MACHINE LEARNING

MEI/1

University of Beira Interior,
Department of Informatics

Hugo Pedro Proença,
hugomcp@di.ubi.pt, 2025/2026

Machine Learning

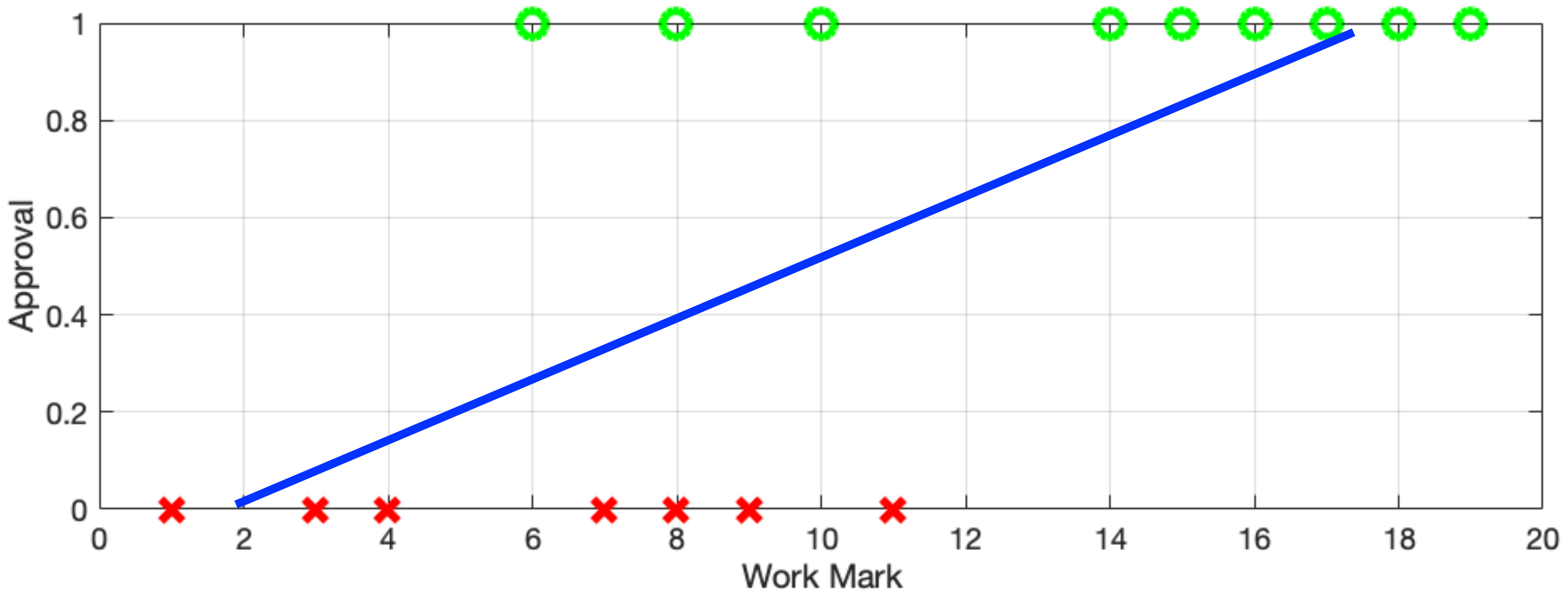
[03]

