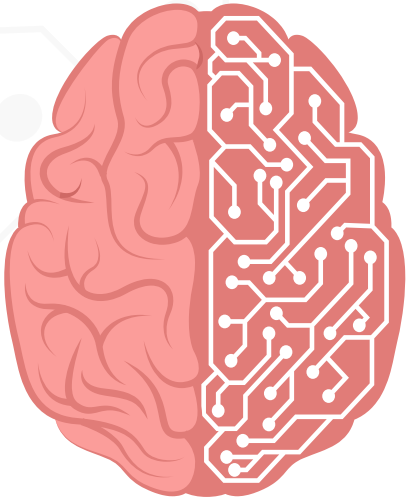


Neural Networks

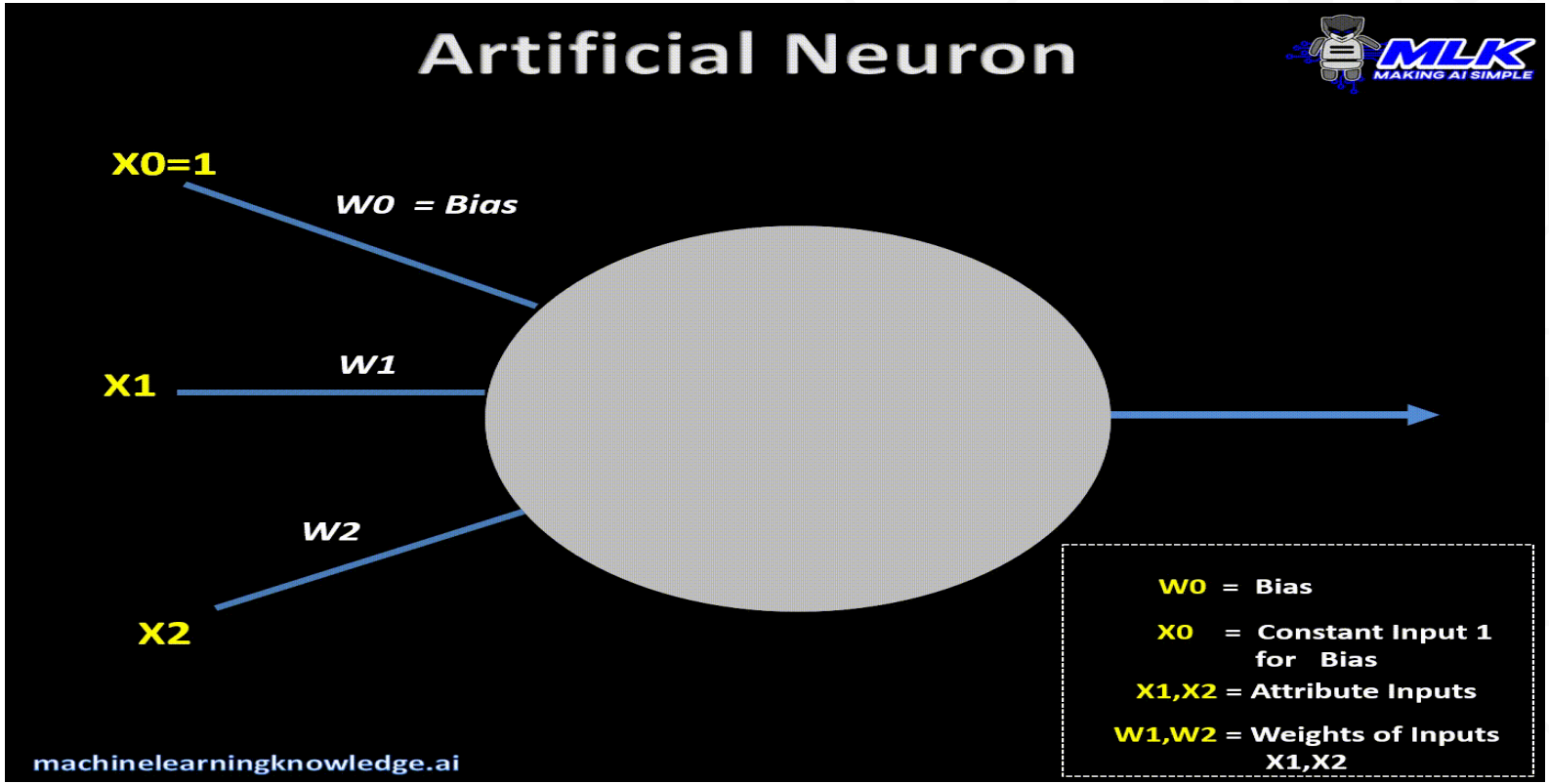
2022-2023

LAB 3

Agenda

1. Single Layer Perceptron
2. Decision Boundary
3. Bias
4. Task 1 Description

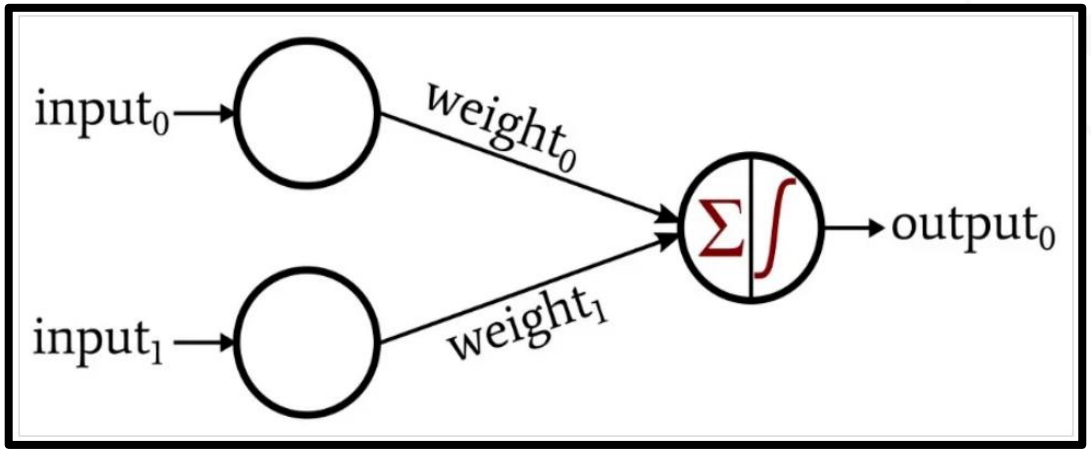
Neural Networks



Single layer Perceptron

- Any neural network consists of interconnected nodes arranged in layers. The nodes in the input layer distribute data, and the nodes in other layers perform summation and then apply an activation function.

