PERCEPTRON AND LOGISTIC REGRESSION

A presentation by **Hans Emmanuel Hernandez** (2020-11387)

OBJECTIVES

In this activity, we construct and train a Perceptron on various blobs with various features. Specifically, we have the following objectives:

- Construct a perceptron
- Discuss the principles of the perceptron model (in the code)
- Perform model evaluation after testing
- Describe what happens to the system when non-linearity is introduced
- Validate the training accuracy for each trial

01 RESULTS AND ANALYSIS

PERCEPTRON PRINCIPLES

In [99]:
```python
# Create perceptron class
class Perceptron:
    # Adds a bias term to account for the offset in the decision boundary
    def add_bias(self,X):
        return np.insert(X, 0, np.ones(X.shape[0]), axis=1)
    # Apply heaviside function after dotting of weights
    # Returns the predicted class based on the value
    # Returns 1 if >= 0, otherwise, it returns 0
    def predict(self, X):
        return (np.dot(self.add_bias(X), self.weights) >= 0) * 1
    # Initialize the weight vector and bias with very small numbers or zeros
    # Iterates for 1000 steps to update the weights based on the prediction error
    # The dot product of the error with the input data is multiplied by the learning rate (lr)
    # This value is added back to the current or 'old' weights
    def fit(self, X, y, itr=1000, lr=0.0001):
        X = self.add_bias(X)
        self.weights = np.zeros(X.shape[1])
        for _ in range(itr):
            self.weights += lr * np.dot((y - self.predict(X[:,1:])), X)
```

Figure 1. Code snippet for constructed perceptron class.

TRIAL 1: LINEARLY SEPARABLE DATASET

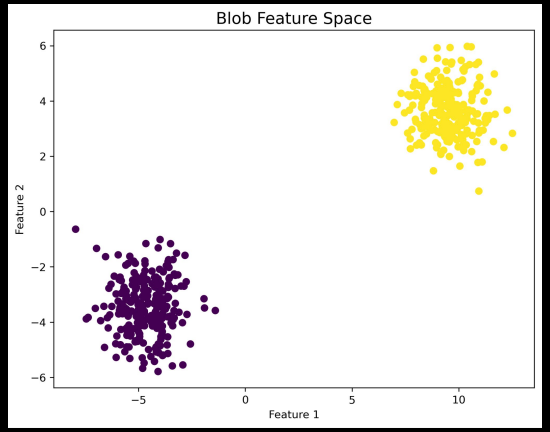


Figure 2. Feature space of a linearly separable dataset.

For our feature space, we generate 2 sets of 250 two-dimensional blobs, each with two arbitrary features. These are first scatter plotted into the feature space. To fit the decision hyperplane, we first train then evaluate our perceptron model.

TRIAL 1: MODEL EVALUATION

```
In [168]: # We separate our data into features (X) and labels (y)
          # X_train and y_train are used to train and fit the model
          # X_test and y_test are used to test and validate the predicted outputs of the trained model
          X_train, X_test, y_train, y_test = train_test_split(X, y)
          p = Perceptron()
          p.fit(X_train, y_train)
          # Training the perceptron
          y_p_train = p.predict(X_train)
          # Validating output and labels
          y_p_test = p.predict(X_test)
          # Print accuracy results
          print("Training accuracy: {} %".format(100 - np.mean(np.abs(y_p_train - y_train)) * 100))
          print("Test accuracy: {} %".format(100 - np.mean(np.abs(y_p_test - y_test)) * 100))

          Training accuracy: 100.0 %
          Test accuracy: 100.0 %
```

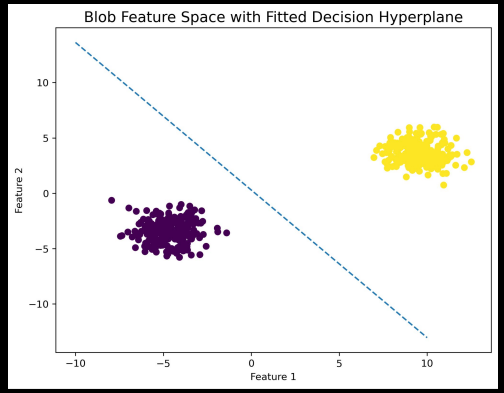
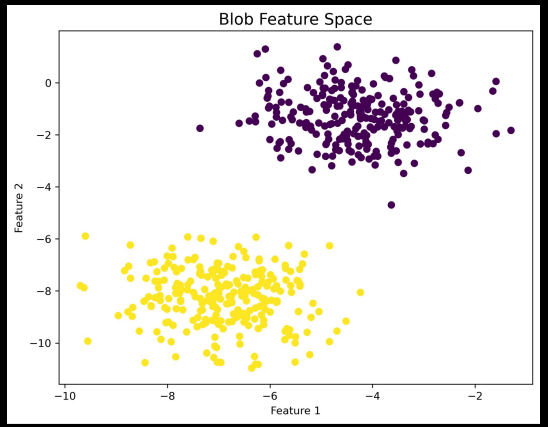


Figure 3. Model evaluation of the trained perceptron model for a linearly separable dataset and the corresponding fitted decision hyperplane.