Syllabus

- Supervised Learning
  - Classification
    - Logistic Regression

Students Performance

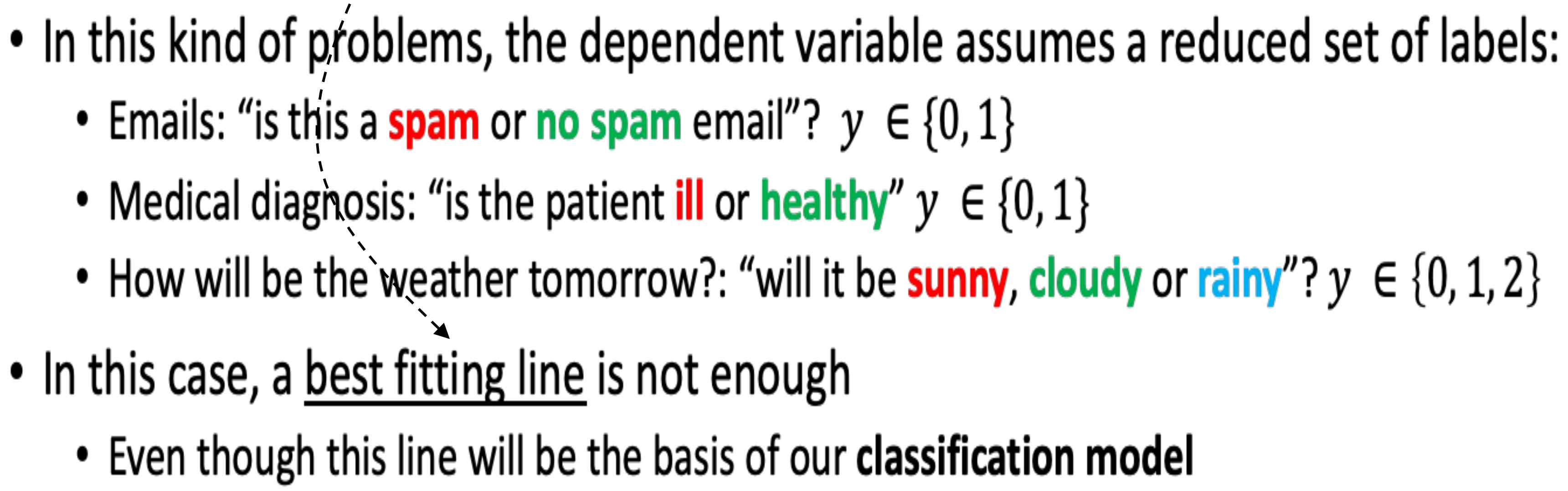
- Suppose that we are interested in predicting the **approval rate** of a class, based on the **students marks in the first practical work**.
  - Typically, students that get good marks in the first work, got approved at the course.
  - Students with very low marks at the first work tend to fail in the final examination.
- Hence, our machine learning model is expected to predict a **binary outcome** (**1: pass** vs. **0: fail**)



Students Performance

- In this kind of problems, the dependent variable assumes a reduced set of labels:
  - Emails: “is this a **spam** or **no spam** email”? $y \in \{0, 1\}$
  - Medical diagnosis: “is the patient **ill** or **healthy**” $y \in \{0, 1\}$
  - How will be the weather tomorrow?: “will it be **sunny**, **cloudy** or **rainy**”? $y \in \{0, 1, 2\}$
- In this case, a <u>best fitting line</u> is not enough
  - Even though this line will be the basis of our **classification model**

$$h_\theta(x) = \theta_1 . x + \theta_2$$



- In this kind of problems, the dependent variable assumes a reduced set of labels:
  - Emails: “is this a **spam** or **no spam** email”? $y \in \{0, 1\}$
  - Medical diagnosis: “is the patient **ill** or **healthy**” $y \in \{0, 1\}$
  - How will be the weather tomorrow?: “will it be **sunny**, **cloudy** or **rainy**”? $y \in \{0, 1, 2\}$
- In this case, a <u>best fitting line</u> is not enough
  - Even though this line will be the basis of our **classification model**

