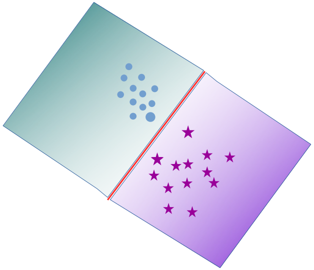
Introduction to AI
Logistic Regression



Rodrigo Cabral

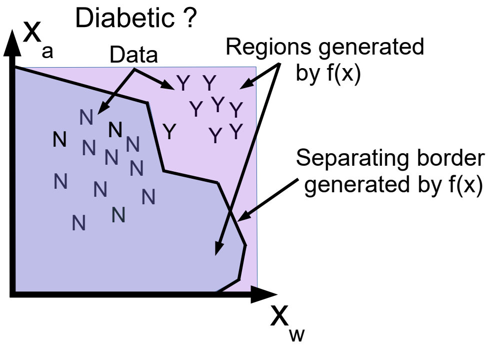
EUR DS4H-LIFE-SPECTRUM

cabral@unice.fr

Outline

Classification

- Supervised learning
- **y** can take only a finite number of predefined values (quantitative) or labels/classes (qualitative)

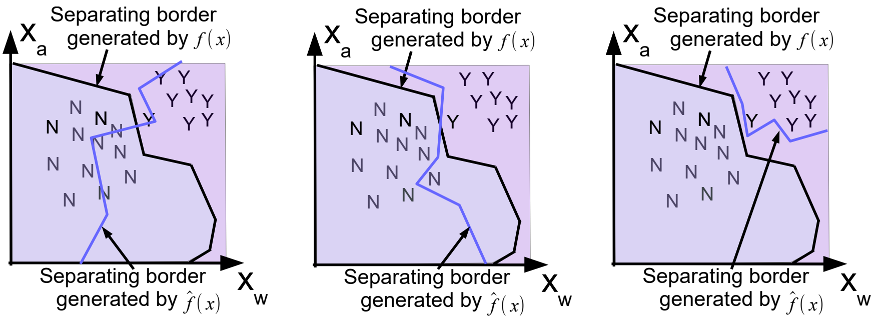

Diabetic ?
X_a
Data
Regions generated by f(x)
Separating border generated by f(x)
X_w
N
Y

- It is assumed that there is an underlying unknown function
$$f(\mathbf{x}) : \mathbb{R}^2 \Rightarrow \{Y, N\}$$
that separates the two classes (even unseen data) in the best possible manner.
- Function $f(\mathbf{x})$ separates the input space into regions.

Classification

Objective:

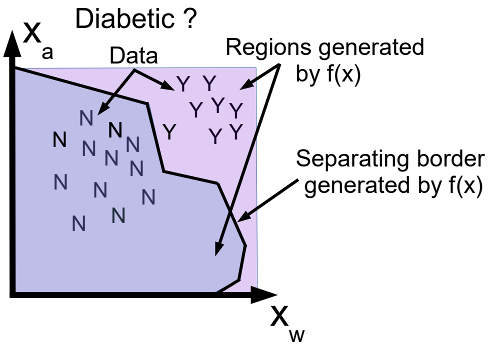
- Adjust a function $\hat{f}(\mathbf{x})$ to data, such that $\hat{f}(\mathbf{x})$ is close in some sense to the unknown $f(\mathbf{x})$.


Separating border generated by $f(x)$
Separating border generated by $\hat{f}(x)$
X_a
X_w
N
Y

- In practice, we are going to define a class of candidate functions for $\hat{f}(\mathbf{x})$ and then choose a function from that class such that $\hat{\mathbf{y}}$ are mostly close to $\mathbf{y}$ from the available dataset.

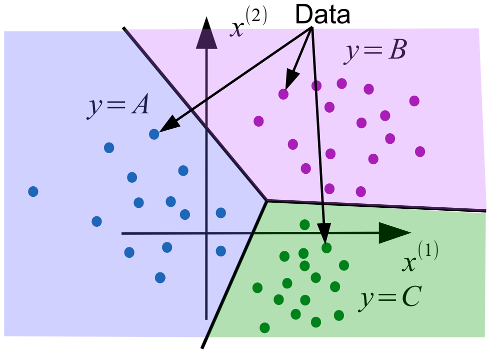
Binary classification

- When we have only **2 possible labels**, as in the example below, we say that we want to solve a **binary classification** problem.


Diabetic ?
X_a
Data
Regions generated by f(x)
Separating border generated by f(x)
X_w
Y Y Y Y Y Y Y Y
N N N N N N N N N N N N N N

Multi-class classification

- When we have **more than 2 possible labels**, as in the example below, we say that we want to solve a **multi-class classification** problem.



Linear classifier

- The class of candidate functions is often a parametric class, that is, the prediction function depends on the values of a parameter vector $\boldsymbol{\beta} \in \mathbb{R}^M$.

  Choosing a good candidate within a class of parametric functions $\hat{f}(\mathbf{x}; \boldsymbol{\beta})$ means choosing the parameter vector $\boldsymbol{\beta}$ that leads to good fit to data.
- One such class of parametrized classifiers are **linear classifiers**.

