Prof. Dr. Volker Roth
volker.roth@unibas.ch

Vitali Nesterov
vitali.nesterov@unibas.ch

Department of Mathematics and Computer Science
Spiegelgasse 1
4051 Basel

Exercise 6

Due date: **Wednesday, April 15$^{\text{th}}$ 2020**

6.1: Training a multilayer neural network

In the original perceptron learning setting, the model consists of a single unit, such that the gradient can be easily computed. However, a multilayer feed-forward neural network is a collection of multiple units, a so called neurons, involving non-linear activation in the hidden layers and a particular network topology. In order to update the model parameters, a gradient is computed in the output layer and is then back-propagated to the hidden layers.

Definitions

- The **potential** of a neuron is given by

$$\phi(\boldsymbol{x};\boldsymbol{w}) \triangleq \boldsymbol{w}^\top \boldsymbol{x},$$

  such that $\boldsymbol{x}$ is an input of a particular neuron and $\boldsymbol{w}$ are the neuron parameters.

- The **output of a neuron** is defined as follows:

$$y \triangleq f(\phi(\boldsymbol{x};\boldsymbol{w})),$$

  such that the potential $\phi(\boldsymbol{x};\boldsymbol{w})$ is mapped by some activation function $f(\cdot)$.

- The **hyperbolic tangent function** is defined as follows:

$$\tanh(x) \triangleq \frac{e^{2x}-1}{e^{2x}+1},$$

  and can be used as an activation function.

- The **loss function** in the output layer is the squared error:

$$J(\boldsymbol{w}) \triangleq \frac{1}{2}||\boldsymbol{t}-\boldsymbol{y}||^2,$$

  with $\boldsymbol{\xi} := \boldsymbol{t}-\boldsymbol{y}$ as a prediction error.

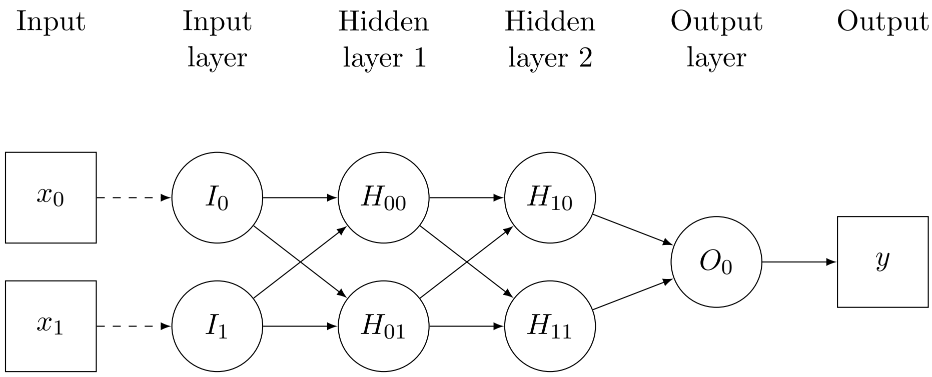

Figure 1: A multilayer perceptron consisting of an input layer, two hidden layers, and an output layer. The input is a vector $\boldsymbol{x} = (x_0, x_1)^\top$ and the output is a scalar y.

Exercise

Determine the gradient $\nabla_{\boldsymbol{w}} J(\boldsymbol{w})$ with respect to the parameters $\boldsymbol{w_i}$ of each i-th neuron in the output layer. Assume the hyperbolic tangent function as the activation function in the hidden layers and a learning rate η. How to compute a gradient in a hidden layer?

6.2: Multilayer feed-forward neural network (toy example)

A multilayer feed-forward neural network is a collection of multiple units with a non-linear activation in the hidden layers and a particular model architecture that forms a directed acyclic graph. In this exercise, the neural network model approximates a function, based on a single-feature input. Implement a simple multilayer perceptron architecture (see Figure 1) to solve the least-squares problem. Use the PyTorch package as specified in the Python script.

- Expand the input by adding $x_0 = 1$ for the bias and store the result as a torch tensor.
- Define the model parameters as differentiable torch tensors.
- Use a PyTorch package for matrix and vector operations.
- In the training loop, compute the squared error between labels and predictions, and update the model parameters.
- Evaluate the regression function and plot the corresponding curve and the observed data points.

Why is linear activation in the hidden layers insufficient for most of the problems?

6.3: Multilayer feed-forward neural network

In the previous exercise, the neural network is initialized and trained manually. In this exercise, the full functionality of PyTorch can be used. Implement a Python program, include PyTorch for implementation of a neural network model to solve the least-squares problem.

Neural network architecture

- A fully connected layer with a single input and 64 outputs including ReLU activation.
- A fully connected layer with 64 inputs and 32 outputs including ReLU activation.
- A fully connected layer with 32 inputs and 16 outputs including ReLU activation.
- An output layer with 16 inputs and a single output including a linear activation.

Exercise

- Implement a custom model class *Net*. Define the layers in the initialization method and the forward propagation in the *forward* method.
- Initialize a PyTorch MSE loss function and an ADAM optimizer.
- In the training loop, compute the mean squared error between labels and predictions and update the model parameters.
- Evaluate the regression function and plot the corresponding curve and the observed data points.