Logistic Regression: Classification

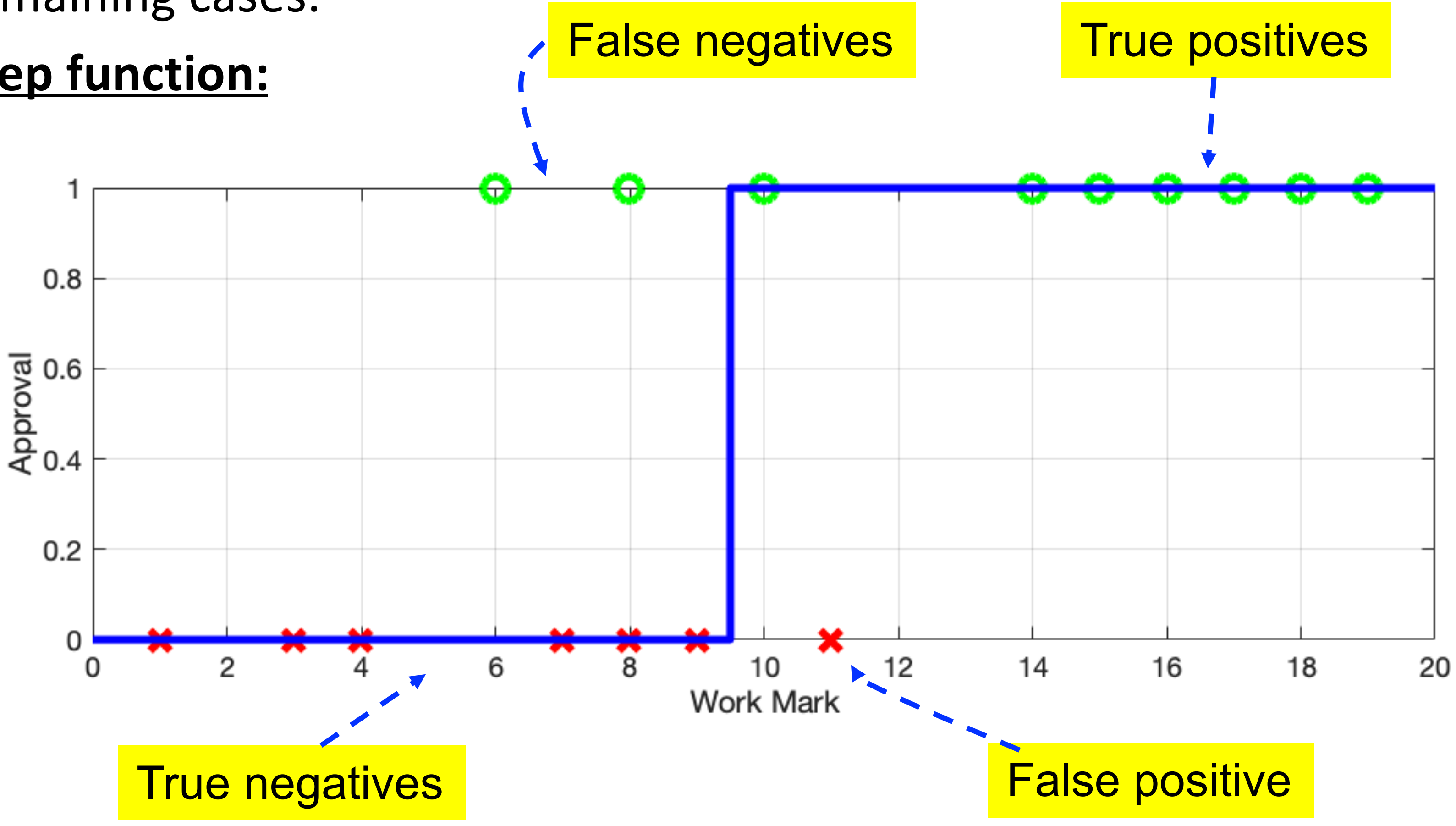
- The obvious idea will be to define a threshold at the classifier output $h_\theta(x)$, that binarizes the system response:
  - Typically, "0.5" would be the choice, for "*equal classification risk*"
    - It might be more dangerous to predict erroneously one class instead of other one.
    - For example, in a machine learning-based systems for medical diagnosis, classes have different risk.
      - Predict a "**malignant cancer**" on a "**healthy**" subject represents a unnecessary concern for the patient and would probably imply to perform additional (an unnecessary) exams.
      - However, provide a "**healthy**" response for a patient suffering of a "**malignant cancer**" might represent the patient dead sentence.
  - $f(x) = \begin{cases} 0, h_\theta(x) < 0 \\ 1, h_\theta(x) \geq 0 \end{cases}$
- Hence, the response of our classification system can be seen as a composition of two functions: $f = g\ o\ h$

"f" is "g" after "h"

- *We have seen "h" before…*
  - *It is the best fitting line, of the previously seen regression problem*
- *…but what is "g"?*