  In the binary classification case, a linear classifier decides on the predicted value based on a linear combination of the input features:

$$\hat{y} = g(\mathbf{x}\boldsymbol{\beta}) = g\left(\beta_0 + \sum_{i=1}^{M} \beta_i x^{(i)}\right)$$

  where $g(\cdot)$ is a decision function.

  Note that the weights of the combination are the parameters of the classifier.

Linear classifier

- The class of candidate functions is often a parametric class, that is, the prediction function depends on the values of a parameter vector $\boldsymbol{\beta} \in \mathbb{R}^M$.

  Choosing a good candidate within a class of parametric functions $\hat{f}(\mathbf{x}; \boldsymbol{\beta})$ means choosing the parameter vector $\boldsymbol{\beta}$ that leads to good fit to data.
- One such class of parametrized classifiers are **linear classifiers**.

  In the binary classification case, a linear classifier decides on the predicted value based on a linear combination of the input features:

$$\hat{y} = g(\mathbf{x}\boldsymbol{\beta}) = g\left(\beta_0 + \sum_{i=1}^{M} \beta_i x^{(i)}\right)$$

  where $g(\cdot)$ is a decision function.

  Note that the weights of the combination are the parameters of the classifier.

Linear classifier

- In the binary classification case, a linear classifier decides on the predicted value based on a linear combination of the input features:

$$\hat{y} = g(\mathbf{x}\boldsymbol{\beta})$$

- Often $g(\cdot)$ is simply a threshold function, for example, for two classes $y = 0$ or $y = 1$

$$\hat{y} = \begin{cases} 1, \text{if } \mathbf{x}\boldsymbol{\beta} \geq \tau, \\ 0, \text{if } \mathbf{x}\boldsymbol{\beta} < \tau, \end{cases}$$

  for some previously defined threshold value τ.

Linear classifier

- For 1 feature plus an implicit feature of 1, we have

$$\hat{y} = \begin{cases} 1, \text{if } \beta_0 + \beta_1 x \geq \tau, \\ 0, \text{if } \beta_0 + \beta_1 x < \tau, \end{cases}$$

which can be rewritten as (supposing $\beta_1 > 0$)

$$\hat{y} = \begin{cases} 1, \text{if } x \geq \frac{\tau - \beta_0}{\beta_1}, \\ 0, \text{if } x < \frac{\tau - \beta_0}{\beta_1}, \end{cases}$$

thus the predicted classes regions correspond to 2 complementary halfspaces of $\mathbb{R}$.

Linear classifier

- For 2 features plus an implicit feature of 1, we have

$$\hat{y} = \begin{cases} 1, \text{if } \beta_0 + \beta_1 x^{(1)} + \beta_2 x^{(2)} \geq \tau, \\ 0, \text{if } \beta_0 + \beta_1 x^{(1)} + \beta_2 x^{(2)} < \tau, \end{cases}$$

  which can be rewritten as (supposing $\beta_2 > 0$)

$$\hat{y} = \begin{cases} 1, \text{if } x^{(2)} \geq \left(\frac{\beta_1}{\beta_2}\right) x^{(1)} + \left(\frac{\tau-\beta_0}{\beta_2}\right), \\ 0, \text{if } x^{(2)} < \left(\frac{\beta_1}{\beta_2}\right) x^{(1)} + \left(\frac{\tau-\beta_0}{\beta_2}\right), \end{cases}$$

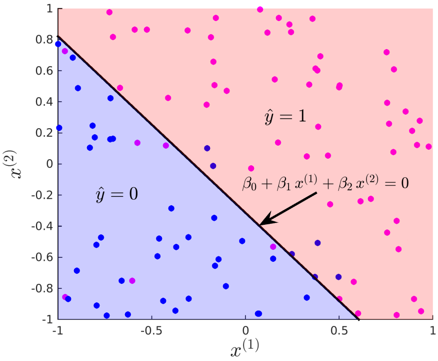
  thus the predicted classes regions correspond to 2 complementary halfspaces of $\mathbb{R}^2$.

- The border between the 2 classes is a line

$$x^{(2)} = \left(\frac{\beta_1}{\beta_2}\right) x^{(1)} + \left(\frac{\tau - \beta_0}{\beta_2}\right).$$

Linear classifier

Example in 2D with $\tau = 0$:



Linear classifier

- For 3 features the predicted classes regions correspond to 2 complementary halfspaces of $\mathbb{R}^3$.
- The border between the 2 classes is a plane.
- For N features the predicted classes regions correspond to 2 complementary halfspaces of $\mathbb{R}^N$.
- The border between the 2 classes is a hyperplane.
- Adjusting a linear classifier is equivalent to choosing a hyperplane that separates adequately available data.

Linear classifier

- For 3 features the predicted classes regions correspond to 2 complementary halfspaces of $\mathbb{R}^3$.
- The border between the 2 classes is a plane.
- For N features the predicted classes regions correspond to 2 complementary halfspaces of $\mathbb{R}^N$.
- The border between the 2 classes is a hyperplane.
- Adjusting a linear classifier is equivalent to choosing a hyperplane that separates adequately available data.

