Mihhail Daniljuk 01.05.2022

EXERCISE 1

Firstly, I thought that writing a neural network for solving XOR problem is like well-known *"Hello-world"* example in usual programming. After that exercise I've changed my opinion. At first, I wanted to write very easy neural net, but after seeing Gabor's example I was really inspired to write neural net with its structure as a hyperparameter, so user can try different number of nodes in each hidden layer and choose the number of these layers. I started to implement the neural net step by step, the first goal was to write a net with one hidden layer and give it only 1 data sample. The second goal was to implement more than one hidden layer, which caused slight changes in backpropagation step, still only 1 input. The final version that I came up with is neural net with customizable structure and Batch Gradient Descent. Last one was a bit hard to implement as it was complicated to follow, what each matrix contains in its rows and columns, where do I need to make matrix multiplication, and where this should be done elementwise. Thanks to this guide, I, literally, had written out all my matrices and checked what each row and column could mean. With all mentioned above I started to test my networks with different structures. Watching at the results (Fig. 1) we can observe that neural net with more hidden layers doesn't always work better, which was surprising for me. Result that was obtained with 2 hidden layers with 10 and 5 neurons draws accurate heatmap, it has some deviations, which could be caused by the size of the training set. As you can see, there is not so much datapoints, that's why heatmap is not perfect XOR.

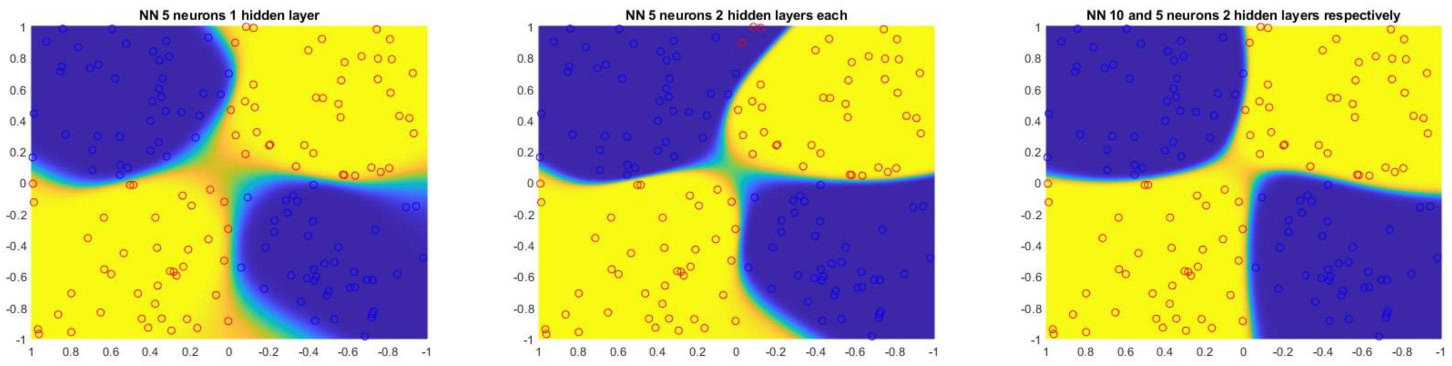

Figure 1. Neural Net XOR problem results with different structures

---

EXERCISE 2

This exercise is the most interesting one in terms of watching at the result. I think it is my second favorite algorithm after decision tree. Doing this exercise, I was struggling with the implementation of 2D neuron grid. Firstly, I tried to use MatLab struct, but later I just decided to use simple matrix and some mathematical operations with indices. Watching at the plots (Fig 2.) I found out that hyperparameters strongly affect the final result. Comparing these two results we can see that on the left picture neural net is very close to get needed shape, whereas second one broke somewhere at 100th iteration, and as a result it failed to get right shape. The only parameters I changed in these experiments was the tau values, which are used to decrease learning rate and neighborhood radiuses through the learning. On the left side, tau values were greater 5 times, and with that model could learn slower and more precise, also, it converges nearly 1500 iterations, the right one converges on 400th iteration. I think that proper tuning of hyperparameters could result in very precise spherical shape of NN.

