# GRA4150
# AI - Technologies and Applications

Lecture 7

February 21, 2023

# TODAY

- Logistic regression;
  - The sigmoid activation function;
  - The cross-entropy loss function;
  - Example on the notebook;
- Multi-class classification:
  - One-versus-All;
  - Softmax regression;
  - Example on the notebook.

# RECAP

- We considered **supervised ML** models for **binary classification**;

# RECAP

- We considered **supervised ML** models for **binary classification**;
- We started with the PERCEPTRON MODEL:
  - Not really a motivation for the updating rule (it is sort of "given");
  - The output $\hat{y}$ is obtained by applying a **threshold function** directly to the net input;
  - The updating step is computed from the final output $\hat{y}$ and the observation $y$;
  - Notice that the convergence of the method can be proved mathematically.

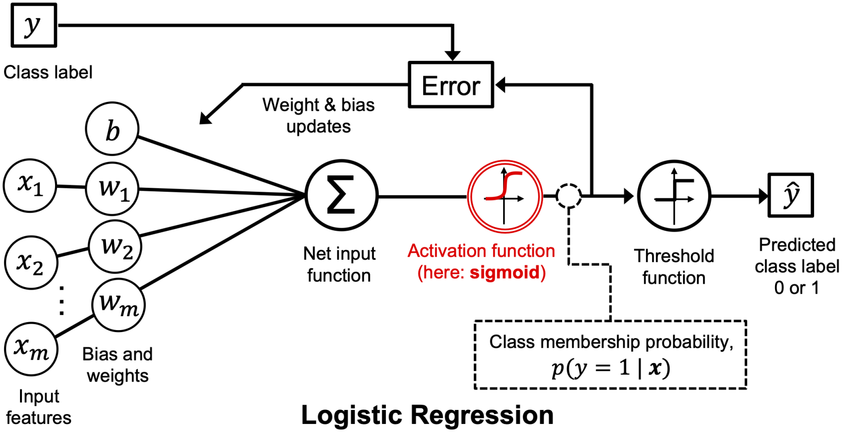
# RECAP

- We considered **supervised ML** models for **binary classification**;
- We started with the PERCEPTRON MODEL:
  - Not really a motivation for the updating rule (it is sort of "given");
  - The output $\hat{y}$ is obtained by applying a **threshold function** directly to the net input;
  - The updating step is computed from the final output $\hat{y}$ and the observation $y$;
  - Notice that the convergence of the method can be proved mathematically.
- The ADALINE METHOD:
  - It is an extension of the previous model;
  - Here we **add an intermediate step**: an **activation function** is applied to the net input to get the intermediate output $\tilde{y}$;
  - The updating step is computed from the intermediate output $\tilde{y}$ and observation $y$;
  - In this case, the activation function is in fact a linear activation function;
  - The benefit of this approach is that the linear activation function is **differentiable**;
  - We can then define an **objective function** (the MSE loss function) and use a **gradient method** for the optimization;
  - Only in the very last step, the intermediate output $\tilde{y}$ is converted in final output $\hat{y}$ via a threshold function.

# THE LOGISTIC REGRESSION MODEL

- Logistic regression is also a model for binary classification;
- It is one of the most widely used algorithms for classification in industry;
- It is a **probabilistic model**;
- It easily generalizes to multi-class classification.

# THE LOGISTIC REGRESSION ALGORITHM



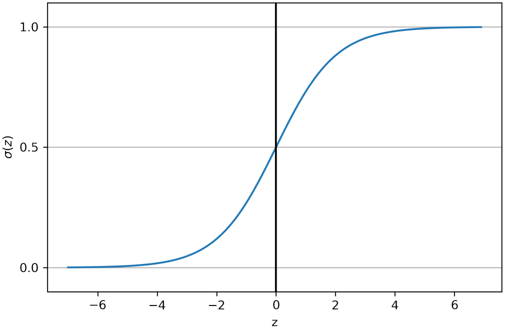

What is this similar to? What is/are the difference(s)?

