INSTITUT DU DÉVELOPPEMENT ET DES RESSOURCES EN INFORMATIQUE SCIENTIFIQUE

Hands-on Introduction to Deep Learning Practice Lab 1

The purpose of this practical work is to become familiar with the training of neural networks and their functioning.

The fundamental properties and multilayer perceptron operations are found in most Deep Learning models.

Tensorflow's "Playground" application allows testing, experimenting and assimilating these mechanisms in a sandbox environment without any risk: https://playground.tensorflow.org/

Below, you will find some avenues of exploration as well as guidance on using the features of the Playground.

Avenues of exploration

The instructions given in this section are to be considered as a guide. Do not hesitate to deviate from them and experiment on your own initiative. On the other hand, return to them if necessary.

I. Choice of characteristics

Start with one of the simple data sets (Exclusive Gold, Gaussian) of the typical Classification problem. For each model, repeat the training several times and compare the results.

1. Train a first model. You should be able to get efficient models with a minimum of neurons and features.

2. Reduce the number of hidden layers to 1 with a linear activation and you will obtain the perceptron. Here we train it with the gradient descent method. Try different data with this simple perceptron. Remember to use the available regularization forms.

3. Still with this simple perceptron, try the different features and hyperparameters. You should be able to get good results on several databases.

II. Perceptron and raw data

For this part, limit yourself to the use of the X1 and X2 features. Start with a single linearly activated perceptron on the Circle data.

1. In increasing the number of hidden layers of neurons with linear activation, observe the generated boundaries.

2. Increase also the number of neurons per layer.

3. Try again to find a model by using a nonlinear activation. Look again at the generated boundaries.

4. Test extreme batch sizes and visualize the behavior of the model during training.

5. Test extreme amounts of noise and visualize the behavior of the model during training.

III. Selection of the appropriate model

In this section we focus on the "Spiral" dataset which requires more complex models.

Do not stop training prematurely. Training a model for this task can take several hundred or even thousands of epochs.

1. By applying the different methods seen above, modify and converge this model towards a satisfactory solution.

2. Change the learning rate during training to speed up training.

3. Try to train a model as complex as possible (number of layers, number of features) and observe: the execution time of an epoch, the time needed to reach a satisfactory model.

Description of the graphical user interface

Horizontal banner

On the left side you will find buttons for :

- Start, pause, reset a training
- Run the training epoch by epoch.

> *Feel free to pause a training to change hyperparameters during the training without resetting the model.*

The top horizontal banner also contains options for changing hyperparameters. These include:

- The learning rate
- The activation function
- The type and rate of regularization (not discussed at this time)
- The type of problem to optimize

If you want the model to converge faster, increase the learning rate. Beware, howerve, of the instabilities generated.

Data section

You can choose between several datasets which will require more or less complex models.

You can also change :

- The ratio of data used for the training and the evaluation
- The amount of noise introduced into the data
- The batch size

> *Changing the data type resets the training. However, if you stay with the same data type, you can generate new data without resetting the model. This could be considered as data augmentation.*

Features

By default, the X1 and X2 coordinates are used but you can also add :

- The square of X1 or X2
- The product of X1 by X2
- The sine of X1 or X2

> *Add or remove features as you see fit to solve a problem. This step requires reflection on the data to allow for a selection of relevant features to solve the problem.*

The squares represent the data domain of the block input. The output is represented in colour with negative values in orange and positive values in blue.

> *By hovering the mouse over these squares, you will be able to see the generated boundaries more accurately.*

Hidden Layers

This part allows you to modify the network architecture.

You can control:

- The number of hidden layers (model depth)
- The number of neurons per hidden layer (model width)

> *Modifying the network can allow more complex relationships between data to be determined.*

You can view the weights determined for each neuron by looking at the thickness and colour of each link.

The biases are appear in very small writing in the left-hand corner of each square.

> *By hovering the mouse over these elements, you can see the values of the weights and biases. This information can be used to determine which elements are useful or not in your model. It can also guide you in choosing hyperparameters.*

Output

In this section, you will find various indicators allowing you to qualify and quantify the quality of the model and its learning:

- Test loss, used to evaluate the model
- Train loss, which we seek to minimise by gradient descent
- Two learning curves representing the evolution of the train loss and the test loss.
- A space representing the prediction made by the model according to the input. On top of this are displayed ground-truth from the database.

It is possible to show the test data as well as discretize the output (using a threshold at 0) in order to obtain a classification and to visualize the boundaries used for the classification