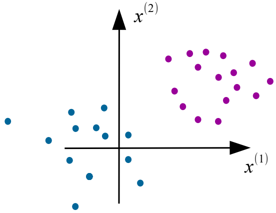
Linear classifier

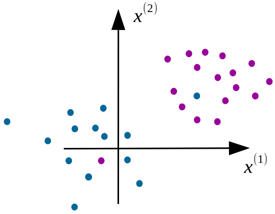
- For 3 features the predicted classes regions correspond to 2 complementary halfspaces of $\mathbb{R}^3$.
- The border between the 2 classes is a plane.
- For N features the predicted classes regions correspond to 2 complementary halfspaces of $\mathbb{R}^N$.
- The border between the 2 classes is a hyperplane.
- Adjusting a linear classifier is equivalent to choosing a hyperplane that separates adequately available data.

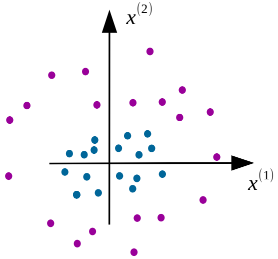
Linear classifier

- We say that the classes of a dataset are **linearly separable** when a linear classifier can be found such that there are no prediction errors.

Which of these datasets are linearly separable?



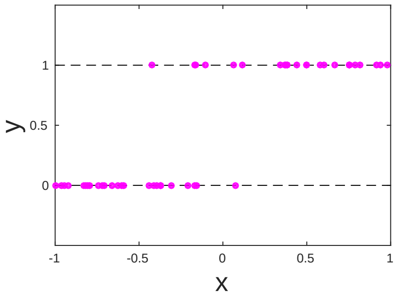




Binary classification

Scatter plot for binary classification with 1 feature

- Assume one input feature **x** with *N* observations, one output feature **y** with two possible labels.
- Without loss of generality we can assume $y_i \in \{0\,,\,1\}$.
- Scatter plot looks like this

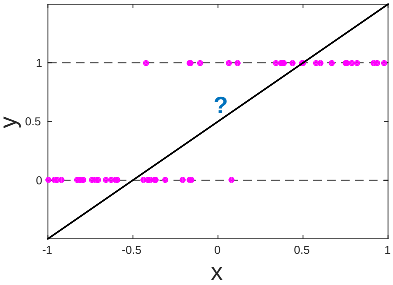


- Can we use a linear regression prediction model?

Binary classification: approximation with regression

Can we use linear regression to approximate the labels?

$$\hat{y} = \beta_0 + \beta_1 x$$



- Large errors for large absolute values of x.
- Can we transform the linear regression output to have more adapted predictions?

  $\Longrightarrow$ Answer: use an increasing smooth function $f(z)$ such that

$$\lim_{z \to -\infty} f(z) = 0$$
$$\lim_{z \to +\infty} f(z) = 1$$

Binary classification: approximation with regression

Sigmoidal functions

- Use an increasing smooth function $f(z)$ such that
$$\lim_{z\to-\infty} f(z) = 0$$
$$\lim_{z\to+\infty} f(z) = 1$$
- Sigmoidal functions, *i.e.* S-shaped functions, can be used.
  $\Longrightarrow$ Smooth versions of the *sign* function.
- Examples:
  - Gompertz: $f(z) = \exp\left[-\exp(-z)\right]$
  - Hyperbolic tangent with offset: $f(z) = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{z}{2}\right)$
  - Gaussian cumulative distribution: $f(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} \exp\left(-\frac{1}{2}t^2\right) \mathrm{d}t$
  - **Logistic function**: $f(z) = \frac{1}{1+\exp(-z)}$

Logistic regression

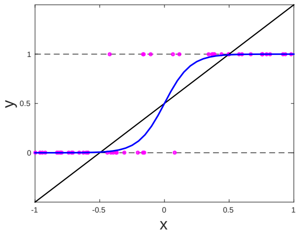
Logistic function

- In practice, the most used function is the logistic function.
- Applying it to the output of the linear regression model we have

$$f_{\boldsymbol{\beta}}(x) = \frac{1}{1 + \exp\left[-\left(\beta_0 + \beta_1 x\right)\right]} \tag{1}$$

- This gives a smooth transition from 0 to 1, without large errors for large absolute values of **x**.
- Interpretation as a probability $\mathbb{P}(y = 1|x) = f_{\boldsymbol{\beta}}(x)$

Logistic regression model in blue below.

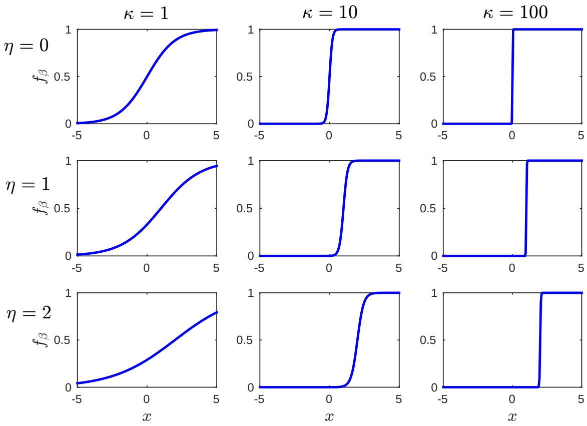


Logistic regression

Logistic function - effect of the parameters

- Rewriting the function with a slightly different parametrization

$$f_{\boldsymbol{\beta}}(x) = \frac{1}{1 + \exp\left[-\frac{\kappa}{\sqrt{1+\eta^2}}(x+\eta)\right]}$$



Binary classification

How do we learn the model parameters?

