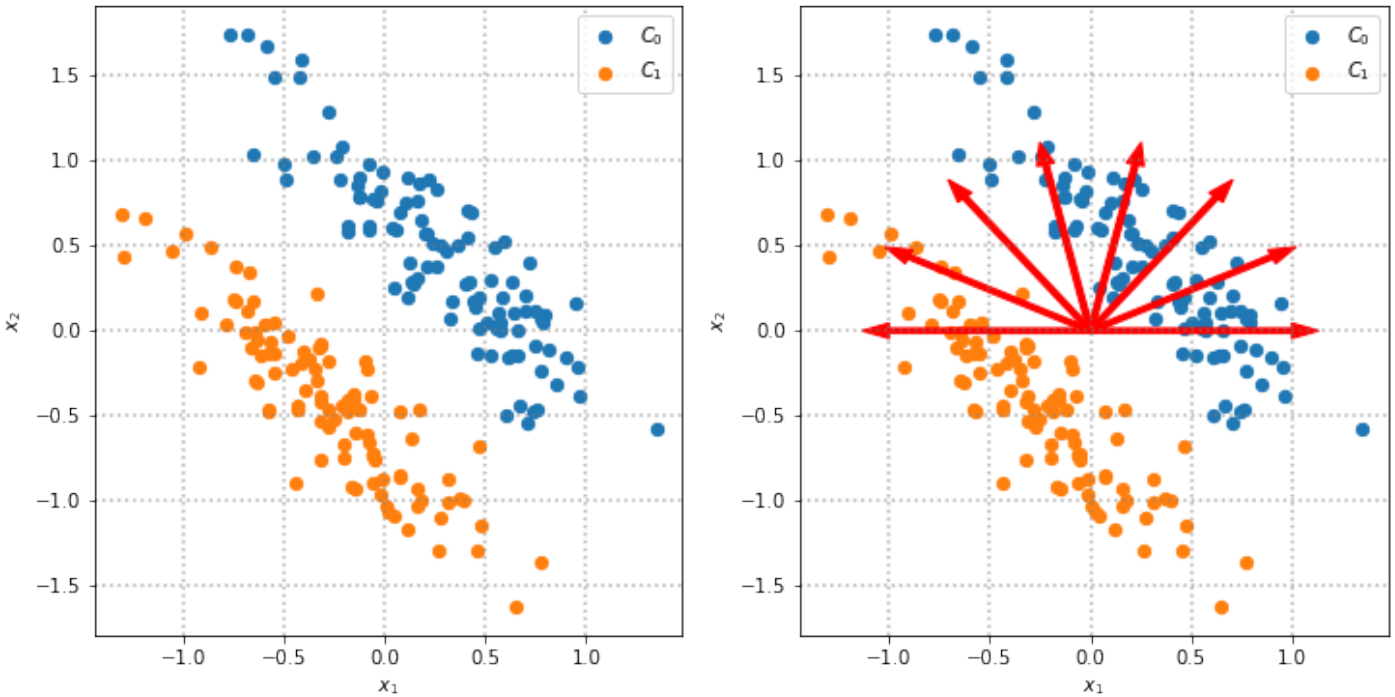
Index

Fisher's criterion of separability

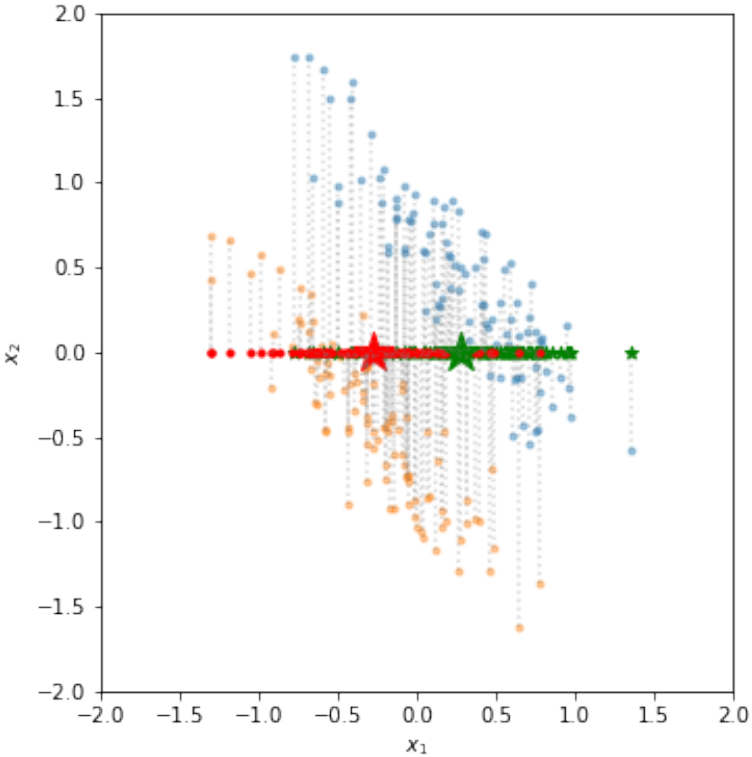
Fisher's criterion

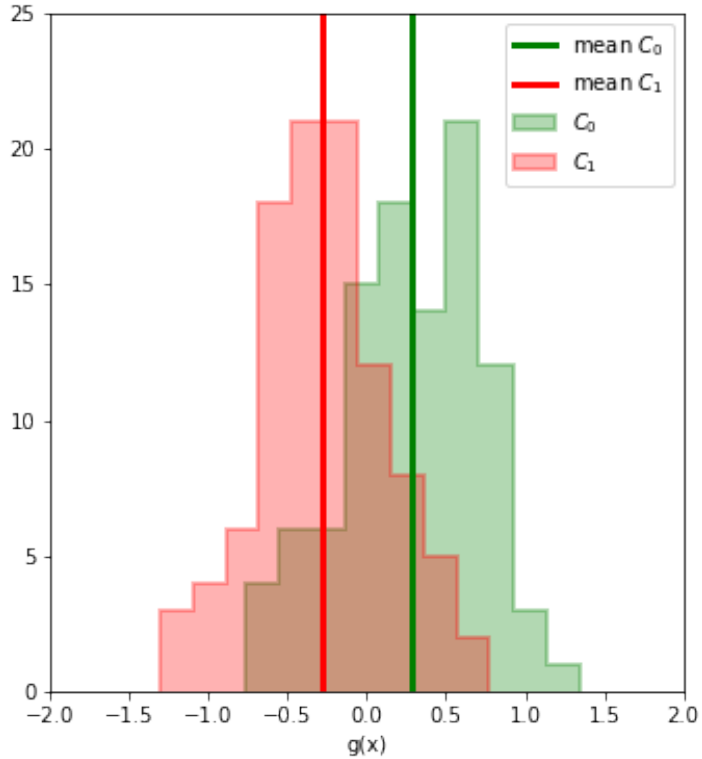
Find the **linear combination** of the input variables that **maximizes the separability** of the two classes

We need to *put into numbers* the criterion of **separability** and then maximize it.

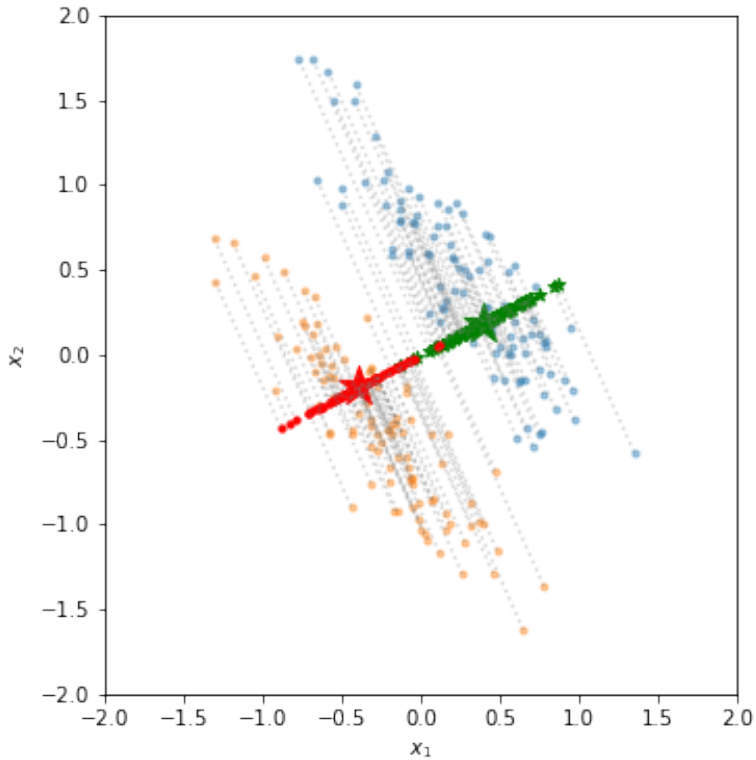


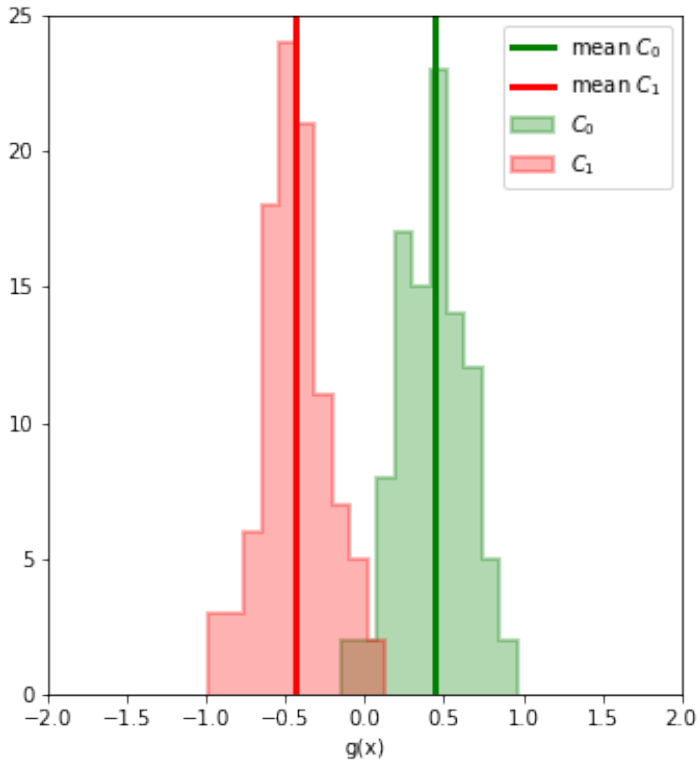
Separability in the output of the discriminant function



