Uppsala University
Department of Information Technology
Division of Systems and Control
NW 2024-12
Last rev. January 14, 2025 by NW

Deep Learning

Instruction to the laboratory work

Lab 1: Linear regression and backpropagation

Language: Python/numpy

Preparation:
Read what is stated in reading instructions below
Solve all preparatory exercises in Section 2

Reading instructions:

- UDL book: Chapter 7.1-7.4
- This lab-pm: Chapter 1-2.

Name		Assistant's comments
Program	Year of reg.	
Date		
Passed prep. ex.	Sign	
Passed lab.	Sign	

Contents

1 Introduction

The goals if this laboratory work is to

- implement a linear classification model from scratch
- understand and implement the equations in backpropagation
- have a clear path forward for solving Hand-in assignment 1

After this lab, you will put the pieces together in Hand-in assignment 1, which has the goal

- implement a fully interconnected multilayer neural network from scratch
- optimize that model using backpropagation together with stochastic gradient descent

2 Preparation exercises

2.1 Linear regression

Consider a regression problem with multivariate input $\mathbf{x} = [x_1, \ldots, x_{D_i}]^\mathsf{T}$ and scalar output $y \in \mathbb{R}$. We want to find a model from the input to the output using a linear regression model

$$\begin{aligned} y &= f[\mathbf{x}, \boldsymbol{\phi}] \\ &= \sum_{j=1}^{D_i} \omega_j x_j + \beta \\ &= \boldsymbol{\omega}^T \mathbf{x} + \beta \end{aligned} \tag{1a}$$

where the weight vector $\boldsymbol{\omega} = [\omega_1, \ldots, \omega_{D_i}]^\mathsf{T}$ and the offset β are the parameters $\boldsymbol{\phi} = \{\boldsymbol{\omega}, \beta\}$ of the model.

Consider a dataset $\{\mathbf{x}_i, y_i\}_{i=1}^I$. The cost L is computed by summing the following loss ℓ_i (mean squared error, MSE) over all training data points

$$\ell_i = (f_i - y_i)^2, \tag{1b}$$

$$L = \frac{1}{I} \sum_{i=1}^{I} \ell_i. \tag{1c}$$

where

$$f_i = f[\mathbf{x}_i, \boldsymbol{\phi}] \tag{1d}$$

$$= \sum_{j=1}^{D_i} \omega_j x_{ij} + \beta \tag{1e}$$

is the output of the model for input $\mathbf{x}_i$ and where x_{ij} is the j^{th} component of $\mathbf{x}_i$.

To train this model with gradient descent, we need access to the gradient of the loss with respect to the model parameters, i.e. the partial derivatives $\frac{\partial \ell_i}{\partial \omega_1}, \ldots, \frac{\partial \ell_i}{\partial \omega_{D_i}}$, and $\frac{\partial \ell_i}{\partial \beta}$, which you will derive below.

Question 2.1: *Given a data point $\mathbf{x}_i, y_i$, based on the model in* (1)*, derive expressions for*

$$\frac{\partial \ell_i}{\partial \beta} \quad \textit{and} \quad \frac{\partial \ell_i}{\partial \omega_j} \quad \textit{expressed in terms of} \quad \frac{\partial \ell_i}{\partial f_i}, \quad \frac{\partial f_i}{\partial \beta}, \quad \textit{and} \quad \frac{\partial f_i}{\partial \omega_j}. \tag{2}$$

Note: The "In terms of" here means that the answer should only include these three terms, no other variables.

