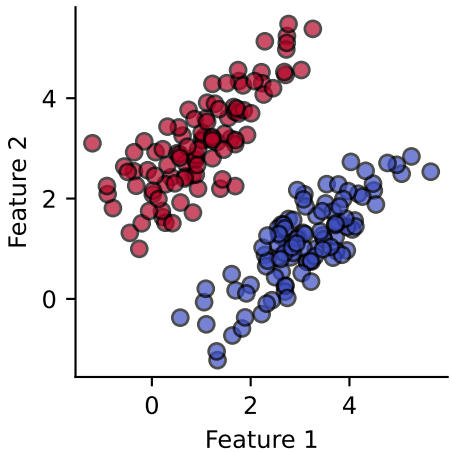
Exercise Sheet #6

Problem 1 (*Linear Classifier*)

A neuron is a linear classifying unit. To put that to the test, implement a single layer perceptron that performs a linear classification task.

(a) **Data generation:** First we need some data. Implement a function that generates $N \in \mathbb{N}$ samples with two features each in equal parts from two different multivariate normal distributions with centers $\vec{\mu}_i$ and covariances $\mathbf{\Sigma}_i$, $i = 1, 2$. The result should look something like this:



Hint: `numpy.random.multivariate_normal` is helpful.

(b) **Model:** Implement a single layer perceptron as a PyTorch module with the forward pass

$$f(\vec{x}) = \sigma(\vec{w} \cdot \vec{x} + b), \qquad \sigma(x) = \frac{1}{1 + \mathrm{e}^{-x}} \ .$$

(c) **Training:** Split the generated data from part (a) into 80% training dataset and 20% test dataset, ensuring that both clusters are equally represented in the splits. Using a suitable loss function (e.g. `torch.nn.MSELoss`) and a suitable optimizer (e.g. `torch.optim.SGD`), train your model on the training dataset for 1000 epochs.

(d) **Inference:** Using the test dataset, verify that our model learned the classification task and generate a plot visualizing this.

The same classification can be done using a support vector machine (SVM).

(e) Using the data from part (a), solve the classification task using a SVM with linear kernel. Extract the hyperplane and plot the hyperplane alongside the data.
Hint: `sklearn.svm.LinearSVC` is helpful.