- The connections between these nodes are weighted, meaning that each connection multiplies the transferred datum by a scalar value.



Single layer Perceptron

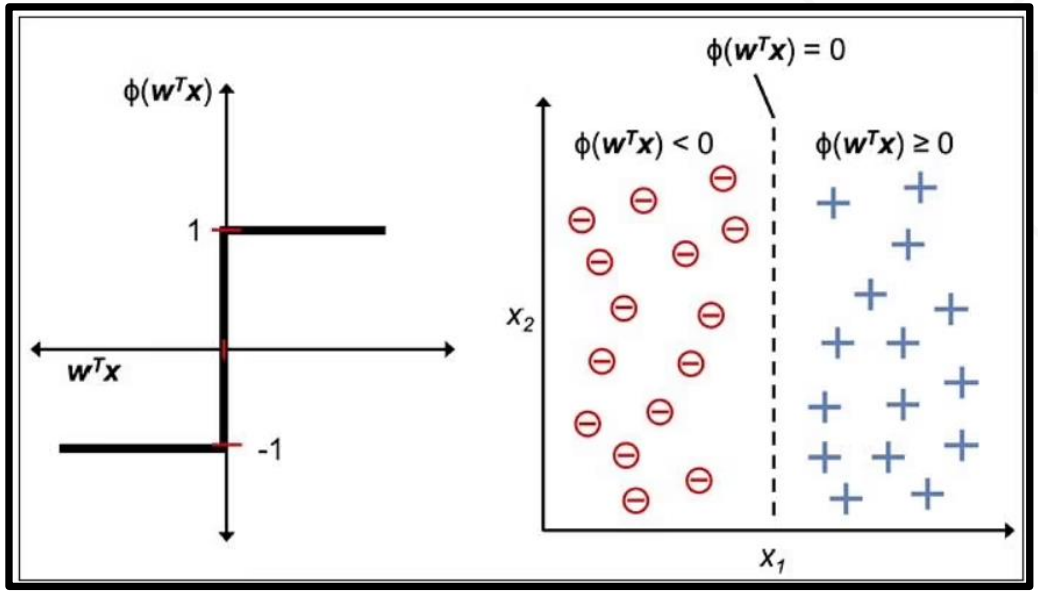
1. Start with a randomly chosen weight vector w(0)
2. Repeat
3. For each training vector pair (x_i, t_i)
4. evaluate the output y_i when x_i is the input according to
   y_i = signum $[\![[w(i)]\!]$^T. $x(i)$+b]
5. if $y\_i \neq t\_i$ then
   calculate loss L = (t_i - y_i)
   form a new weight vector wi+1 according to
   $w\_(i+1) = w\_i + \eta.(t\_i - y\_i).x\_i$
   else
   do nothing