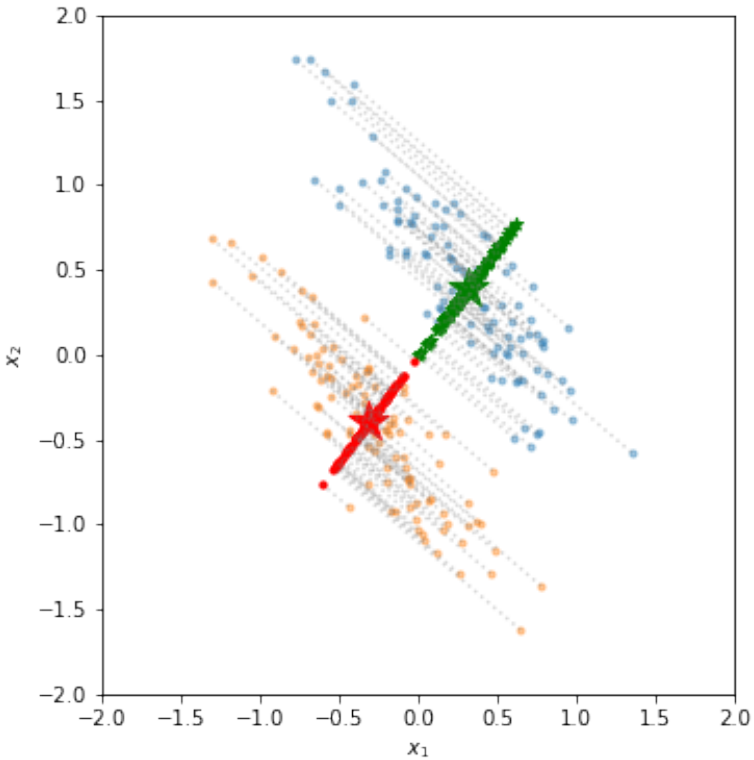


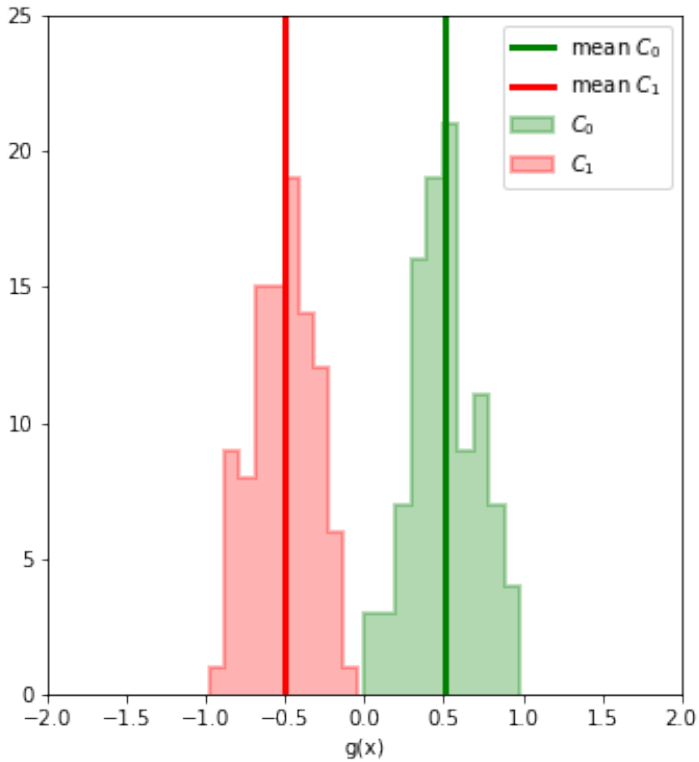
Separability in the output of the discriminant function



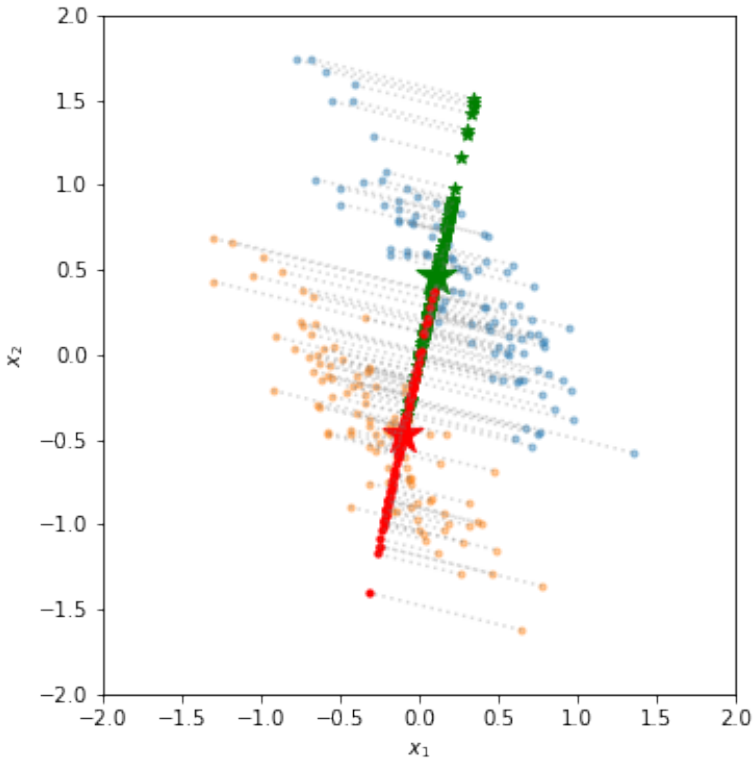


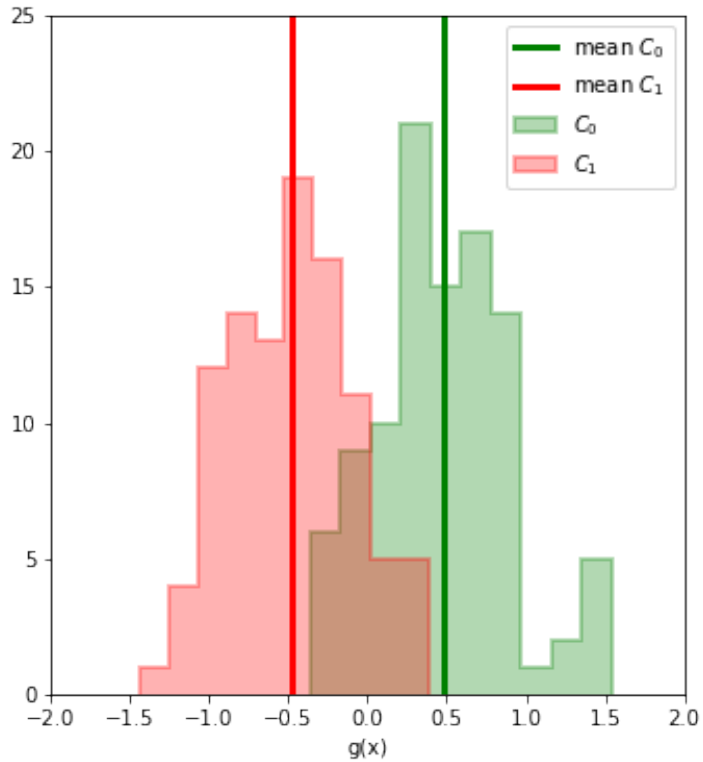
Separability in the output of the discriminant function



