Task 1A – Resources

Build a Fully Connected 2-Layer Neural Network to Classify Digits

This document covers resources for Task 1A.

How to use these Resources ?

Let's cover the Do's and Don'ts (but mostly Don'ts). First things first, don't be overwhelmed by the quantity of resources. Don't try to read all of them and then do the task in one go (unless it works very well for you). For all other cases, ***Only chew how much you can eat!*** Understand one thing, implement it, test it rigorously and move on. You'll discover that you don't need all the resources to complete the task. May be you know some of the stuff before hand, may be some resources were provided for the sake of exhaustiveness. You may feel you don't see the complete picture since you are only looking at one part of it. For this just learn overall concepts or theory in brief. That way you may not feel left out. And then target one part of it very well.

Don't over complicate things unnecessarily. We already took care of it for you. Don't deep dive into math unnecessarily. You may not require it. We have tailored the tasks in such a way that by the end of it you will definitely know enough math behind basics of machine learning and neural networks in particular. So, if you are able to do tasks by yourself, you know enough of it already.

We hope we have bolstered you enough. So lets do the easy part now.

Resources:

Neural Networks in brief

1. Neural networks with some maths. Great playlist to learn what is a neural network and why it works (with a little bit of maths)? **(recommended)** (link)
2. Neural networks overview. Good if you need a refresher and don't have the patience to watch complete playlist, even though we'll recommend you watch it later **(recommended)** (link)

By now, we suppose you have a brief understanding of neural networks and different things that they are composed from. We can start implementing each of them one by one.

Implementation Resources

Data loading and pre-processing

1. PyTorch tutorial on Data Loading. Covers normalization, image transformation like resizing, centre-cropping, etc. This should be more than enough for transforming images. **Remember to convert image matrices to 1-d vectors, each element acting as a neuron.** (link1) (link2) (link3)
2. PyTorch Tensor API (link1) (link2) (link3)

Architecture decisions and model implementation

1. How to decide on number of input, output and hidden layer neurons? (link) **See this NN demo:** (link)
2. Activation functions **(recommended)** (link)
3. NeuralNetworksAndDeepLearning is in our opinion one such resource that covers neural networks in their entirety including sample code. Use this resource specially to learn backpropagation. **(recommended)** (link)
4. Improving efficiency of neural networks (in depth) **(recommended)** (link)
5. **CS231n** is Stanford's course. For this task you can watch **first 7 lectures** except the one which covers **CNN** (only if you want to learn more theory. We think only by using above resources you should be able to implement the task) (link)
6. Reference for writing code (link1) (link2)

That's it. Checkout what you are supposed do in the code. That's the next step if you are unsure what else we have provided. Also note that you are free to use resources other than the provided ones if you want to. But these are some which we suggest you to use.

…Best Wishes ! …