WEEKLY EXERCISES

IN5400 / IN9400 — MACHINE LEARNING FOR IMAGE ANALYSIS
DEPARTMENT OF INFORMATICS, UNIVERSITY OF OSLO

Dense neural network classifiers

1 Coding exercise of the week

Work in the Jupyter Notebook file `in5400_w4_exercise_1.ipynb` on how to build, train and validate a dense neural network on the MNIST Fashion dataset using PyTorch.

2 Linear algebra

Consider the arrays

$$a = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \quad b = \begin{pmatrix} 4 \\ 2 \end{pmatrix}$$

$$P = \begin{pmatrix} 3 & 6 \\ 2 & 4 \end{pmatrix}, \quad Q = \begin{pmatrix} 2 & 2 \\ 2 & 4 \end{pmatrix}$$

Compute x in the following cases (if it is not possible, state why).

a

$$x = a^\intercal b$$

b

$$x = Pa$$

c

$$x = PQ$$

d

$$Px = a$$

e

$$Qx = b$$

3 Chain rule

For single-variable, scalar-valued functions $f, g : \mathbb{R} \to \mathbb{R}$, the derivative of the composition $(f \circ g)(x) = f(g(x))$ w.r.t. x is given by the so-called *chain rule* of differentiation

$$\frac{\partial}{\partial x} f(g(x)) = \frac{\partial f}{\partial g} \frac{\partial g}{\partial x}.$$

Compute the derivative $\frac{\partial f}{\partial x}$ on the following expressions.

a

$$f(x) = \sin(x^2)$$

b

$$f(x) = e^{\sin(x^2)}$$

c

In the case where $f : \mathbb{R}^m \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$, and $x \in \mathbb{R}^n$, the derivative of f

$$\begin{aligned} f(g(x)) &= f(g_1(x), \ldots, g_m(x)) \\ &= f(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n)) \end{aligned}$$

w.r.t. one of the components of x, can be given by a generalisation of the above chain rule

$$\frac{\partial f}{\partial x_i} = \sum_{j=1}^{m} \frac{\partial f}{\partial g_j} \frac{\partial g_j}{\partial x_i}.$$

Compute the derivatives $\frac{\partial f}{\partial x_1}$ and $\frac{\partial f}{\partial x_2}$ when

$$\begin{cases} f &= \sin g_1 + g_2^2 \\ g_1 &= x_1 e^{x_2} \\ g_2 &= x_1 + x_2^2. \end{cases}$$

4 Forward propagation

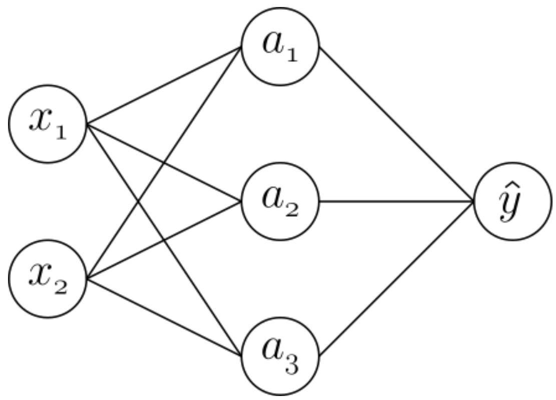

Figure 1: A small dense neural network

Suppose we have a small dense neural network as is shown in fig. 1. The input vector is

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}.$$

In the first layer, we have the following weight parameters $w_{jk}^{[1]}$ and bias parameters $b_k^{[1]}$

$$\begin{pmatrix} w_{11}^{[1]} & w_{12}^{[1]} & w_{13}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} & w_{23}^{[1]} \end{pmatrix} = \begin{pmatrix} 2 & 1 & 3 \\ 2 & -1 & 1 \end{pmatrix}, \quad \begin{pmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}.$$

In the second layer, we have the following weight parameters $w_{j1}^{[2]}$ and bias parameter $b_1^{[2]}$

$$\begin{pmatrix} w_{11}^{[2]} \\ w_{21}^{[2]} \\ w_{31}^{[2]} \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}, \quad \begin{pmatrix} b_1^{[2]} \end{pmatrix} = \begin{pmatrix} 1 \end{pmatrix}.$$

a

Compute the value of the activation in the second layer, $\hat{y}$, when the activation functions in the first and second layer are identity functions.

b

Compute the value of the activation in the second layer, $\hat{y}$, when the activation functions in the first layer are ReLU functions, and in the second layer is the identity function.

5 Cost functions and optimization

Let $\theta^k = [1,3]^\top$ be the value of some parameter $\theta = [\theta_1, \theta_2]^\top$ at iteration k of a gradient descent method. Let the loss function be

$$L(\theta) = 2(\theta_1 - 2)^2 + \theta_2$$

With a step length of 2, find the value of θ^{k+1} when it has been updated with the gradient descent method.

6 Optimizing a convex objective function

Let the loss function L be convex and quadratic

$$L(\theta) = \frac{1}{2}\theta^\top Q\theta - b^\top\theta$$

where $Q \in \mathbb{R}^{n\times n}$ is a symmetric and positive definite matrix, $b \in \mathbb{R}^n$ is a constant vector, and $\theta \in \mathbb{R}^n$ is a vector of parameters.

a

Find an expression for the unique minimizer θ^* of L.

b

Instead of solving the optimization problem analytically, we could to take an iterative approach using gradient descent. Let ∇L_k be the gradient of L w.r.t. θ evaluated at θ_k. For all non-zero ∇L_k, show that the optimal step length at this iteration is given by

$$\lambda_k = \frac{\nabla L_k^\top \nabla L_k}{\nabla L_k^\top Q \nabla L_k}.$$

By optimal we mean the step length that yields the smallest value of L at step $k+1$. Note that if ∇L_k is zero, then $\theta_k = \theta^*$, which means that we are at the unique minimizer of L and should stop iterating.