- Can we use the least squares approach?
- For N observations, we would solve the following optimization problem to retrieve $\boldsymbol{\beta}$:

$$\text{minimize} \qquad J(\boldsymbol{\beta}) = \sum_{i=1}^{N} j(y_i, f_{\boldsymbol{\beta}}(x_i))$$

$$\text{with respect to} \qquad \boldsymbol{\beta}$$

$$\text{where} \qquad j(y_i, f_{\boldsymbol{\beta}}(x_i)) = \varepsilon_i^2 = (y_i - f_{\boldsymbol{\beta}}(x_i))^2$$

Binary classification

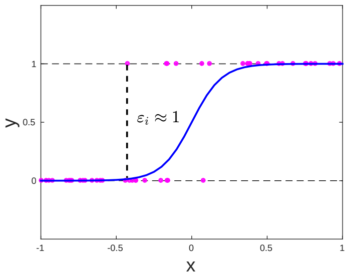
How do we learn the model parameters?

- Can we use the least squares approach?
- For N observations, we would solve the following optimization problem to retrieve $\boldsymbol{\beta}$:

$$\text{minimize} \qquad J(\boldsymbol{\beta}) = \sum_{i=1}^{N} j(y_i, f_{\boldsymbol{\beta}}(x_i))$$

$$\text{with respect to} \qquad \boldsymbol{\beta}$$

$$\text{where} \qquad j(y_i, f_{\boldsymbol{\beta}}(x_i)) = \varepsilon_i^2 = (y_i - f_{\boldsymbol{\beta}}(x_i))^2$$



- Issue: $j(y_i, f_{\boldsymbol{\beta}}(x_i)) \leq 1$
- The cost term should grow unbounded when prediction is completely false.

Logistic regression

How do we learn the model parameters?

- Which cost function should be used?
- Cost function should grow unbounded when prediction is completely false.

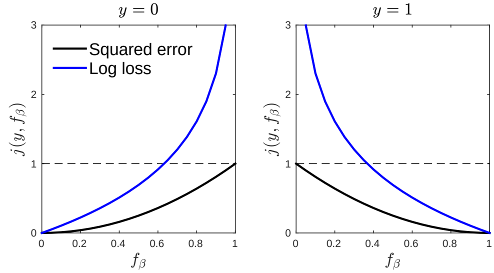
Logistic regression

How do we learn the model parameters?

- Which cost function should be used?
- Cost function should grow unbounded when prediction is completely false.
- Commonly used cost function is the following:

$$j(y, f_{\boldsymbol{\beta}}(x)) = -y \log(f_{\boldsymbol{\beta}}(x)) - (1 - y) \log(1 - f_{\boldsymbol{\beta}}(x)) \tag{2}$$

which is called the **log-loss** function.



Logistic regression

How do we learn the model parameters?

- Optimization problem corresponding to logistic regression parameter learning:

minimize $$J(\boldsymbol{\beta}) = \sum_{i=1}^{N} j(y_i, f_{\boldsymbol{\beta}}(x_i))$$

with respect to $$\boldsymbol{\beta}$$

where $$j(y_i, f_{\boldsymbol{\beta}}(x_i)) = -y_i \log(f_{\boldsymbol{\beta}}(x_i)) - (1 - y_i) \log(1 - f_{\boldsymbol{\beta}}(x_i))$$

with $$f_{\boldsymbol{\beta}}(x_i) = \frac{1}{1+\exp\left[-(\beta_0+\beta_1\, x_i)\right]}$$

Logistic regression

How do we learn the model parameters?

- Optimization problem corresponding to logistic regression parameter learning:

$$\text{minimize} \quad J(\boldsymbol{\beta}) = \sum_{i=1}^{N} j(y_i, f_{\boldsymbol{\beta}}(x_i))$$

$$\text{with respect to} \quad \boldsymbol{\beta}$$

$$\text{where} \quad j(y_i, f_{\boldsymbol{\beta}}(x_i)) = -y_i \log(f_{\boldsymbol{\beta}}(x_i)) - (1 - y_i) \log(1 - f_{\boldsymbol{\beta}}(x_i))$$

$$\text{with} \quad f_{\boldsymbol{\beta}}(x_i) = \frac{1}{1+\exp\left[-(\beta_0+\beta_1 x_i)\right]}$$

- Unfortunately, there is no closed-form solution for this problem.
- However, it is a **smooth** convex optimization problem.
$\Longrightarrow$ Efficient optimization algorithms can be used to solve it, *e.g.* gradient descent method or Newton's method.