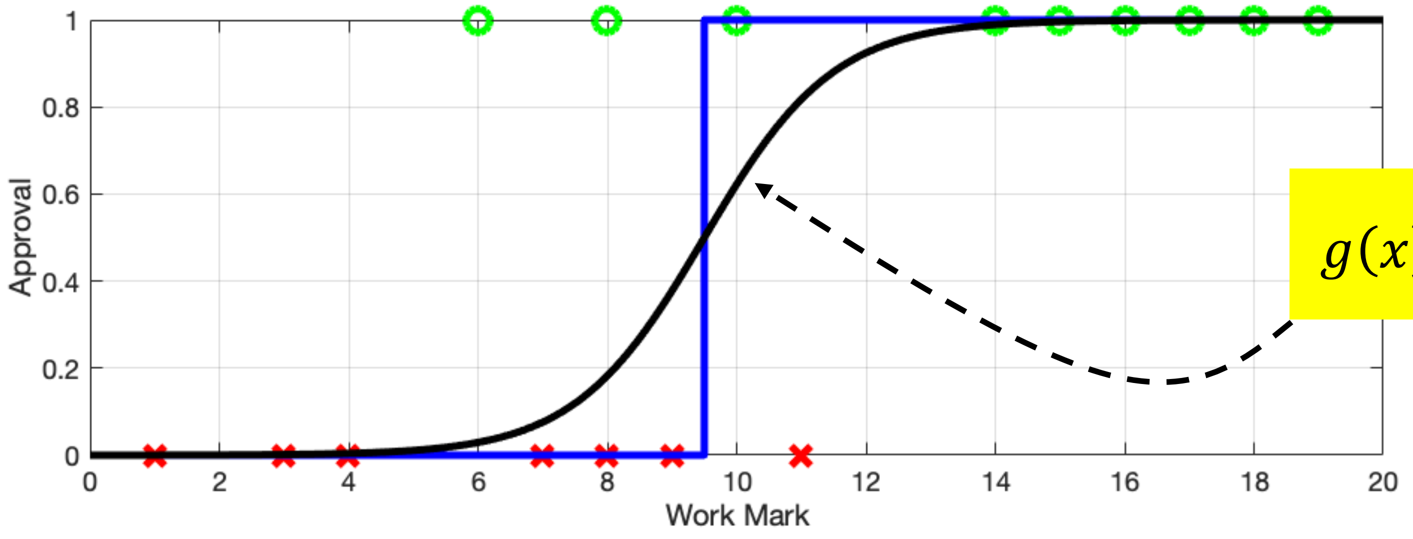
Logistic Regression: Classification

- Essentially, “g” performs a binarization of its input, and produces ”1” responses when the input is higher than some threshold, and “0” in the remaining cases.
- **<u>Step function:</u>**



Logistic Regression: Classification

- Assuming the step function as “g”, and $f = g \; o \; h$, obtaining the automatic optimal parameterization of “f” with respect to our data (i.e., machine learning) yields two problems:
  - **<u>Problem 1</u>**: “g” is **not differentiable**
    - It has not a continuous derivative at a single point
  - **Problem 2:** in every other points **“g” has derivative 0**
- The solution is to use a function is close to the step function, without suffering of the above described problems.
  - **Sigmoid Function**



$$g(x) = \frac{1}{1 + e^{-x}}$$

Logistic Regression: Classification

- Using this composition of functions, our classification system is given by:

