EE–559: Practical Session 5

François Fleuret

https://fleuret.org/dlc/

February 15, 2018

Draft, do not distribute

Introduction

The objective of this session is to illustrate on a 2D synthetic toy data-set how poorly a naive weight initialization procedure performs when a network has multiple layers of different sizes.

You can get an information sheet about the practical sessions and provided helper functions at

https://fleuret.org/dlc/dlc-info-sheet.tex

1 Toy data-set

Write a function

`generate_disc_set(nb)`

that returns a pair `torch.Tensor`, `torch.LongTensor` of dimensions respectively `nb` $\times 2$ and `nb`, corresponding to the input and target of a toy data-set where the input is uniformly distributed in $[-1, 1] \times [-1, 1]$ and the label is 1 inside the disc of radius $\sqrt{\frac{2}{\pi}}$ and 0 outside.

Create a train and test set of $1,000$ samples, and normalize their mean and variance to 0 and 1.

A simple sanity check is to ensure that the two classes are balanced.

Hint: My version of `generate_disc_set` is 172 characters.

2 Training and test

Write functions

`train_model(model, train_input, train_target)`

`compute_nb_errors(model, data_input, data_target)`

The first should train the model with cross-entropy and 250 epochs of standard sgd with $\eta = 0.1$, and mini-batches of size 100.

The second should also use mini-batches, and return an integer.

Hint: My versions of `train_model` and `compute_nb_errors` are respectively 512 and 457 characters.

3 Models

Write

`create_shallow_model()`

that returns a mlp with 2 input units, a single hidden layer of size 128, and 2 output units, and

`create_deep_model()`

that returns a mlp with 2 input units, hidden layers of sizes respectively $4, 8, 16, 32, 64, 128$, and 2 output units.

Hint: You can use the `nn.Sequential` container to make things simpler. My versions of these two functions are respectively 132 and 355 characters long.

4 Benchmarking

Compute and print the train and test errors of these two models when they are initialized either with the default pytorch rule, or with a normal distribution of standard deviation $10^{-3}, 10^{-2}, 10^{-1}, 1,$ and 10.

The error rate with the shallow network for any initialization should be around 1.5%. It should be around 3% with the deep network using the default rule, and around 50% most of the time with the other initializations.

Hint: My version is 562 characters long.