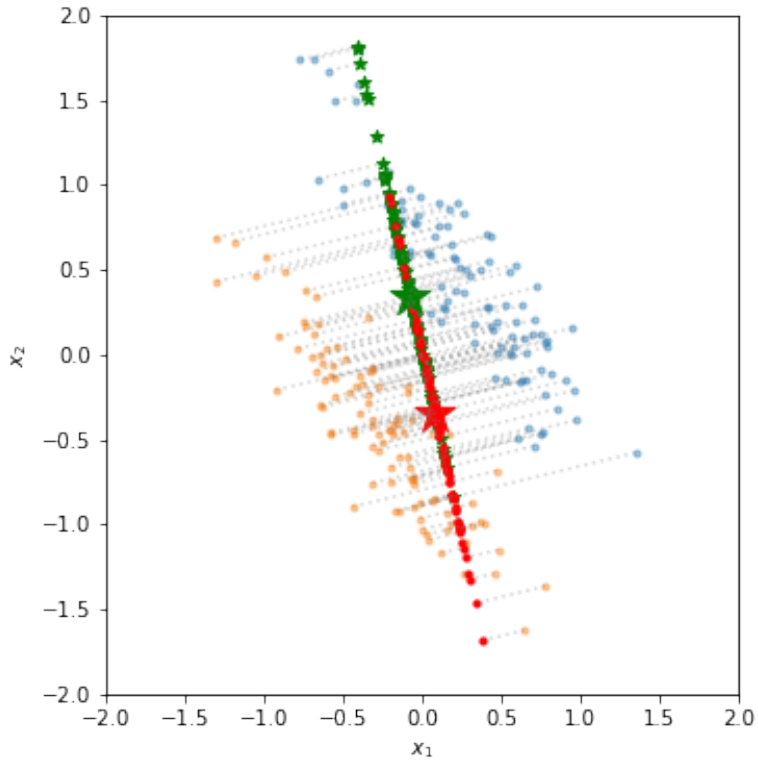


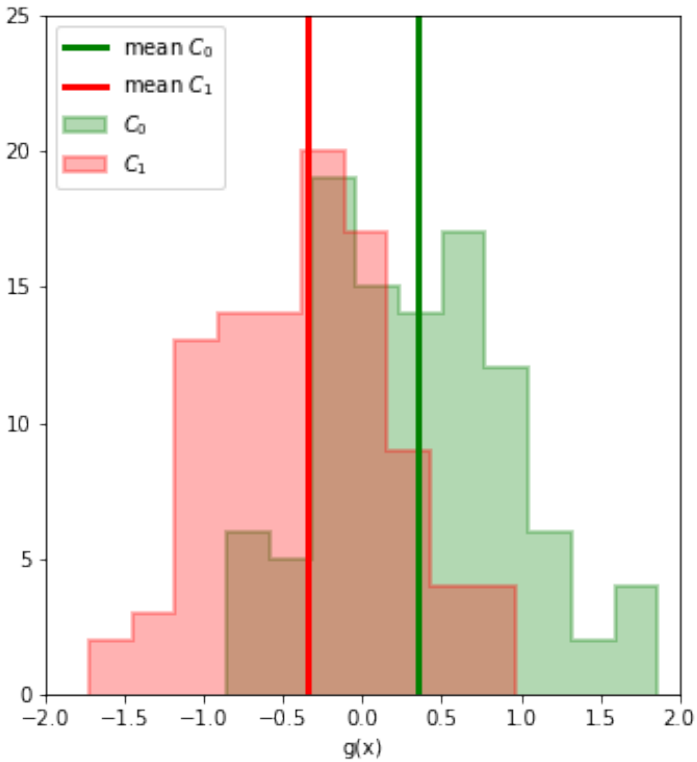
Separability in the output of the discriminant function



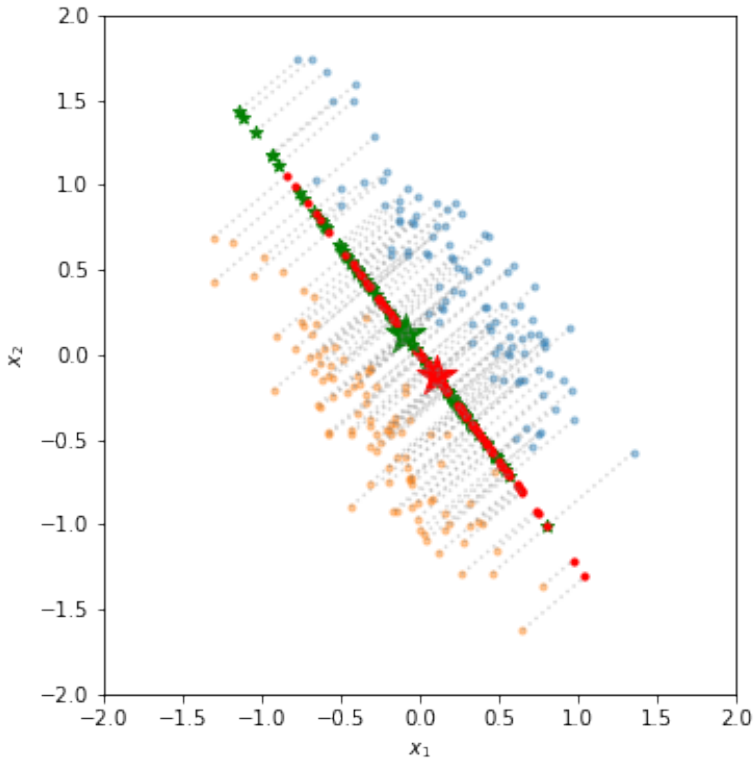


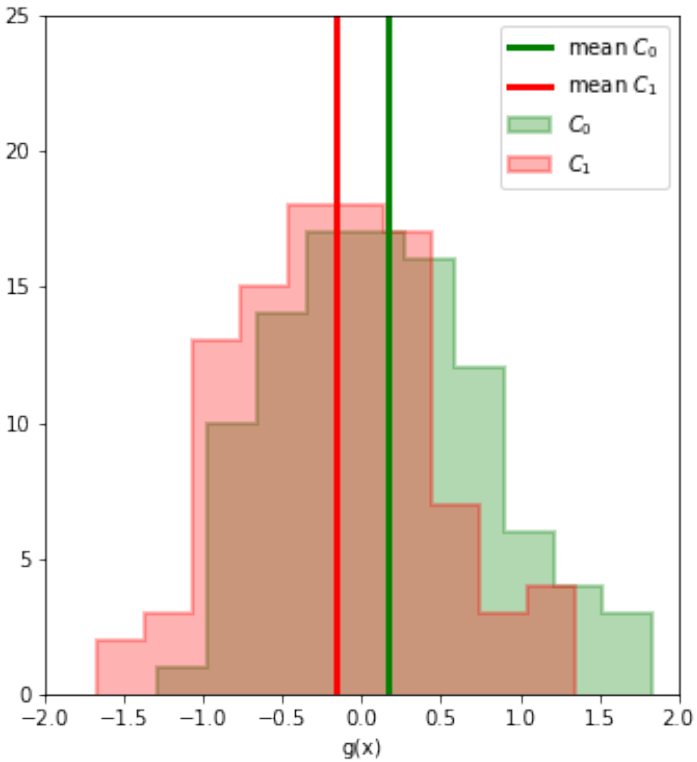
Separability in the output of the discriminant function



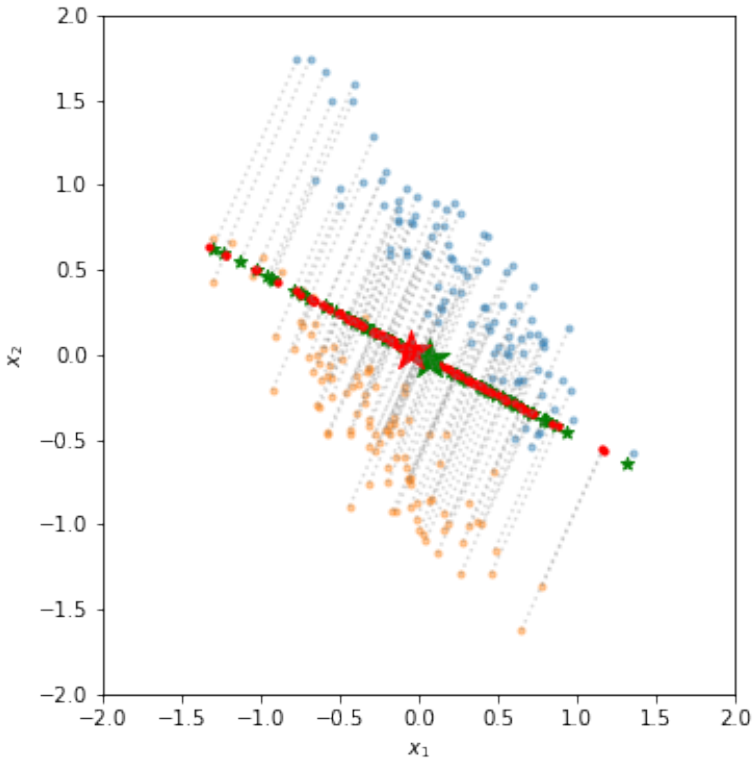


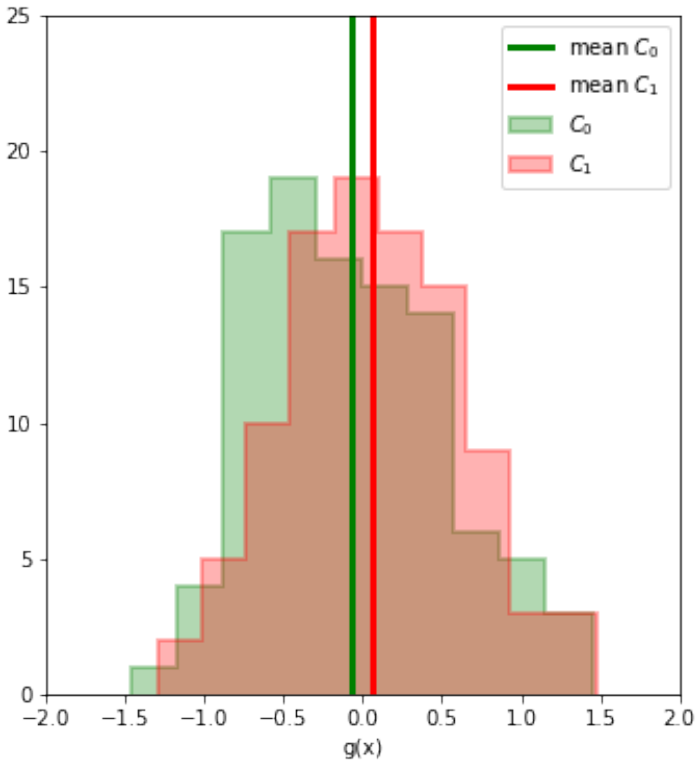
Separability in the output of the discriminant function