Logistic regression

Prediction - 1D example

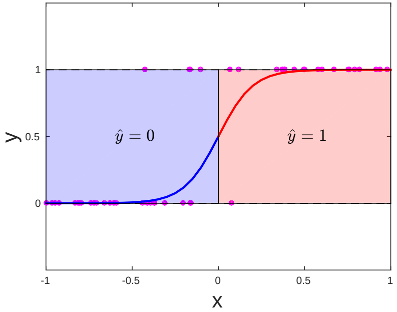
- Prediction can be done as follows

$$\hat{y} = \begin{cases} 1, & \text{if } f_{\hat{\beta}}(x_i) \geq 0.5 \\ 0, & \text{if } f_{\hat{\beta}}(x_i) < 0.5 \end{cases}$$

- This can be rewritten in a simpler equivalent form

$$\hat{y} = \begin{cases} 1, & \text{if } \hat{\beta}_0 + \hat{\beta}_1 \, x \geq 0 \\ 0, & \text{if } \hat{\beta}_0 + \hat{\beta}_1 \, x < 0 \end{cases}$$

- Feature space is partitioned in half-axis as in the example below:



Logistic regression

Generalization to *p* feature vectors

- If you have *p* features corresponding to different explanatory variables and/or transformations of them

$$\mathbf{x} = \begin{bmatrix} 1 & x^{(1)} & \cdots & x^{(p)} \end{bmatrix}$$

simply change $f_{\boldsymbol\beta}(x)$ to $f_{\boldsymbol\beta}(\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{x}\boldsymbol\beta)}$ with $\boldsymbol\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}$

- The prediction is given by

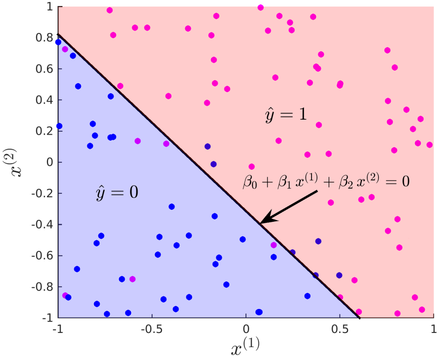
$$\hat{y} = \begin{cases} 1, & \text{if } \mathbf{x}\boldsymbol\beta \geq 0 \\ 0, & \text{if } \mathbf{x}\boldsymbol\beta < 0 \end{cases}$$

This is a linear classifier.

Logistic regression

Prediction - 2D example

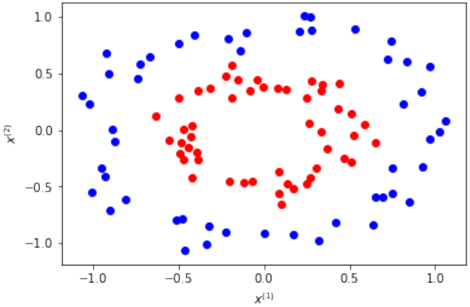
- The previously shown example of a 2D linear classifier was obtained through logistic regression:



- Slightly different parametrization: $\frac{\kappa}{\sqrt{1+\eta^2}}(\cos(\theta)x^{(1)} + \sin(\theta)x^{(2)} + \eta)$
  $\Longrightarrow (\theta, \eta)$ describe the separation line and κ the steepness of the transition from one class to the other.

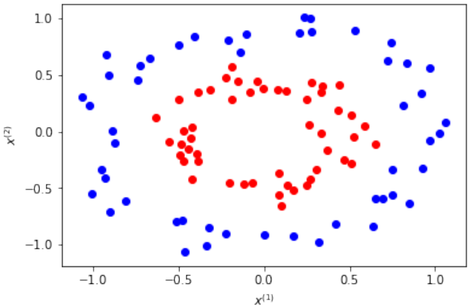
Classes not linearly separable

What do we do if the classes are not close to linearly separability ?



Classes not linearly separable

What do we do if the classes are not close to linearly separability ?



Separating border is close to an ellipse. General equation:

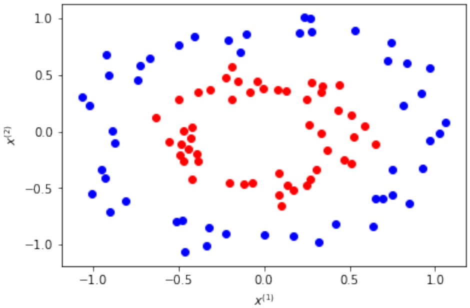
$$\frac{\left[(x^{(1)}-a)\cos(\theta)+(x^{(2)}-b)\sin(\theta)\right]^2}{c^2}+\frac{\left[(x^{(1)}-a)\sin(\theta)+(x^{(2)}-b)\cos(\theta)\right]^2}{d^2}=1$$

Developing, we can rewrite it in the following form:

$$\beta_5\left(x^{(2)}\right)^2+\beta_4\left(x^{(1)}\right)^2+\beta_3x^{(1)}x^{(2)}+\beta_2x^{(2)}+\beta_1x^{(1)}+\beta_0=0$$

Classes not linearly separable

What do we do if the classes are not close to linearly separability ?



Separating border is close to an ellipse. General equation:

$$\frac{\left[(x^{(1)}-a)\cos(\theta)+(x^{(2)}-b)\sin(\theta)\right]^2}{c^2}+\frac{\left[(x^{(1)}-a)\sin(\theta)+(x^{(2)}-b)\cos(\theta)\right]^2}{d^2}=1$$

Developing, we can rewrite it in the following form:

$$\beta_5\left(x^{(2)}\right)^2+\beta_4\left(x^{(1)}\right)^2+\beta_3x^{(1)}x^{(2)}+\beta_2x^{(2)}+\beta_1x^{(1)}+\beta_0=0$$

Classes not linearly separable

Polynomial equation for the separating border:

$$\beta_5 \left(x^{(2)}\right)^2 + \beta_4 \left(x^{(1)}\right)^2 + \beta_3 x^{(1)} x^{(2)} + \beta_2 x^{(2)} + \beta_1 x^{(1)} + \beta_0 = 0$$

We apply the same trick as for fitting polynomials with linear regression.