# THE SIGMOID FUNCTION

We define the **logistic sigmoid function**, or simply **sigmoid function** by:

$$\sigma(z) := \frac{1}{1+e^{-z}}$$

where $z = \mathbf{w}^\top \mathbf{x} + b$ is the net input.




Notice that:

- For $z$ approaching $+\infty$, $\sigma(z)$ approaches 1;
- For $z$ approaching $-\infty$, $\sigma(z)$ approaches 0;
- Hence $\sigma : \mathbb{R} \to [0,1]$ with intercept $\sigma(0) = 0.5$.

# WHY THE SIGMOID FUNCTION?

Let $p \in [0, 1]$ the probability of a "positive event" (here "positive" does not mean "good", but refers to the event that we want to predict).

## Example

*We consider p as the probability that a patient has a certain disease given certain symptoms. Then we think of the positive event as class label* $y = 1$ *and the symptoms as feature* $\mathbf{x}$*, and* $p := p(y = 1|\mathbf{x})$ *is the probability that a particular example belongs to the class* $1$ *given* $\mathbf{x}$*.*

**The output of the sigmoid function is interpreted as the probability of a particular example belonging to class** $1$:

$$\sigma(z) = p(y = 1|\mathbf{x}; \mathbf{w}, b)$$

given its features $\mathbf{x}$ and parameterized by the weights and bias, $\mathbf{w}$ and $b$.

## Example

*If* $\sigma(z) = 0.8$ *for a particular flower example, it means that the chance that this example is an Iris-versicolor flower is* $80\%$*.Therefore, the probability that this flower is an Iris-setosa flower can be calculated as* $p(y = 0|\mathbf{x}; \mathbf{w}, b) = 1 - p(y = 1|\mathbf{x}; \mathbf{w}, b) = 0.2$*, or* $20\%$*.*

# THE (same) THRESHOLD FUNCTION (as Adaline)

The predicted probability can then simply be converted into a binary outcome via a threshold function:

$$\hat{y} = \begin{cases} 1 & \text{if } \sigma(z) \geq 0.5 \\ 0 & \text{if } \sigma(z) < 0.5 \end{cases}$$

## REMARK

**There are many applications where we are interested in** both the predicted class labels and in **the estimation of the class-membership probability** (the output of the sigmoid function prior to applying the threshold function).

Examples:

- Weather forecasting: we are interested not only to predict whether it will rain on a particular day, but also to report the chance of rain.
- Medicine: logistic regression can be used to predict the chance that a patient has a particular disease given certain symptoms.

# WHAT ABOUT THE LEARNING PROCESS?

- We need to define an objective function that we seek to minimize;
- The idea is that we want to **maximize the likelihood**, namely the probability of classifying the dataset correctly, given some parameters **w** and $b$;
- This is equivalent to minimize the **CROSS-ENTROPY loss function**:

$$L(\mathbf{w}, b) := \sum_{i=1}^{n} \left[ -y^{(i)} \log\left(\sigma(z^{(i)})\right) - (1 - y^{(i)}) \log\left(1 - \sigma(z^{(i)})\right) \right].$$

## Example: $n = 1$

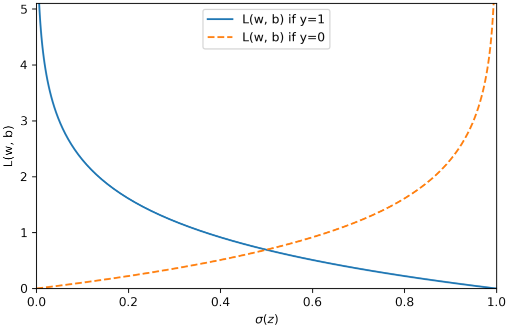
$$L(\mathbf{w}, b) = -y^{(1)} \log\left(\sigma(z^{(1)})\right) - (1 - y^{(1)}) \log\left(1 - \sigma(z^{(1)})\right)$$