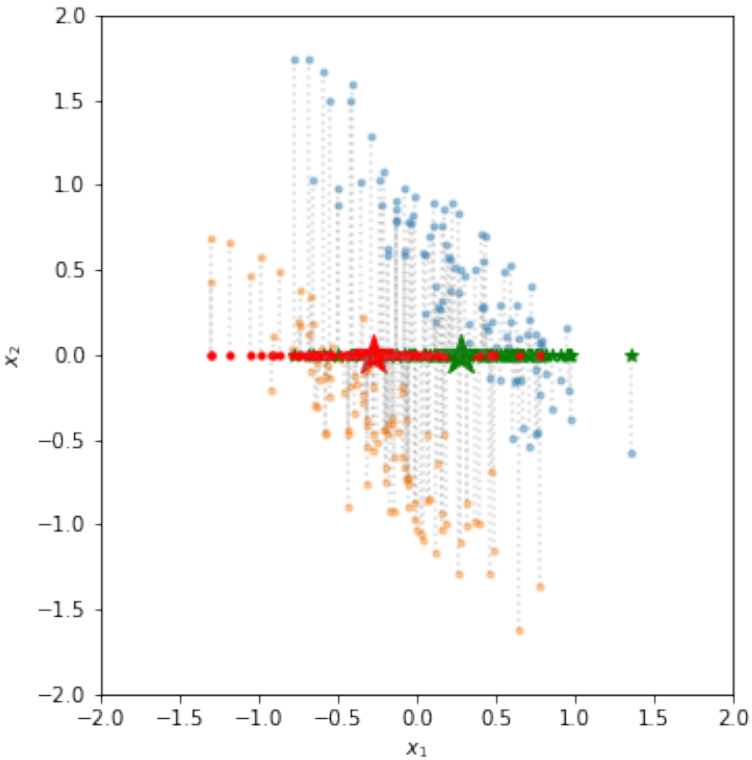


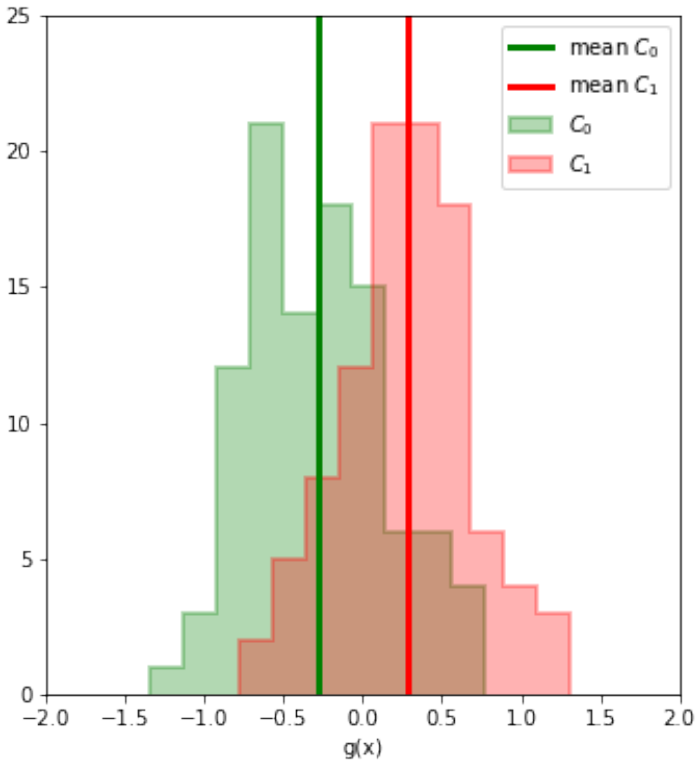
Separability in the output of the discriminant function





Separability in the output of the discriminant function





Maths for the criterion for separability

- **Separate the means** of the scalar datasets that result after applying a linear discriminant function to the training observations of each class.
- And **minimize the variances** of the 1D variables resulting from projecting the original training data with the discriminant function

Separate the means will be a better proxy for class separability if the classes are somewhat **compact** (all observations are concentrated around their corresponding means)

Optimization

$$\max_{\mathbf{w}} J(\mathbf{w}) = \frac{(m_1 - m_{-1})^2}{s_{-1}^2 + s_1^2}$$

$$\text{subject to } \mathbf{w}^\top \mathbf{w} = 1$$

where

$$s_{-1}^2 = \sum_{i \in C_{-1}} (g(\mathbf{x}_i) - m_{-1})^2, \quad s_1^2 = \sum_{i \in C_1} (g(\mathbf{x}_i) - m_1)^2$$

Fisher's Linear Discriminant Optimization

$$\max_{\mathbf{w}} J(\mathbf{w}) = \frac{(m_1 - m_{-1})^2}{s_{-1}^2 + s_1^2} =$$

$$= \frac{\mathbf{w}^\top (\mathbf{m}_1 - \mathbf{m}_{-1})(\mathbf{m}_1 - \mathbf{m}_{-1})^\top \mathbf{w}}{\sum_{i \in C_{-1}} \mathbf{w}^\top (\mathbf{x}_i - \mathbf{m}_{-1})(\mathbf{x}_i - \mathbf{m}_{-1})^\top \mathbf{w} + \sum_{i \in C_1} \mathbf{w}^\top (\mathbf{x}_i - \mathbf{m}_1)(\mathbf{x}_i - \mathbf{m}_1)^\top \mathbf{w}}$$

$$= \max_{\mathbf{w}} \frac{\mathbf{w}^\top (\mathbf{m}_1 - \mathbf{m}_{-1})(\mathbf{m}_1 - \mathbf{m}_{-1})^\top \mathbf{w}}{\mathbf{w}^\top \left(\sum_{i \in C_{-1}} (\mathbf{x}_i - \mathbf{m}_{-1})(\mathbf{x}_i - \mathbf{m}_{-1})^\top + \sum_{i \in C_1} (\mathbf{x}_i - \mathbf{m}_1)(\mathbf{x}_i - \mathbf{m}_1)^\top \right) \mathbf{w}}$$

where

$$\mathbf{m}_{-1} = \frac{1}{N_{-1}} \sum_{y_i = -1} \mathbf{x}_i, \quad m_{-1} = \mathbf{w}^\top \mathbf{m}_{-1} \qquad \mathbf{m}_1 = \frac{1}{N_1} \sum_{y_i = +1} \mathbf{x}_i, \quad m_1 = \mathbf{w}^\top \mathbf{m}_1$$

Between class and Intra-class covariance matrices

The numerator can be written as

$$\mathbf{w}^\top(\mathbf{m}_1 - \mathbf{m}_{-1})(\mathbf{m}_1 - \mathbf{m}_{-1})^\top\mathbf{w} = \mathbf{w}^\top S_B\mathbf{w}$$

where $S_B = (\mathbf{m}_1 - \mathbf{m}_{-1})(\mathbf{m}_1 - \mathbf{m}_{-1})^\top$ is the **between class covariance matrix**

The denominator can also be written in a more compact way:

$$\mathbf{w}^\top\left(\sum_{i\in C_{-1}}(\mathbf{x}_i - \mathbf{m}_{-1})(\mathbf{x}_i - \mathbf{m}_{-1})^\top + \sum_{i\in C_1}(\mathbf{x}_i - \mathbf{m}_1)(\mathbf{x}_i - \mathbf{m}_1)^\top\right)\mathbf{w} = \mathbf{w}^\top S_W\mathbf{w}$$

where

$$S_W = \sum_{i\in C_{-1}}(\mathbf{x}_i - \mathbf{m}_{-1})(\mathbf{x}_i - \mathbf{m}_{-1})^\top + \sum_{i\in C_1}(\mathbf{x}_i - \mathbf{m}_1)(\mathbf{x}_i - \mathbf{m}_1)^\top$$

is the **intra-class covariance matrix**.

Fisher's Linear Discriminant solution

$$\max_{\mathbf{w}} J(\mathbf{w}) = \frac{\mathbf{w}^\top S_B \mathbf{w}}{\mathbf{w}^\top S_W \mathbf{w}}$$

$$\text{subject to } \mathbf{w}^\top \mathbf{w} = 1$$

Gradientes wrt $\mathbf{w}$ equal to zero

$$\frac{2S_B\mathbf{w}(\mathbf{w}^\top S_W \mathbf{w}) - 2S_W\mathbf{w}(\mathbf{w}^\top S_B \mathbf{w})}{(\mathbf{w}^\top S_W \mathbf{w})^2} = 0$$

$$\Rightarrow S_B\mathbf{w}(\mathbf{w}^\top S_W \mathbf{w}) = S_W\mathbf{w}(\mathbf{w}^\top S_B \mathbf{w}) \Rightarrow S_B\mathbf{w} = \frac{(\mathbf{w}^\top S_B \mathbf{w})}{(\mathbf{w}^\top S_W \mathbf{w})} S_W\mathbf{w}$$

Notice $\frac{(\mathbf{w}^\top S_B \mathbf{w})}{(\mathbf{w}^\top S_W \mathbf{w})}$ is a scalar. Since we are actually seeking the direction of $\mathbf{w}$, not its size, we can replace this scalar by a constant c

$$\Rightarrow S_B\mathbf{w}(\mathbf{w}^\top S_W \mathbf{w}) = S_W\mathbf{w}(\mathbf{w}^\top S_B \mathbf{w}) \Rightarrow S_B\mathbf{w} = cS_W\mathbf{w}$$

Fisher's Linear Discriminant solution

Multiply both sides by S_W^{-1}

$$S_W^{-1} S_B \mathbf{w} = c\mathbf{w}$$

From the definition of S_B:

$$S_B \mathbf{w} = (\mathbf{m}_1 - \mathbf{m}_{-1})(\mathbf{m}_1 - \mathbf{m}_{-1})^\top \mathbf{w} = \beta(\mathbf{m}_1 - \mathbf{m}_{-1})$$

