Deep Learning (IST, 2022-23)

Practical 4: Multilayer Perceptron and Backpropagation

André Martins, Andreas Wichert, Luis Sá-Couto, Margarida Campos

Pen-and-Paper Exercises

The following questions should be solved by hand. You can use, of course, tools for auxiliary numerical computations.

Question 1

Consider a network with four layers with the following numbers of units: $4, 4, 3, 3$ - including the input and output layers. Assume all units, except the ones in the output layer, use the hyperbolic tangent activation function.

1. Initialize all connection weights and biases to 0.1. Using the squared error loss do a **stochastic gradient descent** update (with learning rate $\eta = 0.1$) for the training example:

$$\boldsymbol{x} = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}^\top, \quad \boldsymbol{y} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

2. Reusing the computations from the previous exercise do a **gradient descent** update (with learning rate $\eta = 0.1$) for the batch with the training example from the a) and the following:

$$\boldsymbol{x} = \begin{bmatrix} 0 & 0 & 10 & 0 \end{bmatrix}^\top, \quad \boldsymbol{y} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Question 2

Let us repeat the exact same exercise as in 1) but this time we will change:

- The output units have a softmax activation function
- The error function is cross-entropy

1. Initialize all connection weights and biases to 0.1. Using the cross-entropy loss do a **stochastic gradient descent** update (with learning rate $\eta = 0.1$) for the training example:

$$\boldsymbol{x} = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}^\top, \quad \boldsymbol{y} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

2. Reusing the computations from the previous exercise do a **gradient descent** update (with learning rate $\eta = 0.1$) for the batch with the training example from the a) and the following:

$$\boldsymbol{x} = \begin{bmatrix} 0 & 0 & 10 & 0 \end{bmatrix}^\top, \quad \boldsymbol{y} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Programming Exercises

The following exercises should be solved using Python, you can use the corresponding practical's notebook for guidance.

1. Solve **Question 1** from the Pen-and-Paper exercises with Python.
2. Solve **Question 2** from the Pen-and-Paper exercises with Python.
   (a) Create functions to compute foward and backpropragation and gradient updates for an MLP with any number of hidden units.
3. Now it's time to try multi-class logistic regression on real data and see what happens.
   Load the UCI handwritten digits dataset using `scikit-learn`.
   This is a dataset containing 1797 8x8 input images of digits, each corresponding to one out of 10 output classes.
   (a) Randomly split this data into training (80%) and test (20%) partitions.
   (b) Implement a function to do an epoch of MLP training. You can make use of the previously defined functions.
   (c) Consider a MLP with a single hidden layer of 50 units and a learning rate of 0.001. Run 100 epochs of your algorithm on the training data.
   (d) Compute accuracies on both train and test sets
   (e) Run Sklearn's implementation of the MLP and compare the resulting accuracies. enever