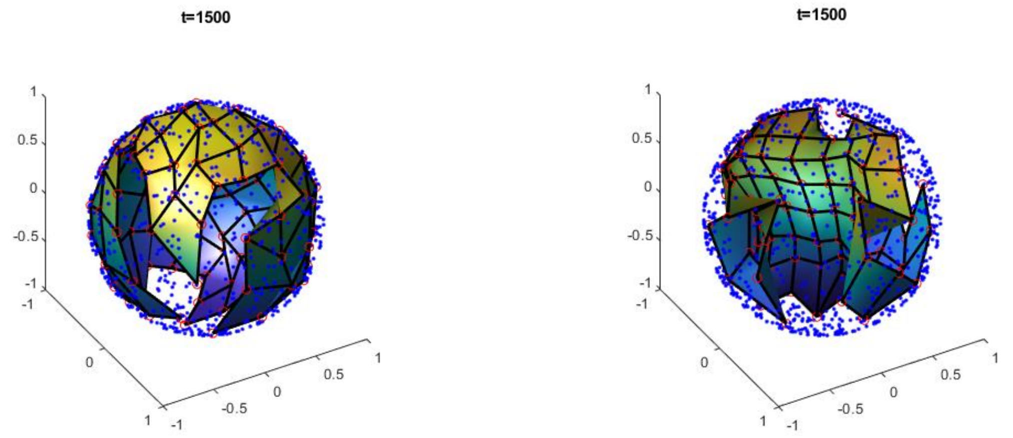

Figure 2. SOM implementation on spherical dataset, different tau values

EXERCISE 3

As we were absolutely free with this exercise in terms of use third-party solutions, I've decided to try something new for me. I've chosen the GAN, which stands for Generative Adversary Network. It has very interesting description and I was very curious about trying it out. Although the training took a lot of time, it was very interesting to watch at generated pictures, and figures showing both *Discriminator* and *Generator* training. To optimize performance of the *Generator*, we have to maximize the loss of the *Discriminator* when give generated data, and to optimize performance of the *Discriminator* we have to minimize the loss of the *Discriminator* when given batches of both real and generated data. Ideally training will produce *Generator* that can generate convincingly realistic images, and *Discriminator* that has learned strong feature representations for this training data. In my case I used Flowers data set, with images of diferent flowers. As you can see from the results below, after 1 hour and 18 minutes of training *Generator* produced images that actually are similar to flowers in some sense. Also we can see that *Discriminator* has score near 0.7 and *Generator* near 0.2 after training.

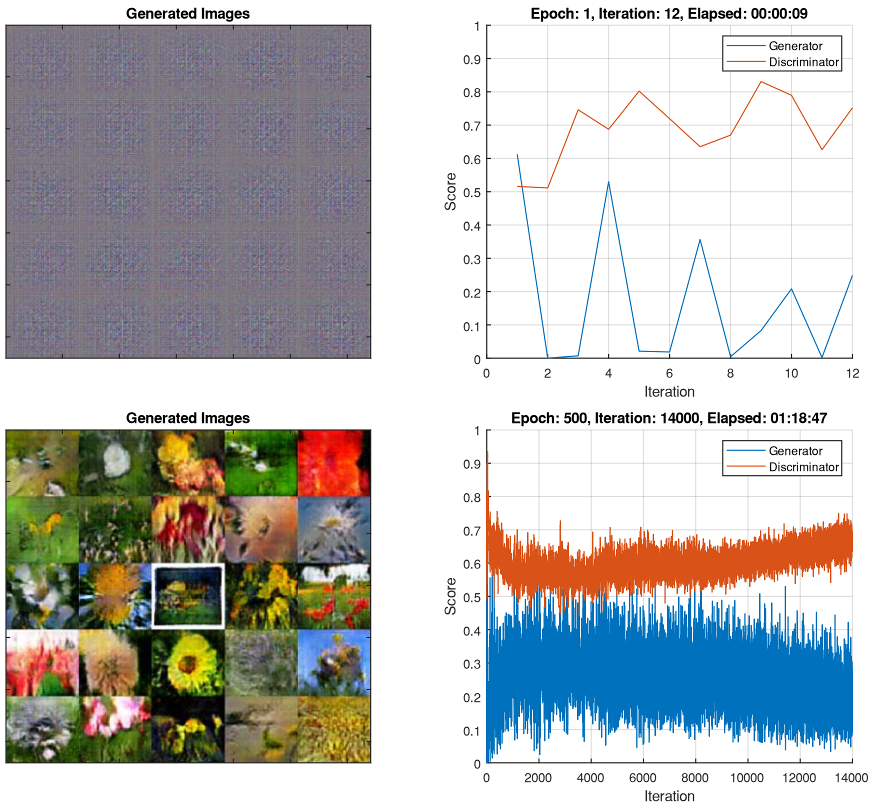

Figure 3. Training of Generative Adversarial Network, starting point – 1st epoch(above) and 500th epoch(below)