$S_B\mathbf{w}$ goes along the direction given by $(\mathbf{m}_1 - \mathbf{m}_{-1})$ with length β. Since we just want the direction of $\mathbf{w}$, we don't need to compute the exact value of β. Using this result in the main equation:

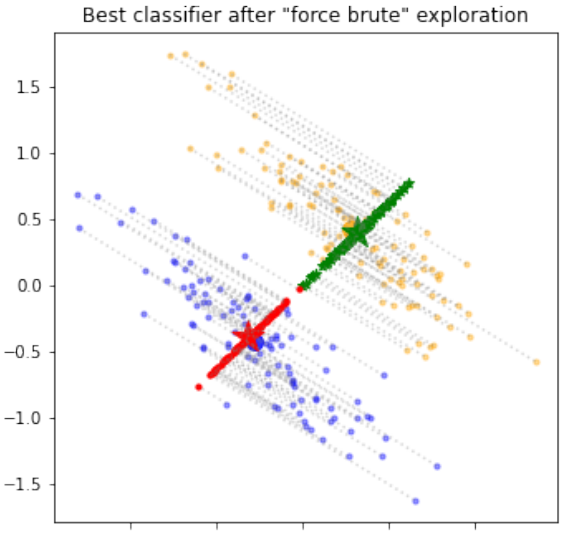
$$\beta' S_W^{-1}(\mathbf{m}_1 - \mathbf{m}_{-1}) = \mathbf{w}$$

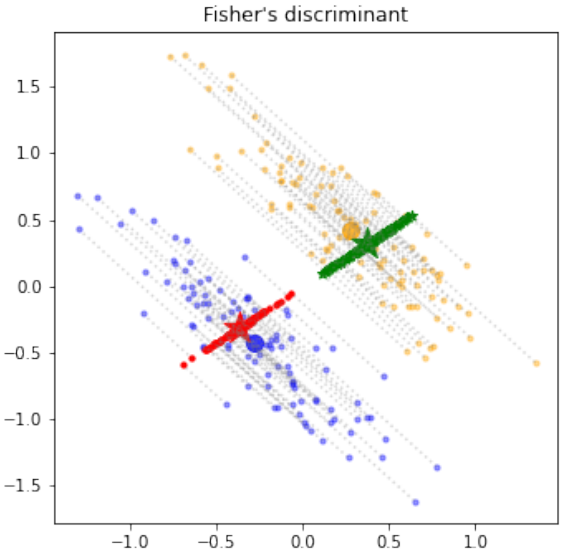
So $\mathbf{w}$ is a unit vector in the direction given by $S_W^{-1}(\mathbf{m}_1 - \mathbf{m}_{-1})$.

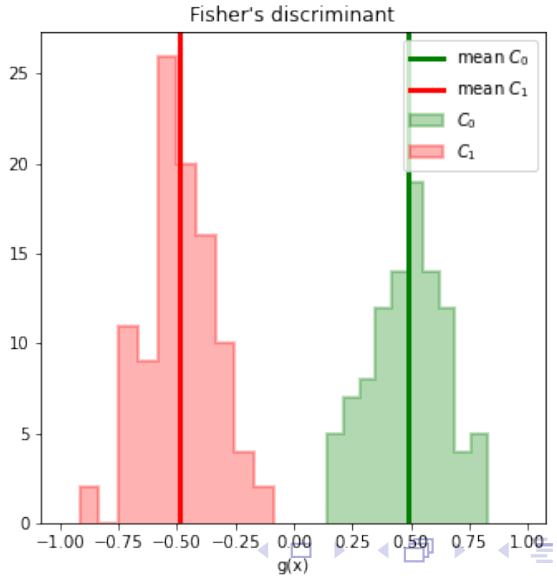
Fisher's discriminant

Vector $\mathbf{w}$ defines the direction of the **Fisher's discriminant**. This classifier must be completed with a threshold w_0 to compare the output of the discriminant function and make the decision

Fisher's discriminant vs. Brute Force







Index

Separating each mean from the overall mean

Separate m_{-1} from m_1 is equivalent to **simultaneously separate m_{-1} and m_1 from m**, where m is the **mean of the whole training set**.

$$(m_1 - m)^2 + (m_0 - m)^2 = m_1^2 - 2m_1 m + m^2 + m_0^2 - 2m_0 m + m^2$$
$$= m_1^2 + m_0^2 + 2m^2 - 2m_0 m - 2m_1 m$$

If we assume equiprobable classes, the overall mean is

$$m = \frac{1}{2}(m_1 + m_0)$$

Using this result in the main equation

$$(m_1 - m)^2 + (m_0 - m)^2 =$$

$$m_1^2 + m_0^2 + \frac{1}{2}(m_1^2 + m_0^2 + 2m_1 m_0) - m_0(m_1 + m_0) - m_1(m_1 + m_0) =$$

$$m_1^2 + m_0^2 + \frac{1}{2}(m_1^2 + m_0^2 + 2m_1 m_0) - 2m_0 m_1 - m_0^2 - m_1^2 =$$

$$= \frac{1}{2}m_1^2 + \frac{1}{2}m_0^2 - m_1 m_0 = \frac{1}{2}(m_1 - m_0)^2$$

Linear separability in more than 2 dimensions

- The Fisher's discriminant in a binary problem can be interpreted as a mapping of the input data into a 1 dimensional space (scalar variable) and then separate the means of the resulting classes in 1D.
- If we have $C > 2$ classes, we would need $C - 1$ dimensions to linearly separate the mean of each class from the overall mean
- Therefore a problem with $C > 2$ classes needs the observations to live in a space of at least $d = C - 1$ dimensions, in order to be able to map these observations into the $C - 1$ subspace in which the linear discrimination can take place
- **The number of linearly independent columns of the dataset imposes a limit in the number of classes that can be separated with a Multiclass Linear Discriminant**

Mapping into $C-1$ dimensions

Fisher's discriminant in a case with $C > 2$ classes must map the input data into a subspace of dimensionality $C-1$.
This mapping is carried out with $C-1$ unit vectors $\{\mathbf{w}_q\}_{q=1}^{C-1}$ that we need to find. These $\mathbf{w}_q$ are the rows of a $(C-1) \times d$ matrix W

$$\mathbf{z}_i = W\mathbf{x}_i = \begin{bmatrix} \mathbf{w}_1^\top \\ \mathbf{w}_2^\top \\ \vdots \\ \mathbf{w}_{C-1}^\top \end{bmatrix} \mathbf{x}_i$$

Therefore, the means of the overall dataset and of each mapped class are

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{z}_i \in \mathbb{R}^q$$

$$\boldsymbol{\mu}_1 = \frac{1}{N_1}\sum_{y_i=1} \mathbf{z}_i \in \mathbb{R}^q, \quad \boldsymbol{\mu}_2 = \frac{1}{N_2}\sum_{y_i=2} \mathbf{z}_i \in \mathbb{R}^q, \quad \ldots \quad \boldsymbol{\mu}_C = \frac{1}{N_C}\sum_{y_i=M} \mathbf{z}_i \in \mathbb{R}^q$$

where N_c is the number of observations of class C_c in the training set

Optimization Problem in the mapped space

- Between-class covariance

$$s_B = \sum_{c=1}^{C} N_c(\boldsymbol{\mu}_c - \boldsymbol{\mu})(\boldsymbol{\mu}_c - \boldsymbol{\mu})^\top$$

- Intra-class covariance

$$s_W = s_1 + s_2 + \cdots + s_C = \sum_{c=1}^{C} \sum_{y_i=c} (\mathbf{z}_i - \boldsymbol{\mu}_c)(\mathbf{z}_i - \boldsymbol{\mu}_c)^\top$$

- Functional

$$J(W) = \max_W \text{Trace}\{s_W^{-1} s_B\}$$

Between-class matrix in the input space

$$s_B = \sum_{c=1}^{C} N_c(\boldsymbol{\mu}_c - \boldsymbol{\mu})(\boldsymbol{\mu}_c - \boldsymbol{\mu})^\top$$

$$= \sum_{c=1}^{C} N_c W(\mathbf{m}_c - \mathbf{m})(\mathbf{m}_c - \mathbf{m})^\top W^\top$$

$$= W\left(\sum_{c=1}^{C} N_c(\mathbf{m}_c - \mathbf{m})(\mathbf{m}_c - \mathbf{m})^\top\right) W^\top =$$

$$W S_B W^\top$$

where

$$S_B = \sum_{c=1}^{C} N_c(\mathbf{m}_c - \mathbf{m})(\mathbf{m}_c - \mathbf{m})^\top$$

Intra-class matrix in the input space

$$s_W = \sum_{c=1}^{C} \sum_{y_i=c} (\mathbf{z}_i - \boldsymbol{\mu}_c)(\mathbf{z}_i - \boldsymbol{\mu}_c)^\top$$

$$= \sum_{c=1}^{C} \sum_{y_i=c} W(\mathbf{x}_i - \mathbf{m}_c)(\mathbf{x}_i - \mathbf{m}_c)^\top W^\top$$

$$= W \left(\sum_{c=1}^{C} \sum_{y_i=c} (\mathbf{x}_i - \mathbf{m}_c)(\mathbf{x}_i - \mathbf{m}_c)^\top \right) W^\top =$$

$$= W S_W W^\top$$

where

$$S_W = \sum_{c=1}^{C} \sum_{y_i=c} (\mathbf{x}_i - \mathbf{m}_c)(\mathbf{x}_i - \mathbf{m}_c)^\top$$

Optimization Problem in input space

$$J(W) = \max_W \text{Trace}\{s_W^{-1} s_B\}$$

Introducing the definitions of the between-class and intra-class covariances in terms of the input space data:

$$s_B = W S_B W^\top$$

$$s_W = W S_W W^\top$$

yields

$$J(W) = \max_W \text{Trace}\{(W S_W W^\top)^{-1} W S_B W^\top\}$$

This optimization turns out into computing the **eigenvalues and eigenvectors** of matrix $S_W^{-1} S_B$. Precisely W is formed with the $C-1$ **eigenvectors** of $S_W^{-1} S_B$ with **largest eigenvalues**
Since S_B has a rank less or equal than $C-1$, $S_W^{-1} S_B$ will have maximum $C-1$ different eigenvalues.