   end if

   End for
6. Until the stopping criterion (convergence) is reached

Single layer Perceptron – Decision Boundary

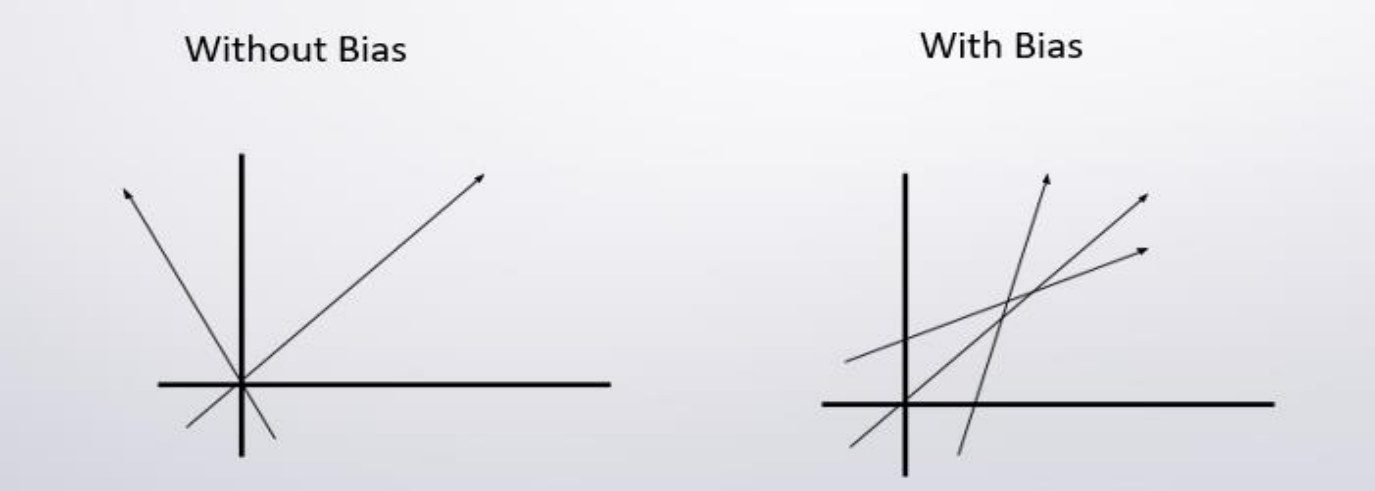
- A decision boundary is a hyperplane that separates the data points into specific classes, where the algorithm switches from one class to another. On one side a decision boundary, a datapoints is more likely to be called as class A — on the other side of the boundary, it's more likely to be called as class B.



Single layer Perceptron – Bias

- Bias is simply a constant value (or a constant vector) that is added to the product of inputs and weights. Bias is utilized to offset the result.

- The bias is used to shift the result of activation function towards the positive or negative side.

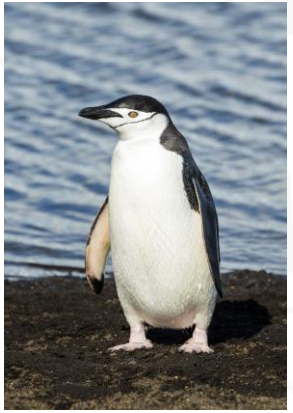


Task 1 – Dataset (Penguins)

- The data set consists of 50 samples from each of three species of Penguins (Adelie, Gentoo and Chinstrap).
- Five features were measured from each sample: bill_length, bill_depth, flipper_length, gender and body_mass, *(in millimeter)*.

Adelie

Gentoo

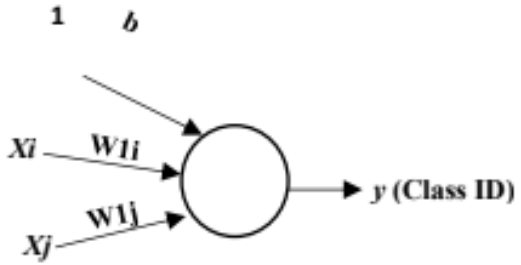


Chinstrap

Task 1 - GUI

1. **User Input:**
- Select two features
- Select two classes (C1 & C2 or C1 & C3 or C2 & C3)
- Enter learning rate (eta)
- Enter number of epochs (m)
- Add bias or not (Checkbox)

2. **Initialization:**
- Number of features = 2.
- Number of classes = 2.
- Weights + Bias = small random numbers