Answer:

Question 2.2: *Based on the model in* (1)*, derive expressions for (the above used variables)*

$$\frac{\partial \ell_i}{\partial f_i}, \quad \frac{\partial f_i}{\partial \beta}, \quad \textit{and} \quad \frac{\partial f_i}{\partial \omega_j}. \tag{3}$$

Answer:

2.2 Softmax and cross-entropy

For classification problem with multiple classes $y \in \{1, \ldots, D_o\}$ we typically use a softmax function

$$\text{softmax}_k[\mathbf{f}] = \frac{\exp[f_k]}{\sum_{k'=1}^{D_o} \exp[f_{k'}]} \tag{4}$$

The likelihood that input $\mathbf{x}$ belongs to class y is then

$$Pr(y = k|\mathbf{x}) = \text{softmax}_k\big[\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]\big] \tag{5}$$

where $\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ is a regression model

$$\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}] = \boldsymbol{\Omega}\mathbf{x} + \boldsymbol{\beta}. \tag{6}$$

with the parameters $\boldsymbol{\phi} = \{\boldsymbol{\Omega}, \boldsymbol{\beta}\}$.

The recommended loss function ℓ_i for a multi-class classification problem is the cross-entropy loss, which, for numerical reasons when $z \ll 0$, is computed *together* with the softmax function[1]

$$\begin{aligned}\ell_i &= -\sum_{k=1}^{D_o} \tilde{y}_{ik} \log\left[\text{softmax}_k(\mathbf{f}_i)\right] \\ &= \log\left(\sum_{k=1}^{D_o} e^{f_{ik}}\right) - \sum_{k=1}^{D_o} \widetilde{y}_{ik} f_{ik}\end{aligned} \tag{7}$$

where $\widetilde{y}_{ik}$ is the one-hot encoding of the true label y_i for data point i

$$\widetilde{y}_{ik} = \begin{cases} 1, & \text{if } y_i = k \\ 0, & \text{if } y_i \neq k \end{cases} \qquad \text{for } k = 1, \ldots, D_o$$

and f_{ik} is the k^{th} output of the regression model $\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]$.

Question 2.3: *Given a data point $\mathbf{x}_i, \tilde{y}_i$, based on the loss function in* (7)*, derive expressions for*

$$\frac{\partial \ell_i}{\partial f_{ik}}$$

.

Answer:

[1] Make sure you understand that (7) is the same expression as the terms that are being summed in eq (5.24) in the course book

3 Laboratory exercises

This section contains instructions for the laboratory session. It consist of three notebooks. In the first notebook you will implement and train a linear regression model by using gradient descent only using ‘numpy‘. In the following two notebooks you will implement the equations required for implementing and training a neural network.

3.1 Linear classification and gradient descent

Task 3.1 Download the Jupyter notebook Auto_linear_regression.ipynb and open it. Alternatively, you can open the notebook on Google Colab. Work through the notebook. You can write down your answers to the questions in the notebook in the box below. ○

Answer:

3.2 Backpropagation in toy model

Now we will start looking at the backpropagation algorithm. We will do so by looking at the Toy model presented in Section 7.3 in the UDL book.

Task 3.2 Download the Jupyter notebook 7_1_Backpropagation_in_Toy_Model.ipynb and open it. Alternatively, you can open the notebook on Google Colab.

Work through the notebook. ○

3.3 Backpropagation in neural network model

We will now proceed with with the back-propagation algorithm for a fully connected neural network model.

Task 3.3 Download the Jupyter notebook 7_2_Backpropagation.ipynb and open it. Alternatively, you can open the notebook on Google Colab. Work through the notebook. ○

Task 3.4 Make a copy of the notebook in Task and extend it in the following way

- The code only runs backpropagation algorithm for one data point. Extend it to work with multiple data points. For doing this extension you need to decide which dimension that represents your data points and be consistent with your choice. In machine learning and python it is common to reserve the first dimension for indexing the data points (i.e., one row equals one sample).
- Replace the squared loss and its derivative with softmax output (see Equation 4) and cross-entropy loss with softmax (see Equation 7) and its derivative (see Preparatory Exercise 2.3).

While making the extensions, make sure that the derivatives of weight matrices and biases still match up with the finite difference approximation! ○

Task 3.5 Read through the instructions for Hand-in assignment 1 and identify the steps that need to be taken to complete that assignment based on the code you have created in this lab. List these items and possible questions related to this that you have for the teaching assistant below. ○

Answer:

Task 3.6 (if time permits) If you have time left in the lab, start executing on these steps! ○