We notice that:

$$L(\mathbf{w}, b) = \begin{cases} -\log\left(\sigma(z^{(1)})\right) & \text{if } y^{(1)} = 1 \\ -\log\left(1 - \sigma(z^{(1)})\right) & \text{if } y^{(1)} = 0 \end{cases}.$$

# THE CROSS-ENTROPY FUNCTION EXPLAINED

$$L(\mathbf{w}, b) = \begin{cases} -\log\left(\sigma(z^{(1)})\right) & \text{if } y^{(1)} = 1 \\ -\log\left(1 - \sigma(z^{(1)})\right) & \text{if } y^{(1)} = 0 \end{cases}$$




- Blue continuous line: the loss approaches 0 if we correctly predict that an example belongs to class 1;
- Orange dashed line: the loss also approaches 0 if we correctly predict $y = 0$;
- If the prediction is wrong, the loss goes toward infinity;
- **We penalize wrong predictions with an increasingly larger loss!**

# THE ALGORITHM

The logistic regression algorithm can be obtained starting from the Adaline implementation:

1. By substituting the MSE loss function with the cross-entropy loss function;
2. By changing the linear activation function with the sigmoid function;

- What about the gradient for the optimization step?

# THE ALGORITHM

The logistic regression algorithm can be obtained starting from the Adaline implementation:

1. By substituting the MSE loss function with the cross-entropy loss function;
2. By changing the linear activation function with the sigmoid function;

- What about the gradient for the optimization step?
- Using calculus, we can show (optional exercise at home) that **the parameter updates via gradient descent are equal to the ones for Adaline**, namely the gradient step is unchanged!

# THE GRADIENT STEP (unchanged)

- By **gradient descent**, we update the model parameters by taking a step in the opposite direction of the gradient of the loss function:

$$\mathbf{w} = \mathbf{w} + \Delta\mathbf{w} \qquad \text{with } \Delta\mathbf{w} = -\eta\nabla_w L(\mathbf{w}, b)$$
$$b = b + \Delta b \qquad \text{with } \Delta b = -\eta\nabla_b L(\mathbf{w}, b)$$

- $\nabla_w L(\mathbf{w}, b)$ is the vector whose components are the partial derivatives of the loss function with respect to each weight $w_j$:

$$\nabla_w L(\mathbf{w}, b) = \begin{pmatrix} \frac{\partial L}{\partial w_1} & \frac{\partial L}{\partial w_2} & \cdots & \frac{\partial L}{\partial w_m} \end{pmatrix}^\top \text{ with } \frac{\partial L}{\partial w_j} = -\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \sigma(z^{(i)})\right)x_j^{(i)};$$

- $\nabla_b L(\mathbf{w}, b)$ corresponds to the partial derivative of the loss function with respect to the bias $b$:

$$\nabla_b L(\mathbf{w}, b) = \frac{\partial L}{\partial b} = -\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \sigma(z^{(i)})\right).$$

# THE LOGISTIC REGRESSION MODEL

- Logistic regression works for binary classification;
- We consider a different activation function $\sigma$, the sigmoid function;
- We consider a different loss function $L$, the cross-entropy function;
- It has the additional property of providing the class-membership likelihood/probability: this is the output of the sigmoid activation function.

# THE LOGISTIC REGRESSION MODEL

- Logistic regression works for binary classification;
- We consider a different activation function $\sigma$, the sigmoid function;
- We consider a different loss function $L$, the cross-entropy function;
- It has the additional property of providing the class-membership likelihood/probability: this is the output of the sigmoid activation function.

**What about multi-class classification?**

# THE LOGISTIC REGRESSION MODEL

- Logistic regression works for binary classification;
- We consider a different activation function $\sigma$, the sigmoid function;
- We consider a different loss function $L$, the cross-entropy function;
- It has the additional property of providing the class-membership likelihood/probability: this is the output of the sigmoid activation function.