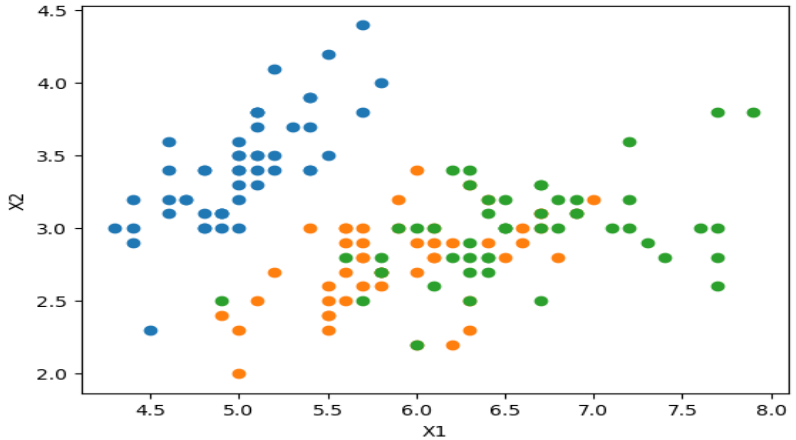
3. **Classification:**
- Sample (single sample to be classified).

Task 1 - Description

1. Visualize Penguins dataset

- Penguins' dataset contains 150 samples (50 samples/class). Each sample consists of 5 features.
- The first part of task(1) is analyzing the data and making a simple report to know the linear/non-linear separable features.

HINT: Drawing all possible combinations of features like (X1, X2), (X1, X3), (X1, X4), (X2, X3), (X2, X4), etc as shown in the following figure and determine which features are discriminative between which classes



Task 1 - Description

2. Implement the Perceptron learning algorithm

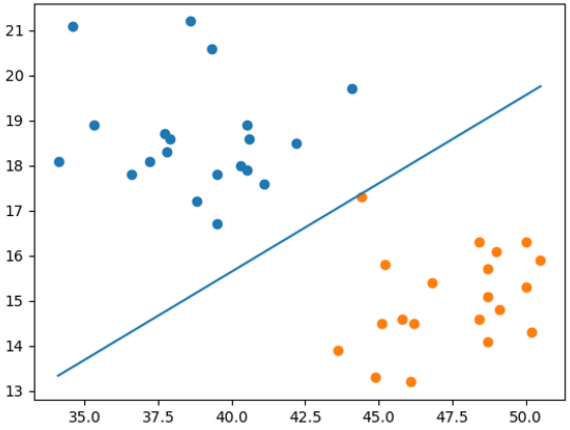
- Single layer neural networks which can be able to classify a stream of input data to one of a set of predefined classes.

- Use the penguins data in both your training and testing processes. (Each class has 50 samples: train NN with 30 non-repeated samples randomly selected, and test it with the remaining 20 samples)

Task 1 - Description

3. After training

- Draw a line that can discriminate between the two learned classes. You should also scatter the points of both classes to visualize the behavior of the line.



- Test the classifier with the remaining 20 samples of each selected classes and find confusion matrix and compute overall accuracy.

Task 1 - Workflow

- **Training Phase: (repeat the following *m* epochs)**

  Assuming that we have ***n*** training samples $\{sample_i : i = 1 \rightarrow n\}$

  - Fetch features (***x***) of $sample_i$, and its desired output (***d***)
  - Calculate the net value (***v***),
  - Calculate actual output (***y***) using ***signum*** activation function,
  - Calculate the ***error*** = ***d*** – ***y***,
  - Update the weights (***new weights*** = ***old weights*** + ***eta*** * ***error*** * ***x***), note: old weights is $\begin{bmatrix} b \\ W1i \\ W1j \end{bmatrix}$

- **Draw line:** line equation is ***W1i*** * ***Xi*** + ***W1j*** * ***Xj*** + ***b*** = ***0***

Task 1 - Workflow

- **Testing Phase:**
  1. Given a sample x
  2. Calculate the net value (v),
  3. Calculate actual output (y) using ***signum*** activation function,
  4. **Output:** y (Class ID).

- **Evaluation:** build the confusion matrix and overall accuracy

Task 1 - Notes

1. You will be asked to deliver a .rar folder containing all your code files (.py), dataset and the visualization and analysis report.

2. In the report you should have screenshots/plots of the visualizations and a written analysis of what you understood from each of these visualizations and how the features discriminate or not between classes. You should have a plot and analysis for each combination (10 combinations)

3. At the end of the report, you should mention which features achieved the highest accuracy after running your algorithm.

4. **You should not drop any row from the dataset.**

5. Using scikit-learn metrics library or any similar built-in function for the confusion matrix is not allowed.

Task 1 - Notes

1. Lab 3 including Task 1 description will be available **Thursday 27/10/2022**.
   Task Deadline: 6/11/2022 11:59 PM
2. Please try to write well designed code.
3. Separate the logic (generation, loading, and classification) from the UI.
4. Writing well-documented, readable, maintainable, and extensible code is an extremely important skill you must master before graduating. So, don't be lazy!
5. Cheating will not be tolerated.

Thank you