We need to add the transformed features

$$\left(x^{(2)}\right)^2, \left(x^{(1)}\right)^2 \text{ and } x^{(1)} x^{(2)}$$

to the dataset.

Classes not linearly separable

Polynomial equation for the separating border:

$$\beta_5 \left(x^{(2)}\right)^2 + \beta_4 \left(x^{(1)}\right)^2 + \beta_3 x^{(1)} x^{(2)} + \beta_2 x^{(2)} + \beta_1 x^{(1)} + \beta_0 = 0$$

We apply the same trick as for fitting polynomials with linear regression.

We need to add the transformed features

$$\left(x^{(2)}\right)^2, \left(x^{(1)}\right)^2 \text{ and } x^{(1)} x^{(2)}$$

to the dataset.

Classes not linearly separable

Polynomial equation for the separating border:

$$\beta_5 \left(x^{(2)}\right)^2 + \beta_4 \left(x^{(1)}\right)^2 + \beta_3 x^{(1)} x^{(2)} + \beta_2 x^{(2)} + \beta_1 x^{(1)} + \beta_0 = 0$$

We apply the same trick as for fitting polynomials with linear regression.

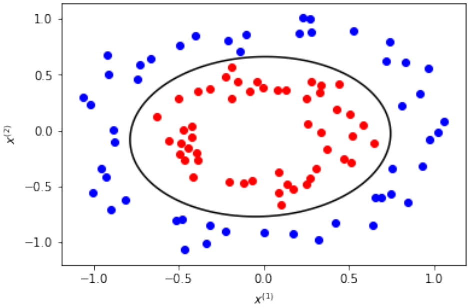
We need to add the transformed features

$$\left(x^{(2)}\right)^2, \left(x^{(1)}\right)^2 \text{ and } x^{(1)} x^{(2)}$$

to the dataset.

Classes not linearly separable

Separating border obtained with logistic regression and transformed features



Accuracy

How do we evaluate the performance of a classifier?

- A natural measure of performance is the **fraction of corrected predictions on a test dataset**. This is called the **accuracy** of the classifier on the test dataset.

  For a dataset $(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}})$ with N_{test} observations, the accuracy $\text{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}})$ of a classifier is

$$\text{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \mathbb{1}\left[\hat{y}(\mathbf{x}^i_{\text{test}}) = y^i_{\text{test}}\right]$$

  where $(\mathbf{x}^i_{\text{test}}, y^i_{\text{test}})$ is the i-th observation of the dataset and $\mathbb{1}\left[\hat{y} = y_{\text{test}}\right]$ is an indicator function:

$$\mathbb{1}\left[\hat{y} = y_{\text{test}}\right] = \begin{cases} 1, & \text{if } \hat{y} = y_{\text{test}}, \\ 0, & \text{otherwise.} \end{cases}$$

Accuracy

- One can also write the accuracy as a function of the fraction of prediction errors:

$$\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\mathrm{test}}, \mathbf{y}_{\mathrm{test}}) = 1 - \left\{ \frac{1}{N_{\mathrm{test}}} \sum_{i=1}^{N_{\mathrm{test}}} \mathbb{1}\left[\hat{y}(\mathbf{x}_{\mathrm{test}}^{i}) \neq y_{\mathrm{test}}^{i}\right] \right\}$$

- $0 \leq \mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\mathrm{test}}, \mathbf{y}_{\mathrm{test}}) \leq 1$.
- Good classifiers should have $\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\mathrm{test}}, \mathbf{y}_{\mathrm{test}})$ close to 1. When $\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\mathrm{test}}, \mathbf{y}_{\mathrm{test}}) = 1$, there are no errors.
- When $\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\mathrm{test}}, \mathbf{y}_{\mathrm{test}}) = 0$, the labels are all flipped.
- In binary classification the worst case is $\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\mathrm{test}}, \mathbf{y}_{\mathrm{test}}) = 0.5$, why?

Accuracy

- One can also write the accuracy as a function of the fraction of prediction errors:

$$\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) = 1 - \left\{ \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \mathbb{1}\left[\hat{y}(\mathbf{x}^i_{\text{test}}) \neq y^i_{\text{test}}\right] \right\}$$

- $0 \leq \mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) \leq 1$.
- Good classifiers should have $\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}})$ close to 1. When $\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) = 1$, there are no errors.
- When $\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) = 0$, the labels are all flipped.
- In binary classification the worst case is $\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) = 0.5$, why?

Accuracy

- One can also write the accuracy as a function of the fraction of prediction errors:

$$\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) = 1 - \left\{ \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \mathbb{1}\left[\hat{y}(\mathbf{x}^i_{\text{test}}) \neq y^i_{\text{test}}\right] \right\}$$

- $0 \leq \mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) \leq 1$.
- Good classifiers should have $\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}})$ close to 1. When $\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) = 1$, there are no errors.
- When $\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) = 0$, the labels are all flipped.
- In binary classification the worst case is $\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) = 0.5$, why?