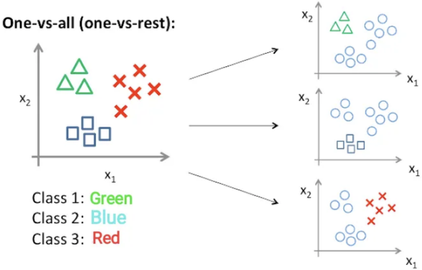
**What about multi-class classification?**

We can consider two different approaches:

- The One-vs-All method;
- The multinomial logistic regression.

# FIRST APPROACH:
# THE OvA METHOD FOR MULTI-CLASS CLASSIFICATION

- The OvA (= One-versus-All) method is a technique that allows to extend probabilistic binary classifiers to multi-class problems;
- The idea is to **train one classifier per class**, where the particular class is treated as the positive class ($y = 1$) and the examples from all other classes are considered negative classes ($y = 0$);
- If there are $N$ classes, then **we train $N$ different classifiers**;
- We choose the class label with the highest confidence.



# OvA: EXAMPLE

We have to generate the same number of classifiers as the number of class labels:

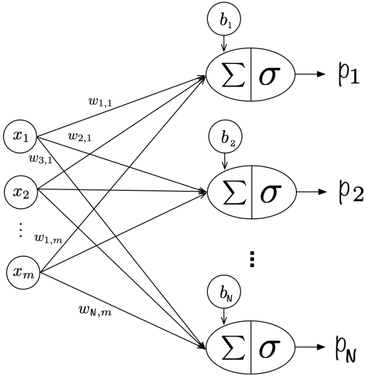
- Classifier 1: [Green] vs [Red, Blue], answers the question "Is it green?"
- Classifier 2: [Blue] vs [Green, Red], answers the question "Is it blue?"
- Classifier 3: [Red] vs [Blue, Green], answers the question "Is it red?"

Important: to train these three classifiers, we need to create three training datasets.

After training, we pass a test data to the model:

- Suppose that from the first classifier we get a probability score of $p = 0.8$;
  (it means probability of being Green 0.8 and probability of not being Green 0.2)
- From the second classifier we get a probability score of $p = 0.4$;
  (it means probability of being Blue 0.4 and probability of not being Blue 0.6)
- From the third classifier we get a probability score of $p = 0.35$;
  (it means probability of being Red 0.35 and probability of not being Red 0.65)
- We can say that the test input belongs to the Green class.

# OvA FOR MULTI-CLASS CLASSIFICATION




Every classifier returns a class probability

$$\sigma(z_h) = p_h \quad \text{with} \quad z_h := \mathbf{w}_h^\top \mathbf{x} + b_h \quad \text{for} \quad h = 1, \ldots, N.$$

Here each $\mathbf{w}_h$ is a vector with components $\mathbf{w}_h^\top = (w_{h,1}, w_{h,2}, \ldots, w_{h,m})$.

# OvA IN MATRIX FORM

Instead of writing for $h = 1, \ldots, N$,

$$\sigma(z_h) = p_h \text{ with } z_h := \mathbf{w}_h^\top \mathbf{x} + b_h,$$

we can use the compact form

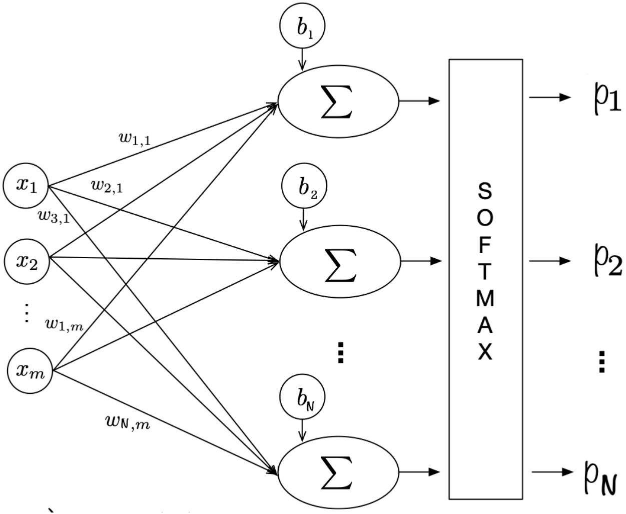
$$\sigma(\mathbf{z}) = \mathbf{p} \text{ with } \mathbf{z} := W\mathbf{x} + \mathbf{b}$$

