Practical work 05 – 21/03/2023
Back-Propagation

Objectives

Main objective is to implement the forward and back propagation algorithm for an arbitrary (fully connected) MLP with Softmax. A suitable structure is again provided in form of a Jupyter Notebook that will allow to easily specify MLP models of arbitrary depth and layer sizes.

The implementation should then be used to train MNIST with one or more hidden layers.

Submission

- **Deadline** : Tuesday 28th March, 15pm
- **Format** :
  - Completed Jupyter notebook with the blanks filled in (the sections to be completed marked as usual).
  - Comments and results (plot with learning curve showing the results for different model complexities) either in the notebook or in a pdf-report.

  Submission of all files in a single zip-file using the naming convention (for team of two students #1, #2) :

  `family name_given name #1- family name_given name #2.zip`

Exercise 1 Implement Forward Propagation

Implement the forward propagation through an arbitrary MLP. Use the Jupyter notebook `backprop-stud.ipynb`. Complete the implementation of the method

— propagate

in the classes 'DenseLayer' and the 'SoftmaxLayer'.

Check your implementation with the corresponding unit tests at the end of the notebook.

Exercise 2 Implement Backpropagation

Now, implement backpropagation. Complete the implementation of the method

— backpropagate

in the classes 'DenseLayer' and the 'SoftmaxLayer'.

Again, check your implementation with the corresponding unit tests at the end of the notebook.

Further check the proper implementation of the gradient of the MLP by running the gradient checking (in section 'Test' subsection 'Test analytical value of derivative using backpropagation ...'). This check iterates through all weights, computes the numeric approximation and tests for discrepancies larger than a given limit accuracy ($4.0 \cdot 10^{-7}$). Numeric output is provided only if this threshold is exceeded. Inspect how the numeric approximation of the gradient is computed.

Finally complete the gradient checking with the missing test for the bias values, which should be straight forward given the check for the weights

Exercise 3 Train MNIST

Now, use your implementation to instantiate and train an MLP for MNIST with mini-batch gradient descent.

Study two different architectures :

a) Shallow Network : Single hidden layer layer with 150 units.

b) Deeper Network : Three hidden layers with 250, 150, 50 hidden layers.

What are suitable learning rates, batch sizes, number of epochs ? Describe your findings including the achieved test error rates and the learning curves.