Accuracy

- One can also write the accuracy as a function of the fraction of prediction errors:

$$\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) = 1 - \left\{ \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \mathbb{1}\left[\hat{y}(\mathbf{x}^i_{\text{test}}) \neq y^i_{\text{test}}\right] \right\}$$

- $0 \leq \mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) \leq 1$.

- Good classifiers should have $\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}})$ close to 1. When $\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) = 1$, there are no errors.

- When $\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) = 0$, the labels are all flipped.

- In binary classification the worst case is $\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) = 0.5$, why?

Accuracy

- One can also write the accuracy as a function of the fraction of prediction errors:

$$\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) = 1 - \left\{ \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \mathbb{1}\left[\hat{y}(\mathbf{x}_{\text{test}}^{i}) \neq y_{\text{test}}^{i}\right] \right\}$$

- $0 \leq \mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) \leq 1$.
- Good classifiers should have $\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}})$ close to 1. When $\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) = 1$, there are no errors.
- When $\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) = 0$, the labels are all flipped.
- In binary classification the worst case is $\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) = 0.5$, why?

Accuracy

- Why not use accuracy as a criterion $J(\beta)$ for learning the parameters of a classifier?

Accuracy

- Why not use accuracy as a criterion $J(\beta)$ for learning the parameters of a classifier?

  Due to the presence of the indicator function, accuracy is a non smooth, non convex criterion. Very difficult to be maximized directly by numerical optimization algorithms.

Accuracy

Interpretation of accuracy in the binary case

- Denote $\mathcal{I}_0$ the set of indexes of the observations in the test dataset for which $y = 0$ and N_{test}^0 the corresponding number of observations.

  Denote $\mathcal{I}_1$ the set of indexes of the observations in the test dataset for which $y = 1$ and N_{test}^1 the corresponding number of observations.

  Then we can write

$$\text{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, y_{\text{test}}) = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \mathbb{1}\left[\hat{y}(\mathbf{x}_{\text{test}}^i) = y_{\text{test}}^i\right]$$

$$= \frac{1}{N_{\text{test}}^0 + N_{\text{test}}^1} \left\{ \sum_{i \in \mathcal{I}_0} \mathbb{1}\left[\hat{y}(\mathbf{x}_{\text{test}}^i) = 0\right] + \sum_{j \in \mathcal{I}_1} \mathbb{1}\left[\hat{y}(\mathbf{x}_{\text{test}}^j) = 1\right] \right\}$$

$$= \left(\frac{N_{\text{test}}^0}{N_{\text{test}}^0 + N_{\text{test}}^1} \right) \frac{\sum_{i \in \mathcal{I}_0} \mathbb{1}\left[\hat{y}(\mathbf{x}_{\text{test}}^i) = 0\right]}{N_{\text{test}}^0} + \left(\frac{N_{\text{test}}^1}{N_{\text{test}}^0 + N_{\text{test}}^1} \right) \frac{\sum_{j \in \mathcal{I}_1} \mathbb{1}\left[\hat{y}(\mathbf{x}_{\text{test}}^j) = 1\right]}{N_{\text{test}}^1}$$

Accuracy

Interpretation of accuracy in the binary case

$$\mathrm{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, y_{\text{test}}) = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \mathbb{1}\left[\hat{y}(\mathbf{x}^i_{\text{test}}) = y^i_{\text{test}}\right]$$

$$= \left(\frac{N^0_{\text{test}}}{N^0_{\text{test}} + N^1_{\text{test}}}\right) \frac{\sum_{i \in \mathcal{I}_0} \mathbb{1}\left[\hat{y}(\mathbf{x}^i_{\text{test}}) = 0\right]}{N^0_{\text{test}}} + \left(\frac{N^1_{\text{test}}}{N^0_{\text{test}} + N^1_{\text{test}}}\right) \frac{\sum_{j \in \mathcal{I}_1} \mathbb{1}\left[\hat{y}(\mathbf{x}^j_{\text{test}}) = 1\right]}{N^1_{\text{test}}}$$

- Accuracy is a weighted combination of the fractions of correct predictions for each class.
- The weights are the fraction of observations for each class.
- Consequence: accuracy is not sensitive to the prediction errors in rare classes.

Confusion matrix

- A more complete picture of the classifier performance can be obtained by gathering in a **table** the **number of occurences of each possible couple (*y*,$\hat{y}$)**.

Example for binary classification:

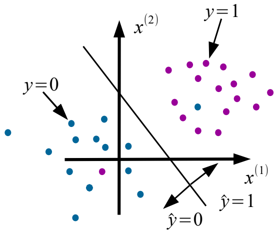
True y \ Predicted $\hat{y}$	0	1
0	$\sum_{i \in \mathcal{I}_0} \mathbb{1}\left[\hat{y}(\mathbf{x}^i_{\text{test}}) = 0\right]$ **True Negatives**	$\sum_{i \in \mathcal{I}_0} \mathbb{1}\left[\hat{y}(\mathbf{x}^i_{\text{test}}) = 1\right]$ **False Positives**
1	$\sum_{i \in \mathcal{I}_1} \mathbb{1}\left[\hat{y}(\mathbf{x}^i_{\text{test}}) = 0\right]$ **False Negatives**	$\sum_{i \in \mathcal{I}_1} \mathbb{1}\left[\hat{y}(\mathbf{x}^i_{\text{test}}) = 1\right]$ **True Positives**

This table is called the **confusion matrix**.