Index

Index

Approximate the posterior with a linear function

The critical pdf in a binary classification problem is the posterior of the class $P(y = 1|\mathbf{x})$. We can use this posterior to build the following discriminant function

Discriminant Function based on the posterior

$$g(\mathbf{x}) = P(y = +1|\mathbf{x}) \underset{D=0}{\overset{D=1}{\gtrless}} \frac{1}{2}$$

Logistic Regression proposes to **approximate the posterior** with the composition of a **linear regressor** and a **sigmoid**

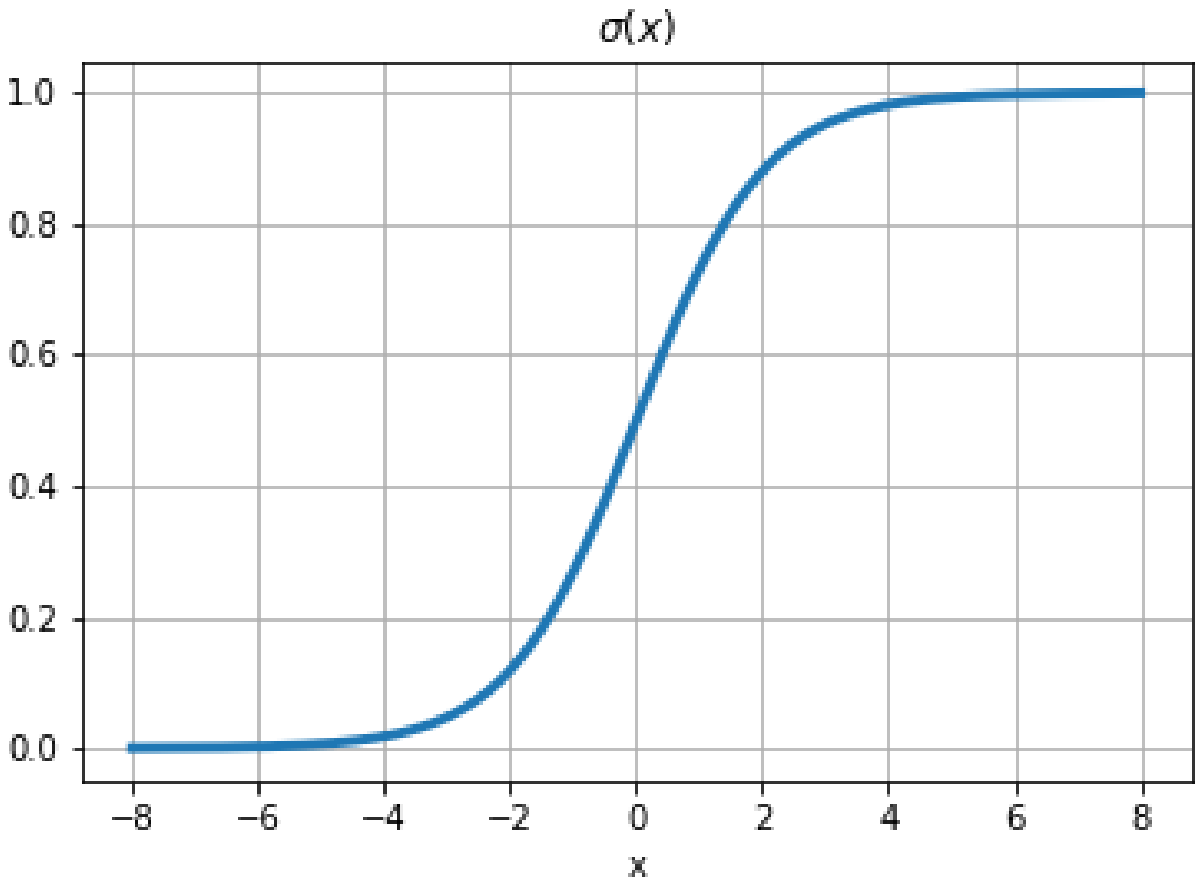
Logistic Regression

$$g(\mathbf{x}) = P(y = +1|\mathbf{x}) = \sigma(w_0 + \mathbf{w}^\top \mathbf{x}) = \sigma(\mathbf{w}_e^\top \mathbf{x}_e)$$

Sigmoid functions

Sigmoid

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$



Derivatives of sigmoids

$$\frac{d}{dx}\sigma(x) = \frac{e^{-x}}{(1+e^{-x})^2} = \sigma(x)\frac{e^{-x}}{1+e^{-x}} = \sigma(x)(1-\sigma(x))$$

$$\frac{d}{dx}\log\sigma(x) = \frac{1}{\sigma(x)}\frac{d}{dx}\sigma(x) = (1-\sigma(x))$$

$$\frac{d}{dx}\log(1-\sigma(x)) = \frac{1}{1-\sigma(x)}\frac{d}{dx}(-\sigma(x)) = \frac{-\sigma(x)(1-\sigma(x))}{1-\sigma(x)} - \sigma(x)$$

Joint posterior of the training set

In the logistic regression framework we will use the following notation to relate classes and hypothesis:

$$y_i = \begin{cases} +1 & \text{if } \mathbf{x}_i \in H_1 \\ 0 & \text{if } \mathbf{x}_i \in H_0 \end{cases}$$

Now we introduce a vector $\mathbf{y}$ including all the true targets of the training set

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

The joint posterior of $\mathbf{y}$ will be

$$P(\mathbf{y}|\mathbf{w}) = \prod_{i=1}^{N} P(y_i|\mathbf{w}) = \prod_{i=1}^{N} \left(\frac{1}{1+\exp(-\mathbf{w}^\top \mathbf{x})} \right)^{y_i} \left(1 - \frac{1}{1+\exp(-\mathbf{w}^\top \mathbf{x})} \right)^{1-y_i}$$

Take logs in the joint posterior

$$J(\mathbf{w}) = \log P(\mathbf{y}|\mathbf{w}) = \sum_{i=1}^{N} y_i \log(\sigma(\mathbf{x}_i, \mathbf{w})) + (1-y_i) \log\left(1 - \sigma(\mathbf{x}_i, \mathbf{w})\right)$$

Index

Newton's Method

2nd order Taylor's expansion

$$J(\mathbf{w}) \approx J(\mathbf{w}_0) + (\mathbf{w} - \mathbf{w}_0)\nabla_{\mathbf{w}} J(\mathbf{w})|_{\mathbf{w}=\mathbf{w}_0} + \frac{1}{2}(\mathbf{w} - \mathbf{w}_0)^\top H(\mathbf{w}_0)(\mathbf{w} - \mathbf{w}_0)$$

In the minimum the gradient will be zero: $\nabla_{\mathbf{w}} J(\mathbf{w})|_{\mathbf{w}=\mathbf{w}^*} = \mathbf{0}$.
The gradient of the second order expansion is:

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \nabla_{\mathbf{w}} J(\mathbf{w})|_{\mathbf{w}=\mathbf{w}_0} + H(\mathbf{w}_0)(\mathbf{w} - \mathbf{w}_0)$$

Therefore if the minimum sits on $\mathbf{w} = \mathbf{w}^*$

$$\nabla_{\mathbf{w}} J(\mathbf{w})|_{\mathbf{w}=\mathbf{w}^*} = \nabla_{\mathbf{w}} J(\mathbf{w})|_{\mathbf{w}=\mathbf{w}_0} + H(\mathbf{w}_0)(\mathbf{w}^* - \mathbf{w}_0) \Rightarrow$$

$$\mathbf{w}^* = \mathbf{w}_0 - H(\mathbf{w}_0)^{-1}\nabla_{\mathbf{w}} J(\mathbf{w})|_{\mathbf{w}=\mathbf{w}_0}$$

Newton's Method in the logistic regression case

- Gradient

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \nabla_{\mathbf{w}} \left(\sum_{i=1}^{N} y_i \log(\sigma(\mathbf{x}_i, \mathbf{w})) + (1 - y_i) \log\left(1 - \sigma(\mathbf{x}_i, \mathbf{w})\right) \right)$$

$$= \sum_{i=1}^{N} \mathbf{x}_i (y_i - \sigma(\mathbf{x}_i, \mathbf{w})) = X^\top (\mathbf{y} - \boldsymbol{\sigma})$$

- Hessian

$$H(\mathbf{w}_0) = \nabla_{\mathbf{w}} \nabla_{\mathbf{w}} J(\mathbf{w}) = \nabla_{\mathbf{w}} \left(- \sum_{i=1}^{N} \sigma(\mathbf{x}_i, \mathbf{w}) \mathbf{x}_i \right)$$

$$= - \sum_{i=1}^{N} \nabla_{\mathbf{w}} \sigma(\mathbf{x}_i, \mathbf{w}) \mathbf{x}_i = - \sum_{i=1}^{N} \sigma(\mathbf{x}_i, \mathbf{w})(1 - \sigma(\mathbf{x}_i, \mathbf{w})) \mathbf{x}_i \mathbf{x}_i^\top = X^\top R X$$

where R is a diagonal matrix with elements $R_{ii} = \sigma(\mathbf{x}_i, \mathbf{w})(1 - \sigma(\mathbf{x}_i, \mathbf{w}))$

