WEEKLY EXERSICES

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

# Dense neural network classifiers

## 1 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

$$x = a^{\intercal} b$$

$$x = 8$$

**b**

$$x = Pa$$

$$x = \begin{pmatrix} 15 \\ 10 \end{pmatrix}$$

**c**

$$x = PQ$$

$$x = \begin{pmatrix} 18 & 30 \\ 12 & 20 \end{pmatrix}$$

**d**

$$Px = a$$

$P$ is a singular matrix (or non-invertible), and therefore there is no solution $x$ satisfying the equation. This can be checked by verifying that the determinant of $P$ is zero (a matrix is invertible *iff* its determinant is non-zero).

**e**

$$Qx = b$$

$$x = \begin{pmatrix} 3 \\ -1 \end{pmatrix}$$

## 2 Derivatives in higher dimensions

The gradient of a *scalar-valued, multi-variable* function $f : \mathbb{R}^n \to \mathbb{R}$ is given by

$$\nabla_x f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}.$$

For the same function, we can state the *Hessian* matrix of $f$ w.r.t. $x$ as

$$\mathscr{H}_x(f(x)) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 x_1} & \frac{\partial^2 f}{\partial x_1 x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 x_n} \\ \frac{\partial^2 f}{\partial x_2 x_1} & \frac{\partial^2 f}{\partial x_2 x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n x_1} & \frac{\partial^2 f}{\partial x_n x_2} & \cdots & \frac{\partial^2 f}{\partial x_n x_n} \end{pmatrix}.$$

For a *vector-valued*, multi-variable function $g : \mathbb{R}^n \to \mathbb{R}^m$, the *Jacobian* matrix of $g$ w.r.t. $x$ is given by[1]

$$\mathscr{J}_x(g(x)) = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_2}{\partial x_1} & \cdots & \frac{\partial g_m}{\partial x_1} \\ \frac{\partial g_1}{\partial x_2} & \frac{\partial g_2}{\partial x_2} & \cdots & \frac{\partial g_m}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_1}{\partial x_n} & \frac{\partial g_2}{\partial x_n} & \cdots & \frac{\partial g_m}{\partial x_n} \end{pmatrix}.$$

### a

Let $f : \mathbb{R}^n \to \mathbb{R}$ be given by

$$f(x) = x^\top A x + b^\top x + c,$$

where $A \in \mathbb{R}^{n\times n}$, $b, x \in \mathbb{R}^n$, and $c \in \mathbb{R}$. Give the expression of the gradient of $f$ w.r.t. $x$, $\nabla_x f(x)$.

$$\nabla_x f(x) = (A + A^\top)x + b$$

[1] It is also common to define the Jacobian as the transpose version of our definition.

### b

Compute the Hessian matrix of $f$ w.r.t. $x$, $\mathscr{H}_x(f(x))$.

$$\mathscr{H}_x(f(x)) = A + A^\top$$

### c

Compute the Jacobian matrix of the gradient of $f$ w.r.t. $x$, $\mathscr{J}_x(\nabla_x f(x))$.

$$\mathscr{J}_x(\nabla_x f(x)) = A^\top + A$$

### d

Show how, in general, the Hessian matrix relates to the Jacobian matrix.

In general

$$\mathscr{H}_x(f(x)) = \mathscr{J}_x(\nabla_x f(x))$$

## 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

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

Let $u = x^2$, then

$$\begin{aligned}\frac{\partial f}{\partial x} &= \frac{\partial f}{\partial u}\frac{\partial u}{\partial x}\\ &= \cos(u)2x\\ &= 2x\cos(x^2)\end{aligned}$$

**b**

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

Let $u = \sin v$ and $v = x^2$, then

$$\begin{aligned}\frac{\partial f}{\partial x} &= \frac{\partial f}{\partial u}\frac{\partial u}{\partial v}\frac{\partial v}{\partial x}\\ &= e^u\cos(v)2x\\ &= 2x\cos(x^2)e^{\sin(x^2)}\end{aligned}$$

**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 $f$

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

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

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

$$\begin{aligned}\frac{\partial f}{\partial x_1} &= \frac{\partial f}{\partial g_1}\frac{\partial g_1}{\partial x_1} + \frac{\partial f}{\partial g_2}\frac{\partial g_2}{\partial x_1}\\ &= \cos(g_1)e^{x_2} + 2g_2\\ &= e^{x_2}\cos(x_1e^{x_2}) + 2(x_1 + x_2^2)\end{aligned}$$

$$
\begin{aligned}
\frac{\partial f}{\partial x_2} &= \frac{\partial f}{\partial g_1}\frac{\partial g_1}{\partial x_2} + \frac{\partial f}{\partial g_2}\frac{\partial g_2}{\partial x_2} \\
&= \cos(g_1)x_1e^{x_2} + 2g_2 2x_2 \\
&= x_1e^{x_2}\cos(x_1e^{x_2}) + 4x_2(x_1 + x_2^2)
\end{aligned}
$$

## 4 Forward propagation

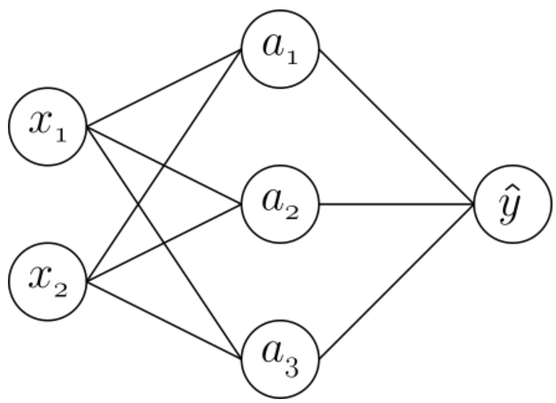


Figure 1: A small dense neural network

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 and bias parameters[2]