$$f_{\boldsymbol{\theta}}(x) = \frac{1}{1 + e^{h_\theta(x)}}$$

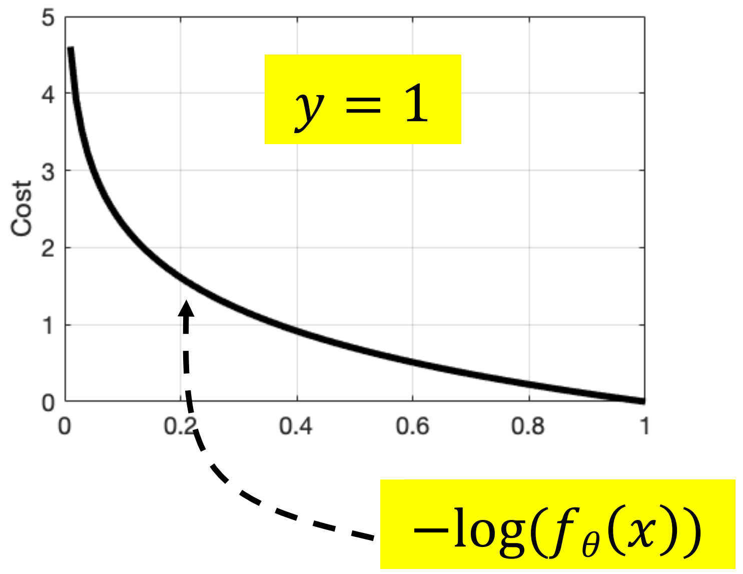
i.e.,

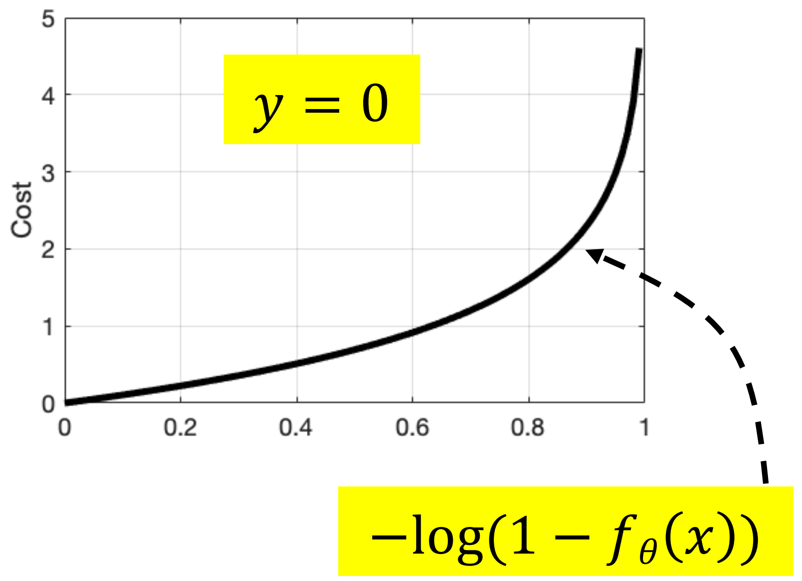
$$f_{\boldsymbol{\theta}}(x) = \frac{1}{1+e^{\theta_1 x + \theta_2}}$$

- The remaining problem is the same as in linear regression:
  - How to find the $\boldsymbol{\theta}$ optimal parameterization?
- According to the basic principles of Machine Learning, up to now we've only defined our model.
  - It is also required to define a "Cost Function" (Loss function) that measures how good it is na hypothesis.
  - ...And a systematic optimization process.

Binary Classification: Cost Function

- As previously, the cost function will measure how well the model responses ($f_{\boldsymbol{\theta}}(x)$) resemble the "ground-truth" (y)
  - Intuitively, in cases where the system is supposed to output a "1" and the model predicts a "1", the cost should be "0".
  - The same thing should hold for "0" responses.
  - However, the cost (loss) should grow in cases when the system response is far from the ground-truth.
    - The log() function is a good choice for representing the desired costs (losses)
    - It varies non-linearly with respect to the distance between the desired and actual responses
      - Attempts to avoid "**very wrong responses**".





Binary Classification: Cost Function

- Hence, the cost function for one instance is given by:
  - $Cost(f_\theta(x), y) = \begin{cases} -\log(f_\theta(x)) \,, & y = 1 \\ -\log\big(1 - f_\theta(x)\big) \,, & y = 0 \end{cases}$
- And the cost function for the whole dataset is given by the sum of the individual costs:

$$\mathrm{J}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \left(Cost\left(f_\theta(x^{(i)}), y^{(i)}\right) \right)$$

- Considering that y can only assume 2 values (0 or 1), we have:

$$\mathrm{J}(\boldsymbol{\theta}) = -\frac{1}{N} \sum_{i=1}^{N} y^{(i)} \log\left(\boldsymbol{f_\theta}(x^{(i)})\right) + (1 - y^{(i)}) \log\left(1 - \boldsymbol{f_\theta}(x^{(i)})\right)$$

Logistic Regression: Optimization

- The optimization can be done exactly as in the linear regression case.
- Using the gradient descent strategy, it is required to find the derivatives of the cost function J() with respect to the $\boldsymbol{\theta}$ parameters:

$$\frac{\int}{\int \boldsymbol{\theta}} \mathrm{J}(\boldsymbol{\theta})$$

- In matrix form, we have:

$$\boldsymbol{\theta} = [\theta_0, \theta_1]^{\mathrm{T}} \qquad \boldsymbol{x}^{(\mathrm{i})} = [x^{(\mathrm{i})}, 1]^{\mathrm{T}}$$

- $\mathrm{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{\theta}^T \boldsymbol{x}}}$
- $\log(f_{\boldsymbol{\theta}}(\boldsymbol{x})) = \log(\frac{1}{1+e^{-\boldsymbol{\theta}^T \boldsymbol{x}}})$