Newton's Method in the logistic regression case

- Recursive main equation, we can't solve in one step as $\sigma()$ depends on $\mathbf{w}$

$$\mathbf{w}^* = \mathbf{w}_0 - H(\mathbf{w}_0)^{-1} \nabla_{\mathbf{w}} J(\mathbf{w})|_{\mathbf{w}=\mathbf{w}_0}$$

$$\Rightarrow \mathbf{w}^* = \mathbf{w}_{\text{old}} - (X^\top R X)^{-1} X^\top (\mathbf{y} - \boldsymbol{\sigma})$$

$$= (X^\top R X)^{-1} \left(X^\top R X \mathbf{w}_{\text{old}} - X^\top (\mathbf{y} - \boldsymbol{\sigma}) \right)$$

$$= (X^\top R X)^{-1} X^\top \left(R X \mathbf{w}_{\text{old}} - (\mathbf{y} - \boldsymbol{\sigma}) \right)$$

$$= (X^\top R X)^{-1} X^\top R \left(X \mathbf{w}_{\text{old}} - R^{-1} (\mathbf{y} - \boldsymbol{\sigma}) \right)$$

$$= (X^\top R X)^{-1} X^\top R \mathbf{z}$$

where

$$\mathbf{z} = \left(X \mathbf{w}_{\text{old}} - R^{-1} (\mathbf{y} - \boldsymbol{\sigma}) \right)$$

Recursive solution: Iterative Recursive Weighted Least Squares

Due to the interdependence between $\sigma()$ and $\mathbf{w}$, we need to implement the following recursion

1. Random initial guess for $\mathbf{w}_{\text{old}}$
2. Compute $\boldsymbol{\sigma}$ using $\mathbf{w}_{\text{old}}$
3. Compute R using $\boldsymbol{\sigma}$
4. Compute $\mathbf{z} = \left(X\mathbf{w}_{\text{old}} - R^{-1}(\mathbf{y} - \boldsymbol{\sigma})\right)$
5. Compute $\mathbf{w}^* = (X^\top RX)^{-1}X^\top R\mathbf{z}$
6. Make $\mathbf{w}_{\text{old}} = \mathbf{w}^*$ and go to step 2 until convergence

Since the Hessian is positive semidefinite the convergence to a global optimum is guaranteed.

Index

Index

Cross Validation

- Method for estimating the **generalization capability** of any machine learning model
- Simulates the process of evaluating the model with a **separate set** of data not used for training.

Algorithm:

1. Split the training set in N subsets called folds
2. Loop for $n = 1, \ldots, N$
   1. Fit the model with a training set formed by the union of all the folds but n (notice your training set size is that of $N - 1$ folds)
   2. Evaluate the model with subset n and store the *score*
3. Use the **average** of th N scores as estimation to the score that you will obtain when you train the model with all the training data and use a real test set in the evaluation.

Grid Search Cross Validation

Commonly used method to obtain **good values for the hyperparmeters** of a model

- k in kNN
- Maximum number of leaf nodes or depth of the tree in decision trees
- Number of trees in a random forest

Algorithm:

1. Define a range of values you want to explore for each hyperparameter to be tuned
2. Construct a **grid** that spans all the possible combinations of hyperparameters. If for instance you have to explore 3 parameters a, b and c with ranges M_a, M_b and M_c the size of the gird will be $M_a \times M_b \times M_c$
3. Estimate the quality of each combination of hyperparameters running a **cross-validation** on each node of the grid
4. return the values for the hyperparameters that form the node of the grid that achieved the best cross validation score.

Index

True positive, False negative...

In a binary classification the outcome of the classifier can be also categorized into 4 groups:

True Positive Those observations whose true target is the positive class and the predicted class is also the positive one. TP: number of True Positives

False Positive Those observations whose true target is the negative class but the predicted class is the positive one. FP: number of False Positives

True Negative Those observations whose true target is the negative class and the predicted class is also the negative one. TN: number of True Negatives

False Negative Those observations whose true target is the positive class but the predicted class is the negative one. FN: number of False Negatives