Simple examples

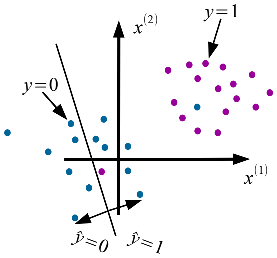
Give the accuracy and the confusion matrix for the classifiers below

$N_{\text{test}} = \ ?$ $\quad N^0_{\text{test}} = \ ?$ $\quad N^1_{\text{test}} = \ ?$



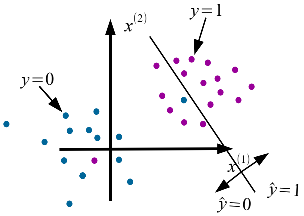
$\text{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, y_{\text{test}}) = \ ?$

$y \backslash \hat{y}$	0	1
0	?	?
1	?	?



$\text{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, y_{\text{test}}) = \ ?$

$y \backslash \hat{y}$	0	1
0	?	?
1	?	?



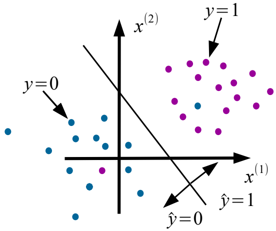
$\text{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, y_{\text{test}}) = \ ?$

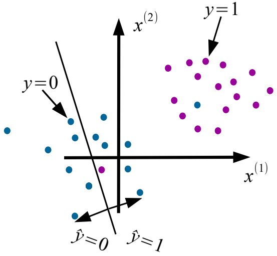
$y \backslash \hat{y}$	0	1
0	?	?
1	?	?

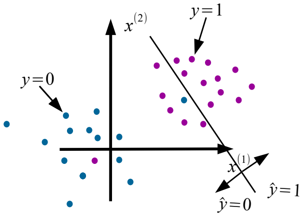
Simple examples

Give the accuracy and the confusion matrix for the classifiers below

$$N_{\text{test}} = 31 \qquad N^0_{\text{test}} = 14 \qquad N^1_{\text{test}} = 17$$







$\text{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, y_{\text{test}}) = 0.94$

y \ $\hat{y}$	0	1
0	13	1
1	1	16

$\text{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, y_{\text{test}}) = 0.77$

y \ $\hat{y}$	0	1
0	7	7
1	0	17

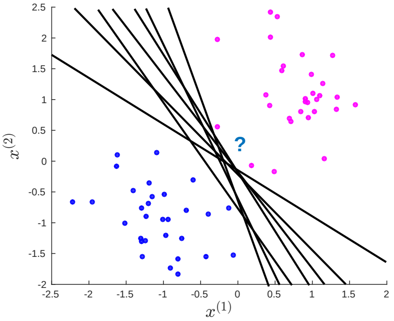
$\text{Acc}_{\hat{\mathbf{y}}(\mathbf{x})}(\mathbf{X}_{\text{test}}, y_{\text{test}}) = 0.81$

y \ $\hat{y}$	0	1
0	14	0
1	6	11

Logistic regression

What happens if the classes are linearly separable?

- Which line do you choose?



- Logistic regression suffers from two major issues in this case:
  1. $\kappa \to +\infty$ to have $J \to 0$: optimization algorithms become unstable.
  2. The solution of the problem is not unique.

Regularized logistic regression

Solution

- Add a regularization term $\|\boldsymbol{\beta}\|_2^2$ to $J(\boldsymbol{\beta})$:

minimize $$J'(\boldsymbol{\beta}) = \left[\sum_{i=1}^{N} j(y_i, f_{\boldsymbol{\beta}}(\mathbf{x}_i))\right] + \lambda \|\boldsymbol{\beta}\|_2^2$$

with respect to $\boldsymbol{\beta}$

where $$j(y_i, f_{\boldsymbol{\beta}}(\mathbf{x}_i)) = -y_i \log(f_{\boldsymbol{\beta}}(\mathbf{x}_i)) - (1 - y_i) \log(1 - f_{\boldsymbol{\beta}}(\mathbf{x}_i))$$

with $$f_{\boldsymbol{\beta}}(\mathbf{x}_i) = \frac{1}{1 + \exp\left[-\mathbf{x}_i \boldsymbol{\beta}\right]}$$

some fixed $\lambda > 0$.

- $\boldsymbol{\beta}$ is not allowed to grow unbounded. This stabilizes numerical optimizers.
- We get a unique solution.
- When $\lambda \to 0$, we get standard logistic regression.

Conclusions

Logistic regression

- Logistic regression can be seen as a simple adaptation of linear regression to do classification. Learning is equivalent to solve a smooth optimization problem.
- Logistic regression is a linear classifier. If the classifier needs nonlinear separation boundaries, we can include nonlinear transformations of the features to the feature matrix (but we need to know what to add).
- Parameter vector $\hat{\beta}$ allows for interpretation of results.
- Complexity of underlying optimization problem depends on the dimension of feature space.
- It is sensible to outliers (it is designed for this).
- Its standard form cannot be used for classifying linear separable classes. Its regularized version should be used in this case.