  $= -\log(\frac{1+e^{-\boldsymbol{\theta}^T \boldsymbol{x}}}{1})$
- $\log(1 - f_{\boldsymbol{\theta}}(\boldsymbol{x})) = -\theta\mathbf{x} - \log(\frac{1+e^{-\boldsymbol{\theta}^T \boldsymbol{x}}}{1})$

Logistic Regression: Optimization

- Plugging the two simplified expressions in the original cost function:

$$\mathrm{J}(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N} - y^{(i)} \log\left(1 + e^{-\boldsymbol{\theta x}}\right) + (1 - y^{(i)}) \left(-\boldsymbol{\theta x} - \log\left(1 + e^{-\boldsymbol{\theta x}}\right)\right.$$

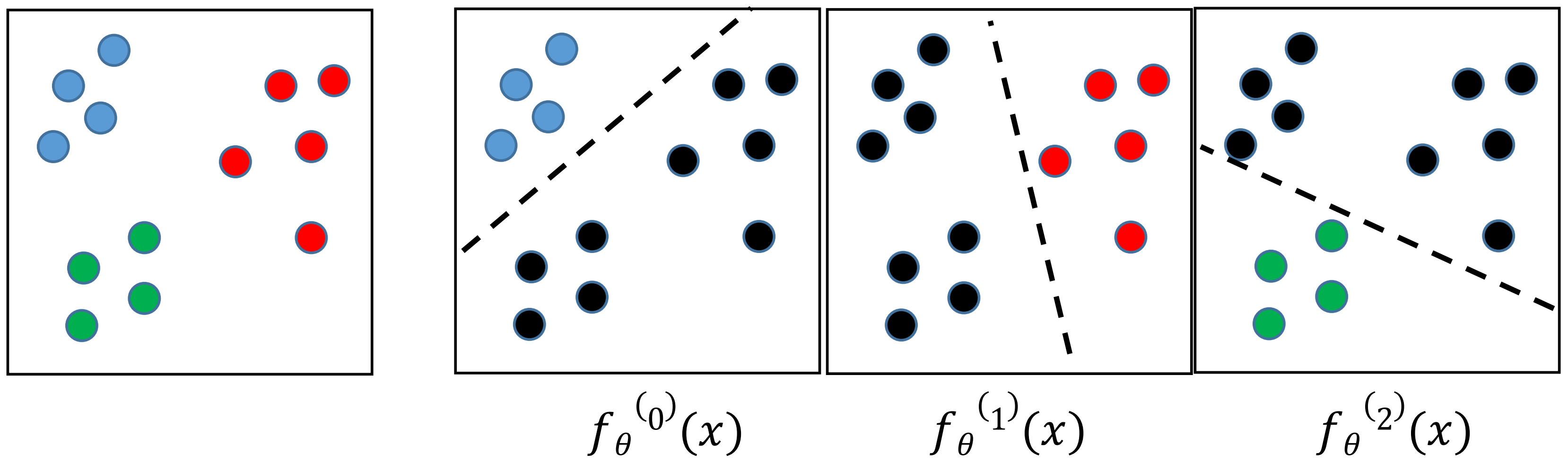
- Which can be simplified to: $\mathrm{J}(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N} y^{(i)}\, \boldsymbol{\theta x} - \log\left(1 + e^{-\boldsymbol{\theta x}}\right)$
- Obtaining the derivative of J($\boldsymbol{\theta}$) with respect to each θ_j requires some effort and calculus.
  - For the moment, we will use the automatic differentiation module of Keras/Pytorch, in the gradient descent step.

```
def custom_mse(y_true, y_pred):
    squared_difference = tf.square(y_true - y_pred)
    return tf.reduce_mean(squared_difference, axis=-1)

model = tf.keras.Sequential([
    tf.keras.layers.Dense(16, input_shape=[4], activation='relu'),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1)
])

model.compile(loss=custom_mse, optimizer='adam')
history = model.fit(x, y, epochs=10)
```

Logistic Regression: Multi-class

- Up to now, we've only considering binary classification problems.
- When the number of classes $c \geq 2$, the typical approach is to train "c" classifiers
  - In each classifier $f_\theta^{(i)}(x)$, instances of the ith class are considered positive examples, whereas instances of al the remaining classes are regarded as negative instances.
- During classification, we pick the class that produces the maximum output response, i.e.:

$$\max_i f_\theta^{(i)}(x)$$


$f_\theta^{(0)}(x)$
$f_\theta^{(1)}(x)$
$f_\theta^{(2)}(x)$