		Predicted class	
		Negative	Positive
True class	Negative	TN	FP
	Positive	FN	TP

Relation with false alarms and detections

- Probability of False Alarm

$$P_{\text{FA}} \approx \frac{FP}{FP + TN}$$

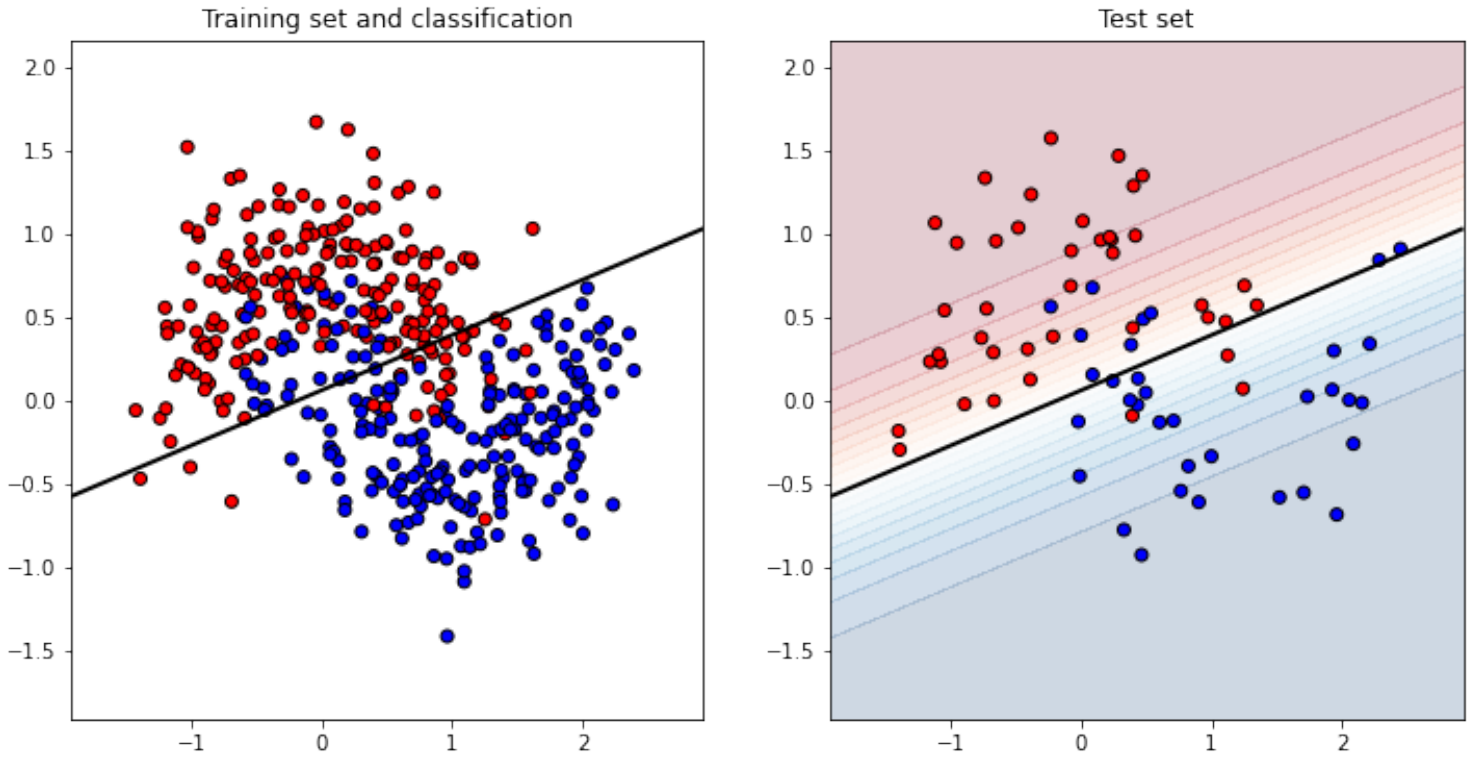
- Probability of Missing Targets

$$P_{\text{M}} \approx \frac{FN}{TP + FN}$$

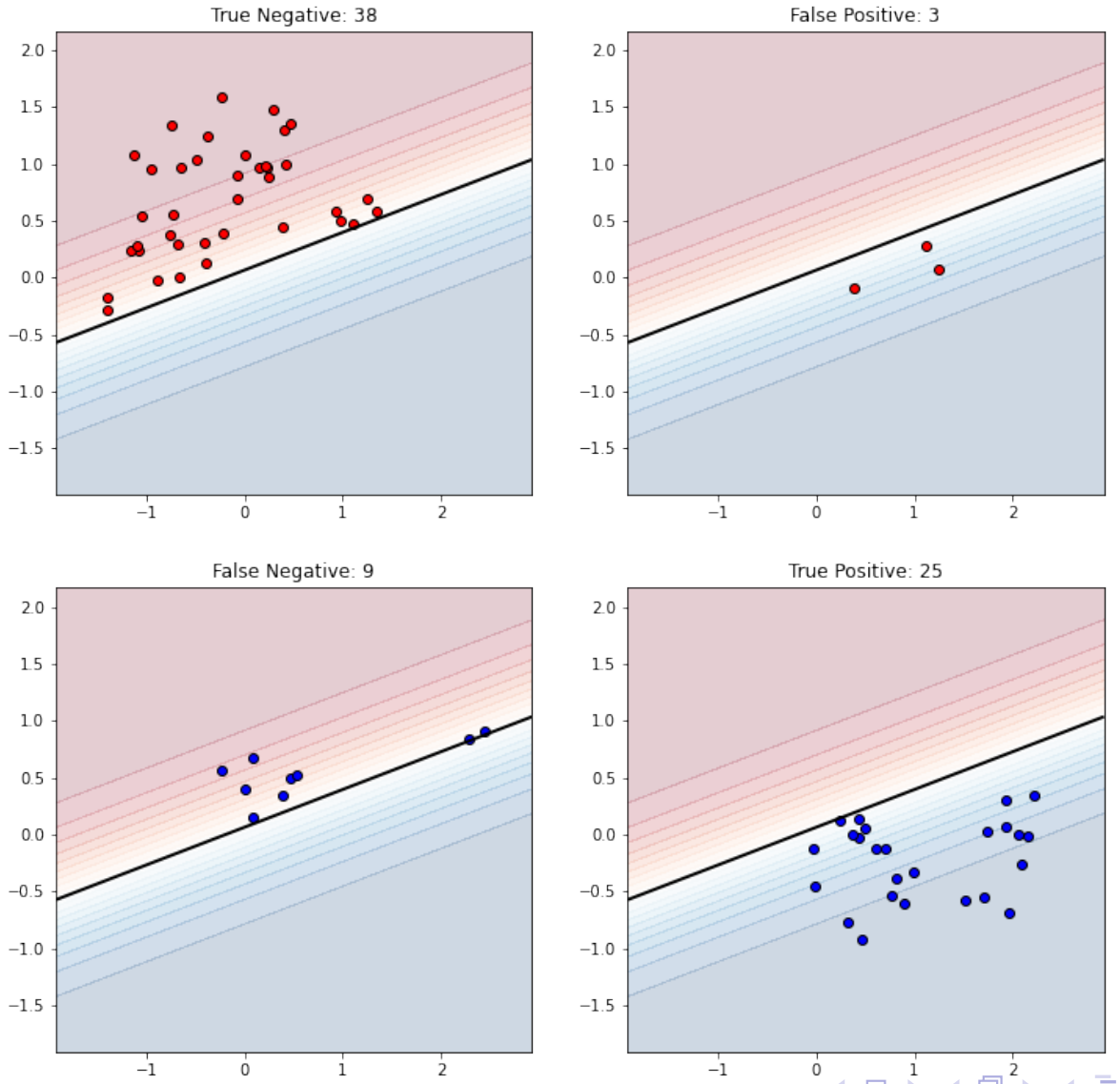
- Probability of Detection

$$P_{\text{D}} \approx \frac{TP}{TP + FN}$$

Example


Training set and classification
Test set
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1
0
1
2

Example


True Negative: 38
False Positive: 3
False Negative: 9
True Positive: 25

Confusion matrix

- Binary classification

$$\begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix}$$

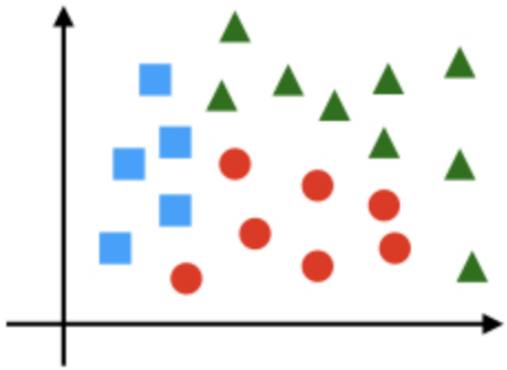
$$\underset{\text{unnormalized}}{\begin{bmatrix} 38 & 3 \\ 9 & 25 \end{bmatrix}} \quad \underset{\text{normalize='true'}}{\begin{bmatrix} 0.93 & 0.07 \\ 0.26 & 0.74 \end{bmatrix}} \quad \underset{\text{normalize='pred'}}{\begin{bmatrix} 0.81 & 0.11 \\ 0.19 & 0.89 \end{bmatrix}} \quad \underset{\text{normalize = 'all'}}{\begin{bmatrix} 0.51 & 0.04 \\ 0.12 & 0.33 \end{bmatrix}}$$

- Multiclass classification
  Element $M[i, j]$ contains the number of observations whose true class is C_i but were put into class C_j by the classifier

  Diagonal elements indicate the hits of the classifier, while off-diagonal elements indicate classification errors.

Index

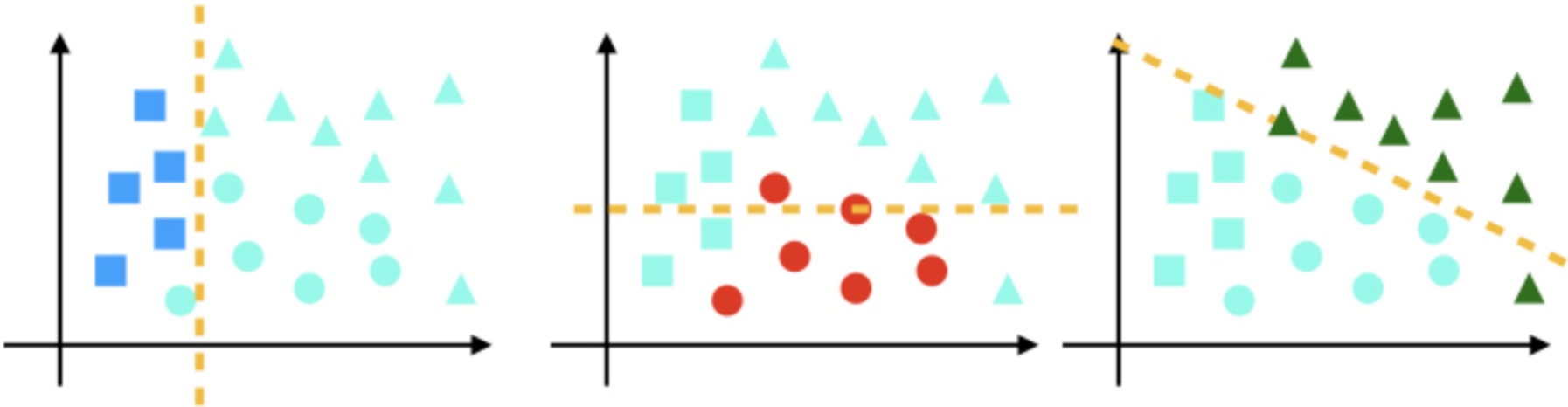
Linear classification in multiclass problems



The linear discriminant function is natural in binary classification. There are two common approaches to **tackle multiclass problems with binary classifiers**

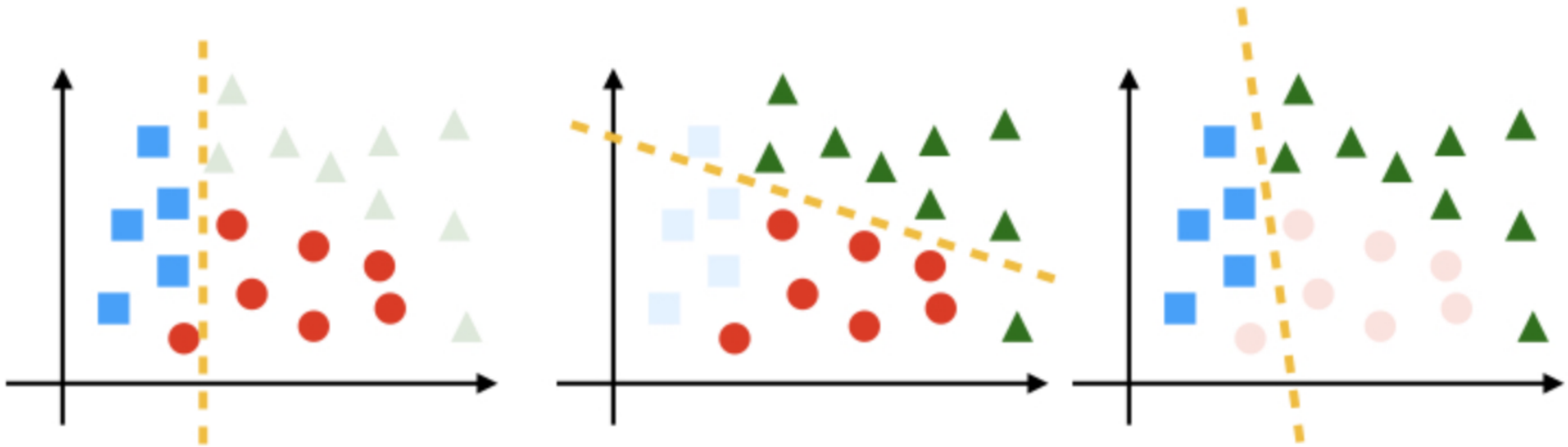
- **One vs All**: In a case with $C > 2$ classes, train C linear classifiers, each focused in separating one of the classes (positive class of the c classifier) from the rest (the training observations that belong to classes different from c form the negative class)
- **One vs One**: n a case with $C > 2$ classes, train $C(C-1)/2$ linear classifiers, one per every pair of classes.

One vs All Classification



Predict each test observation with the C classifiers and assign it to the class whose corresponding classifer achieves the larger value of the discriminant function.

One vs One Classification



Predict each test observation with all the classifiers and assign it to the class that achieves a larger number of votes among all the classifiers.

To solve ties we could also look at the values of the discriminant functions.

Index

Discriminant functions with nonlinear elements

An immediate way to introduce non-linear elements in the discriminant function is to **extend the input matrix with columns that incorporate non-linear functions of the original variables**

These new columns act in the discriminant function as if we had added more variables. The overall discriminant is a linear combination of all the columns. Since these new columns are non-linear functions of the original variables, it turns out that the discriminant function that results from the application of one of the linear methods discussed in this lecture in the extended data is a non-linear function of the original variables.

Example of nonlinear discriminant functions