TP 6 Data Mining: Standard Neural Network and back propagation

Tuesday 21st May, 2019
To be sent to Frantzeska.Lavda@etu.unige.ch
deadline: Friday 31st May, 2019, 23:59
OBLIGATORY

This TP is obligatory, meaning that if you don't submit it you cannot participate to the exam of the course. As for the previous TPs you have to submit both your code and report.

Description

In this TP you are going to implement a two-layer fully-connected neural network. The network has an input dimension of N, a hidden layer dimension of H, and performs classification over C classes. We train the network with a softmax loss function. The network uses a ReLU nonlinearity activation function after the first fully connected layer.
In other words the architecture:of the network is the following:

input - fully connected layer - ReLU - fully connected layer - softmax

Instructions

Before starting open the NN.py and understand the NN class. You have to fill the missing functions of this file.

- Fill the missing parts of the NN.loss(). This function takes the data and weights and computes the class scores, the loss, and the gradients on the parameters.
  - perform the forward pass
  - perform the backward pass (parts of the backward pass are given)
- Fill the missing part in the NN.train()

  - You have just to to update the parameters of the network

- Implement the NN.predict()
  - This method predicts the labels for data points

Once your code is ready use the TP6NeuralNet.ipynb jupyter notebook to train your network.

- Use $1, 10, 100, 1000$ neurons in the hidden layer (hidden size) and for each case compute the test accuracy and calculate the time it takes to train the network.

- Explain if and how the size of the hidden layer influences the prediction (test accuracy) and the time it takes to train the network.