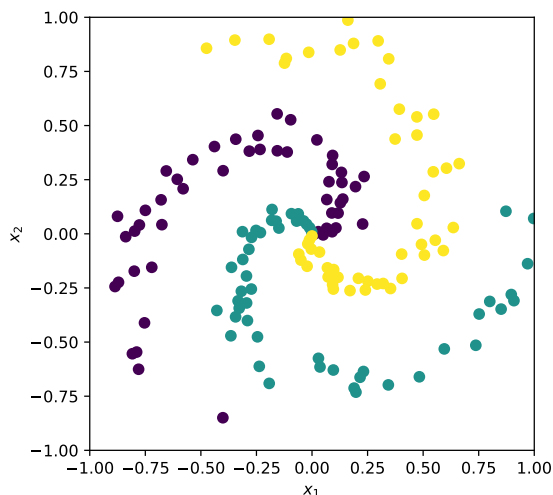


Tutorial 2: **Feedforward neural networks in PyTorch**

March 7, 2023

The objective of this tutorial is to become comfortable with using the Python package PyTorch to create and train a simple feedforward neural network for supervised learning.

You will use and modify the IPython notebook `tutorial2_spirals.ipynb`. The first section of this code generates a random dataset of two-dimensional points with K branches. For example, when $K=3$ this dataset might look as follows:



For each datapoint $\mathbf{x} = (x_1, x_2)$, the label is the branch index such that $y = 0, 1$ or 2 for the example above. Our goal is to implement a neural network capable of classifying the branches.

This network will compare its output with labels in the so-called *one-hot encoding*. For a given label $y=k$, the corresponding one-hot encoding is a K -dimensional vector with all entries zero except for the k^{th} entry (which has value 1). So when $K=3$ the one-hot encodings for the labels are

$$0 \rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad 1 \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad 2 \rightarrow \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

The code first defines the structure of the neural network and then uses the dataset to train this network. The code can generate two files: `spiral_data.pdf` (a plot of the dataset such as the one above) and `spiral_results.pdf` (which displays three plots illustrating the results of training).

1 Neural network architecture, activation functions, cost functions and learning rate

Use the first part of the tutorial to do as many parts of this problem as you can. After 45 minutes, Lauren will explain the solution to part c), which will be needed to proceed to Problem 2.

- a) Run the code and look at how it attempts to classify the two-dimensional space. You should find that the resulting classifier separates the two-dimensional space using lines, and thus does a poor job of representing the data.
- b) Look through the section of code labelled “Define the network architecture and training hyperparameters”. On paper, draw the neural network corresponding to the one in the code for the case of K branches. Pay particular attention to the number of neurons in each layer.
- c) Add in a hidden layer with 4 neurons and study how this hidden layer changes the output. On paper, draw the neural network in this case.
- d) Replace the sigmoid activation function on the first layer with a rectified linear unit (ReLU), and the one on the output layer with a softmax function. Study how the choices of activation functions change the output.
- e) Change the cost function so that it is computed using the cross-entropy instead of the mean-squared error (MSE), and study how the choice of cost function changes the output.
- f) Study the effects of increasing and decreasing the `learning_rate` hyperparameter. Examine these effects using both the cross-entropy and mean-squared error cost functions.
- g) Explain why the K -dimensional one-hot encoding is useful. What do you think would happen if you used a one-dimensional label (such that $y = 0, 1, \dots, K - 1$ or K) instead?

2 Group work

For this part of the tutorial, you will work in groups to explore the capabilities of a feedforward neural network with one hidden layer. Each group will plot the accuracy of the neural network as a function of some quantity Q (a property of the network or the data).

For all plots, the accuracy should be measured once you are convinced that the network has converged, which means that you may need to adjust the `N_epochs` parameter in order to run the code longer. In some cases, you may also find it useful to adjust the learning rate as Q is varied (in particular if you find that your network is getting stuck or if training is very slow for certain values of Q).

At the end of the tutorial, send your most interesting results (as well as a list of your group members) to Lauren and they will be discussed during the next lecture. Each group’s task is to plot the accuracy of the neural network as a function of some quantity Q , where

- **Group 1:** Q = the number of neurons in the hidden layer
- **Group 2:** Q = `mag_noise` (the magnitude of noise in the data)
- **Group 3:** Q = K (the number of different labels)