We start by splitting our dataset into a training set and a testing set, each with respective sets of features and labels. We train our perceptron model and validate the resulting output and labels by testing its accuracy. For the training accuracy, **the algorithm compares the predicted labels from the true labels and generalizes how accurately the model was able to learn the patterns in the training set.** However, for the testing accuracy, **the model is tested on unseen data to estimate how well it is going to perform under new circumstances.** As we can see on the result on the right, **the model was indeed able to fit a decision hyperplane to separate the two sets of blobs with 100% accuracy.**

TRIAL 2: PARTIALLY LINEARLY SEPARABLE DATASET



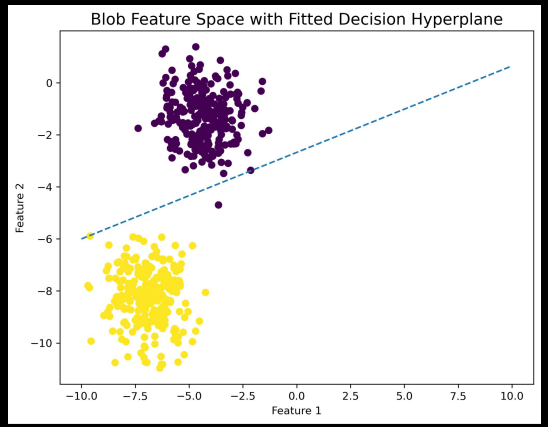


Figure 4. Trained perceptron on imperfectly linear dataset. **Training accuracy: 100.0%, Testing accuracy: 99.2%.**

As we can see in Fig. 2, the trained perceptron was able to fit a decision hyperplane up to a 99.2% testing accuracy. **This is usually the case for when the dataset is only partially linearly separable.** As a result, this induces a ‘noise’ to the decision hyperplane which offsets its resulting accuracy, **effectively misclassifying some of the blobs as we can see on the right.**

TRIAL 3: NON-LINEARLY SEPARABLE DATASET

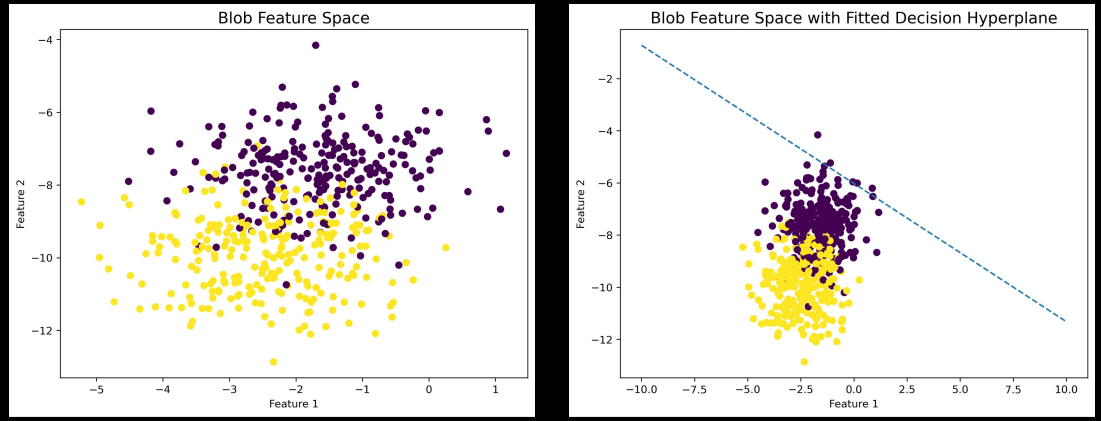


Figure 5. Trained perceptron on non-linearly separable dataset. **Training accuracy: 50.667%, Testing accuracy: 52.0%.**

As we can see in Fig. 5, the trained perceptron was able to fit a decision hyperplane up to a testing accuracy of 52.0%. This is problematic since it raises an issue of non-linear separability to the system. **When we are dealing with non-linearly separable data, the perceptron will most likely misclassify a majority of the dataset especially those located in the overlapping regions which exhibit non-linearity.** Additionally, the perceptron will also most likely struggle to converge which significantly **lowers the learning rate.**

02 REFLECTION

REFLECTION

Overall, this was a great activity!

I enjoyed constructing and modifying perceptron models since this is my first time in really coding them manually. It was interesting to apply them onto variable datasets especially onto the non-linearly separable ones because it looked like it was really struggling LOL. I really enjoyed evaluating the models as well since it future-proofs the machine to some degree if you choose the best one.

Besides that, I believe that my results are accurate and stayed true to the topic, with additional cross references and analyses. I also cross-validated my results with my peers. However, if I was given more time, I would like to train a perceptron on climate data and space data.

SELF-GRADE

Technical Correctness: 35/35

I believe that my results are correct through math, research, and through validation with my peers and with my instructors.

Quality of Presentation: 35/35

I believe that the quality of my powerpoint is up to par with the course expectations. I constructed the figures as instructed, and exported my data accordingly.

Self-Reflection: 30/30

I believe that I have acknowledged and reflected upon the activity well enough. I also have complete citation on the next slide.

Initiative: 10/10

I went above and beyond with my data presentation, and included extra analyses for the activities.

REFERENCES

[1] Banoula, M. (2023). What is Perceptron: A Beginners Guide for Perceptron. *Simplilearn.com*. https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron#:~:text=A%20Perceptron%20is%20a%20neural,value%20%E2%80%9Df(x).

[2] *Perceptron in Machine Learning - Javatpoint*. (n.d.). www.javatpoint.com. https://www.javatpoint.com/perceptron-in-machine-learning

[3] Rosebrock, A. (2021, May 12). *Implementing the Perceptron Neural Network with Python - PyImageSearch*. PyImageSearch. https://pyimagesearch.com/2021/05/06/implementing-the-perceptron-neural-network-with-python/

[4] Verma, S. (2022, January 7). Implementing the Perceptron Algorithm in Python - Towards Data Science. *Medium*. https://towardsdatascience.com/perceptron-algorithm-in-python-f3ac89d2e537