Deep Learning : mini-project 2

Massonnet Julien
Saini Anand

December 16, 2021

1 Introduction

The goal of this project is to design a small deep learning framework without using "torch.nn", autograd and over advance lybrary.
We want to create a module that given a points in $[0,1]^2$ should determine if the points is in the disk of center $(0.5, 0.5)$ and radius $1/\sqrt{2\pi}$.
The network should have 3 hidden layer of 25 units and can work with 2 possible activation function (ReLU and Tanh). To train the network, we're using stochastic gradient descent.

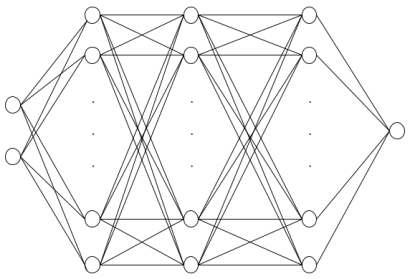

Fig : achitecture of the neural network

2 different Class

To implement such a network we're gonna need a few module, lets run through each one :

2.1 Linear

Linear is a simple module that takes an input of dimension "$in\_features$" and return an output of dimension $out\_features$.
It consist of doing the linear operation $y = x \cdot w^T + b$ where x is the input, y the output, w the weight and b the bias.
Hence it has w ($in\_features \times out\_features$) and b ($out\_features$) as parameters to train.
When doing backwardation, the module needs to compute every partial derivative of the loss for a given gradient ∇y of the output (∇k is the partial derivative of the loss with respect to k).

weight : We get the derivative of the weight as follow :

$$\nabla w = \nabla y \cdot x^T$$

We're doing the outter product between ∇y and x.

bias : We get the derivative of the bias as follow :

$$\nabla b = \nabla y$$

input : We get the derivative of the input as follow :

$$\nabla x = w^T \cdot \nabla y$$

So when doing bacwardation, we should have the gradient of the ouput for each training point as argument, and compute every gradient for each point, then return all the gradient with respect to the input.

2.2 Activation function

We are using two possible activation function : ReLU and Tanh

2.3 Loss function

2.4 Sequential

Sequential is a module that given a list of module will chain them so that when an input is given he will send it to the first module then the output is given to the second module and so on. When we do a backwardation, he will ask the gradients of the last module (given an output) and then will feed these gradients to the module just before and so one (until the first module is reach).

2.5 Module

2.6 optimizer