TOML2021-22-HW-NN

jorge.garcia.vidal

June 2022

1 Introduction

Download the python code that you will find in the Racó. You will see a simple implementation of a 3 layer FNN performing binary classification. The activation function for the first two layers is a $ReLU()$, while for the output layer we use a $sigmoid()$. The width of the input and hidden layers is H.

The input is a two component vector x. We use in fact three components, and we fix the third component to be equal to 1, i.e. $x[2] = 1$. We do this for including a bias term in the first stage without adding an extra complexity to the code. You can try to include biases also in the other stages to see whether this improve the results.

The output $\hat{y}$ gives us the probability the the input belongs to class 0 or to class 1.

The classification is done on a data set that consists in points of class 0 when their distance to origin is either lower that $radius1$ or larger than $radius2$, and in the points of class 1 that lie in the ring between these two values.

We use gradient descent with back-propagation with momentum, controlled by a value $alpha$. The learning rate is set to be constant, but we could make it adaptative. We initialize the weights with a normal distribution. *We are not using stochastic gradient descent.* Given the simplicity of the model, SGD does not have much sense in this example.

2 Underfitting

We will try to observe a undefitting situation by using a small values of H, meaning that the network is not going to have enough flexibility as to classify the ring-shape dataset. Experiment with values of $T = 640$ and small values of H to observe underfitting of the model.

3 Overfitting

We will try to observe a overfitting situation by using a small values of T. Experiment with values of $H = 20$ and small values of T to observe this overfitting of the model.

4 Finding the right values of H

Experiment with $T = 640$ and different values of H, learning rates and alpha, to obtain a model that classifies well the observed data. Observe the influence of the learning rates and *alpha* in the convergence properties of GD. Take into account that every time you execute the code the data will be different, meaning that in every execution you will observe different descending paths to the solution.

5 Problems with the gradient

Experiment with very large values of H. See what happens with the loss function. Check what happens with the gradients during the backpropagation.

6 (Optative) Try yo implement SGD

Modify the code for introducing an SGD solution method for a suitable value of B.