Artificial Neural Network 2

Summary

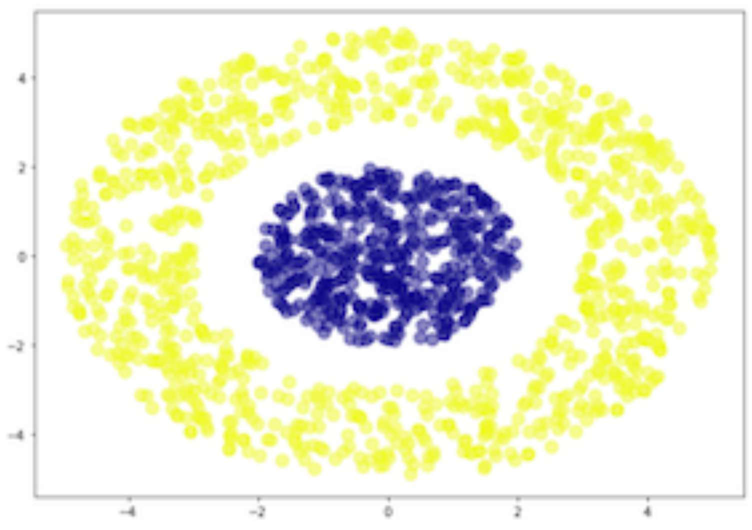
In the first exercise, we write a code to create and train an ANN from scratch. In the second exercise, we use a python library for the same problem.

Exercise 1

Last time you wrote a code for forward pass in an ANN and also for a forward pass and backpropagation in a single neuron.

Here we want to write a code to create and train a simple 2-layer ANN from scratch. Our ANN consists of 2 nodes in the input layer, 4 nodes in a hidden layer, and 1 node in the output layer. The activation functions for both hidden and output layers are sigmoid.

Create a synthetic training dataset with two classes, e.g., similar to the following figure (if you want, you may use syn3 from example_dataset). Likewise, create a test dataset.



Define your ANN:

- Define a matrix containing the weights for connections from input layer to the hidden layer.
- Define a vector containing the biases to all the hidden nodes.
- Define a matrix containing the weights for connections from hidden layer to the output layer.
- Define a vector containing the biases to all the output nodes.
- Randomly initialize the weights (e.g., you may sample from $\mathcal{N}(0, 0.3)$ for weights and set all the biases to zero).

Train your ANN:

- Set a very small learning rate, e.g., 0.0001.
- We want to use a **Batch Gradient Descent** (where batch size = whole training dataset) for optimization of the parameters (weights and biases).
- For that, we repeat the following steps in a loop (for a maximum number of epochs, e.g., 1000 times):
  - **Forward pass**: use all the training data points and calculate all the predicted outputs:
    - use the data to compute the activations of the hidden layer.
    - then use the activations of the hidden layer to compute the network output.
  - compute the values of the **loss function** for all the data points.
  - **Back-propagation (backward pass)**: For all the data points, we need to compute the values of the partial derivatives of the loss function with respect to all the parameters (weights and biases), first for the output layer, and then for the hidden layer.
    - For that, you need to take derivatives in steps backward in the network. Take a look at the lecture notes (a simple network with one hidden layer for binary classification) and figure out the steps backward.
    - Note that you have all the training data packed in a matrix. Try to find a way to implement the backpropagation only with matrix operationns (without a loop going through the data points).
  - Update all the parameters (weights and biases) in the direction of gradient descent.

Use your trained ANN and predict the outputs for your test data.

Exercise 2

Use a python library (e.g., pyTorch or Keras) and create, train, and test an ANN as above. Feel free to reuse and modify the code in the simpleANNExample_pytorch.ipynb.

Compare the results of this ANN with the one that you created in the above exercise.