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

In the second layer we have the following weight and bias parameters

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

**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.

For the activations in the first layer, with $g$ as the identity function, we have

$$
\begin{aligned}
a_1^1 &= g(w_{11}^1 \cdot x_1 + w_{21}^1 \cdot x_2 + b_1^1) \\
&= g(2 \cdot 1 + 2 \cdot 3 + 1) \\
&= 9 \\
a_2^1 &= g(w_{12}^1 \cdot x_1 + w_{22}^1 \cdot x_2 + b_2^1) \\
&= g(1 \cdot 1 - 1 \cdot 3 + 0) \\
&= -2 \\
a_3^1 &= g(w_{13}^1 \cdot x_1 + w_{23}^1 \cdot x_2 + b_3^1) \\
&= g(3 \cdot 1 + 1 \cdot 3 - 1) \\
&= 5
\end{aligned}
$$

We then get the following activation in the second layer

$$
\begin{aligned}
\hat{y} &= w_{11}^2 \cdot a_1^2 + w_{21}^2 \cdot a_2^1 + w_{31}^2 \cdot a_3^1 + b_1^2 \\
&= 3 \cdot 9 - 1 \cdot 2 + 2 \cdot 5 + 1 \\
&= 36
\end{aligned}
$$

**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.

[2]Note that we drop the superscript bracket notation for layers, $[l]$, for convenience, as there should be no risk of confusion.

For the activations in the first layer, with $g$ as the ReLU function, we have

$$\begin{aligned}
a_1^1 &= g(w_{11}^1 \cdot x_1 + w_{21}^1 \cdot x_2 + b_1^1) \\
&= g(2 \cdot 1 + 2 \cdot 3 + 1) \\
&= 9 \\
a_2^1 &= g(w_{12}^1 \cdot x_1 + w_{22}^1 \cdot x_2 + b_2^1) \\
&= g(1 \cdot 1 - 1 \cdot 3 + 0) \\
&= 0 \\
a_3^1 &= g(w_{13}^1 \cdot x_1 + w_{23}^1 \cdot x_2 + b_3^1) \\
&= g(3 \cdot 1 + 1 \cdot 3 - 1) \\
&= 5
\end{aligned}$$

We then get the following activation in the second layer

$$\begin{aligned}
\hat{y} &= w_{11}^2 \cdot a_1^2 + w_{21}^2 \cdot a_2^1 + w_{31}^2 \cdot a_3^1 + b_1^2 \\
&= 3 \cdot 9 + 1 \cdot 0 + 2 \cdot 5 + 1 \\
&= 38
\end{aligned}$$

## 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 $\theta^{k+1}$ when it has been updated with the gradient descent method.

With gradient descent, the update rule for the parameters $\theta$ is

$$\theta^{k+1} = \theta_k - \lambda \nabla_\theta L(\theta^k)$$

where $\lambda$ is the scalar step length (learning rate). From the given expression, the gradient of $L$ w.r.t. $\theta$ is

$$\nabla_\theta L = \begin{pmatrix} 4(\theta_1 - 2) \\ 1 \end{pmatrix}$$

The updated value of $\theta$ is then

$$\begin{aligned}
\theta^{k+1} &= \theta^k - \lambda \nabla_\theta L(\theta^k) \\
&= \begin{pmatrix} 1 \\ 3 \end{pmatrix} - 2 \begin{pmatrix} 4(1-2) \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} 9 \\ 1 \end{pmatrix}
\end{aligned}$$

## 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 $\theta^*$ of $L$.

The gradient is given by $\nabla_\theta L(\theta) = Q\theta - b$, and setting this equal to zero reveals that $\theta^*$ is the unique solution to the system of equations

$$\theta^* = Q^{-1}b$$

### b

Instead of solving the optimization problem analytically, we want to take an iterative approach using gradient descent. Let $\nabla L_k$ be the gradient of $L$ w.r.t. $\theta$ evaluated at $\theta_k$. 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 $k+1$.

The value of $L$ at step $k+1$ is

$$\begin{aligned} L(\theta_k - \lambda\nabla L_k) &= \frac{1}{2}(\theta_k - \lambda\nabla L_k)^\top Q(\theta_k - \lambda\nabla L_k) - b^\top(\theta_k - \lambda\nabla L_k) \\ &= \frac{1}{2}\theta_k^\top Q\theta_k - \lambda\theta_k^\top Q\nabla L_k + \frac{1}{2}\lambda^2\nabla L_k^\top Q\nabla L_k - b^\top\theta_k + \lambda b^\top\nabla L_k \end{aligned}$$

Differentiating this w.r.t. $\lambda$

$$\begin{aligned} \frac{\mathrm{d}L(\theta_k - \lambda\nabla L_k)}{\mathrm{d}\lambda} &= \lambda\nabla L_k^\top Q\nabla L_k - \theta_k^\top Q\nabla L_k + b^\top\nabla L_k \\ &= \lambda\nabla L_k^\top Q\nabla L_k - (\theta_k^\top Q - b^\top)\nabla L_k \\ &= \lambda\nabla L_k^\top Q\nabla L_k - \nabla L_k^\top\nabla L_k \end{aligned}$$

Setting this equal to zero gives the desired result.