where

- $\mathbf{z} \in \mathbb{R}^N$ is the vector with components $z_h$;
- $\mathbf{p} \in \mathbb{R}^N$ is the vector with components $p_h$;
- $\mathbf{b} \in \mathbb{R}^N$ is the vector with components $b_h$;
- $W \in \mathbb{R}^{N \times m}$ is the matrix whose rows are the vectors $\mathbf{w}_h^\top$;
- ...and $N$ is the number of classes.
- Notice that with this notation we mean that $\sigma$ acts component-wise, namely $\sigma(\mathbf{z}) = (\sigma(z_1), \ldots, \sigma(z_N))^\top$.

# SECOND APPROACH: SOFTMAX REGRESSION

## Multinomial logistic regression



# THE SOFTMAX ACTIVATION FUNCTION

The softmax function $\sigma : \mathbb{R}^N \to [0,1]^N$ is defined as

$$\sigma(\mathbf{z})_h = \frac{e^{z_h}}{\sum_{\ell=1}^{N} e^{z_\ell}},$$

where $\sigma(\mathbf{z})_h$ is the $h$-th component of $\sigma(\mathbf{z}) = \sigma\left(W\mathbf{x} + \mathbf{b}\right)$, for $h = 1, \ldots, N$.

In words:

- It applies the exponential function to each element $z_h$ of the input vector $\mathbf{z}$;
- It normalizes these values by dividing by the sum of all the exponentials;
- The normalization ensures that the sum of the components of $\sigma(\mathbf{z})$ is 1;
- This implies that all the components of $\sigma(\mathbf{z})$ take values between 0 and 1;
- They indeed represent probabilities:

$$\sigma(\mathbf{z})_h = p_h = p\left(y = h | \mathbf{x}; W, \mathbf{b}\right).$$

# COMMENTS

- Softmax returns independent probabilities;
- Softmax returns in fact a global distribution where the sum of all the probabilities equal to 1;
- This is not the case for One-vs-All;
- The big difference is that Softmax assumes that each example is a member of exactly one class. Some examples, however, can simultaneously be a member of multiple classes (think of an image with both a pear and an apple).
- One-vs-All is reasonable when the total number of classes is small, but becomes inefficient as the number of classes rises.

Notice also that:

- We have now covered the case of multi-class classification;
- With the logistic regression, we have added "non-linearity" to the model because of the sigmoid function;
- This allows us to also threat non-linearly separable classes.

# RECAP: SINGLE-LAYER NN

- We implemented the Adaline algorithm to perform **binary classification**;
- We used the **gradient descent** optimization algorithm to learn the weight and bias coefficients of the model;
- The function to optimize was the MSE (**Mean Squared Error**) loss function;
- The weights were updated by taking a step in the opposite direction of the gradient, with the learning rate $\eta$ as a scale factor;
- We updated the weights simultaneously after each epoch;
- For the final prediction, we introduced a **threshold function**;
- By changing loss function and activation function, we easily obtain the **logistic regression** model: their structure is essentially the same!!!

Both Adaline and logistic regression can be referred to as **single-layer neural network**.

- We obtained multi-class classification by rewriting the problem in **matrix form**.

# OUTLINE NEXT LECTURES

1. Lecture 8: *On robustness, hallucinations and safety of AI algorithms* with V. Antun (UiO);
2. Lecture 9:
   - Discussion Exercise 2;
   - Discussion Exercise 3;
3. Lecture 10:
   - Neural Networks: backpropagation and convergence;
   - Guest lecture with L. Pararasasingam (Statkraft);
4. Lecture 11:
   - Neural Networks: Stochatic Gradient Descent;
   - Discussion Exercise 4.
5. Lecture 12:
   - TBD.