Machine Learning Assignment 2: Neural Networks

Mattia Palmiotto

Question 1 (20 points).

I decompose the loss L into

$$L = \sum_{k=1}^{4} l_k$$
$$l_k = \frac{1}{2}(y_k - t_k)^2$$

It follows that

$$\frac{\partial L}{\partial s} = \sum_{k=1}^{4} \frac{\partial l_k}{\partial s}$$

for any value s. Hence, I compute the gradients of l_k and then sum them to obtain the desired quantities.

I do the calculations using the NumPy library of Python. Fist, I initialize the needed quantities:

```
import numpy as np

x_1 = np.array([0.6,-1.0])
x_2 = np.array([0.8,-1.0])
x_3 = np.array([-0.4,0.9])
x_4 = np.array([0.2,0.0])

X = np.array([x_1,x_2,x_3,x_4])
t = np.array([-0.8, -0.1, 0.9, 0.7])

W1 = np.array([[-0.8, -0.7, 0.6],[-1.0, 0.5, -1.0]])
b1 = np.array([-0.2, -1.0, -0.7])
w2 = np.array([0.1, -1.0, 0.5])
b2 = -0.7
```

```
U1 = X @ W1 + b1

def relu(x):
   return(np.maximum(0,x))

Z1 = relu(U1)
Z1_T = np.transpose(Z1)

Y = w2 @ Z1_T + b2

diff = Y - t

# L = (1/2)*sum(diff**2)

##################################################

# Useful quantities

w2_T = w2.reshape(3,1)

def der_relu(x):
   if x>0:
      return 1
   else:
      return 0

der_relu_array = np.vectorize(der_relu)

Z1_der = der_relu_array(U1)
```

Derivative with respect to $b^{(2)}$

$$\frac{\partial l_k}{\partial b^{(2)}} = \frac{\partial l_k}{\partial y_k}\frac{\partial y_k}{\partial b^{(2)}} = y_k - t_k = \mathbf{w}^{(2)} f(x_k \mathbf{W}^{(1)} + \mathbf{b}^{(1)})^T + b^{(2)} - t_k$$

```
L_wrt_b2 = np.round(sum(diff),5)

print('\nThe derivative of L wrt b2 is\n',L_wrt_b2)

The derivative of L wrt b2 is
-2.732
```

Gradient with respect to $\mathbf{b}^{(1)}$

$$\frac{\partial l_k}{\partial \mathbf{b}^{(1)}} = \frac{\partial l_k}{\partial y_k}\frac{\partial y_k}{\partial z_k^{(1)}}\frac{\partial z_k^{(1)}}{\partial \mathbf{b}^{(1)}} = (\mathbf{w}^{(2)} f(x_k \mathbf{W}^{(1)} + \mathbf{b}^{(1)})^T + b^{(2)} - t_k) \cdot \mathbf{w}^{(2)} f'(x_k \mathbf{W}^{(1)} + \mathbf{b}^{(1)})^T$$

```
lk_wrt_b1 = diff*np.transpose(w2*Z1_der)

L_wrt_b1 = np.round(np.sum(lk_wrt_b1,axis=1),5)

print('\nThe gradient of L wrt b1 is\n',L_wrt_b1)

The gradient of L wrt b1 is
[0.0268 0.     0.134 ]
```

Gradient with respect to $\mathbf{w}^{(2)}$

$$\frac{\partial l_k}{\partial \mathbf{w}^{(2)}} = \frac{\partial l_k}{\partial y_k}\frac{\partial y_k}{\partial \mathbf{w}^{(2)}} = (\mathbf{w}^{(2)} f(x_k \mathbf{W}^{(1)} + \mathbf{b}^{(1)})^T + b^{(2)} - t_k) f(x_k \mathbf{W}^{(1)} + \mathbf{b}^{(1)})^T$$

```
lk_wrt_w2 = diff*Z1_T

L_wrt_w2 = np.round(np.sum(lk_wrt_w2,axis=1),5)

print('\nThe gradient of L wrt w2 is\n',L_wrt_w2)

The gradient of L wrt w2 is
[0.1168 0.     0.1536]
```

Gradient with respect to $\mathbf{W}^{(1)}$

$$\frac{\partial l_k}{\partial w_{mn}^{(1)}} = \frac{\partial l_k}{\partial y_k}\frac{\partial y_k}{\partial z_k^{(1)}}\frac{\partial z_k^{(1)}}{\partial w_{mn}^{(1)}} = (\mathbf{w}^{(2)} f(x_k \mathbf{W}^{(1)} + \mathbf{b}^{(1)})^T + b^{(2)} - t_k) \cdot w_n^{(2)} \cdot f'(x_k \mathbf{W}^{(1)} + \mathbf{b}^{(1)})^T \cdot x_{km}$$

```
l1_wrt_W1 = diff[0] * (np.transpose(w2_T * x_1)*Z1_der[0])
l2_wrt_W1 = diff[1] * (np.transpose(w2_T * x_2)*Z1_der[1])
l3_wrt_W1 = diff[2] * (np.transpose(w2_T * x_3)*Z1_der[2])
l4_wrt_W1 = diff[3] * (np.transpose(w2_T * x_4)*Z1_der[3])

lk_wrt_W1 = np.array([l1_wrt_W1, l2_wrt_W1, l3_wrt_W1, l4_wrt_W1])

L_wrt_W1 = np.round(np.sum(lk_wrt_W1,axis=0),5)
```

```
print('\nThe gradient of L wrt W1 is\n',L_wrt_W1)

The gradient of L wrt W1 is
[[ 0.0122  0.      0.061 ]
[-0.0268  0.     -0.134 ]]
```

Question 2 (15 points).

In the general case:

$$E = \sum_k e_k$$

$$e_k = -t_k \log\left(\frac{\exp(y_k)}{\sum_j \exp(y_j)}\right) = -\left(t_k \log(\exp(y_k)) - t_k \log\left(\sum_j \exp(y_j)\right)\right) =$$

$$= t_k \log\left(\sum_j \exp(y_j)\right) - t_k y_k$$

$$\frac{\partial e_k}{\partial y_i} = \begin{cases} t_k \cdot \frac{1}{\sum_j \exp(y_j)} \cdot \exp(y_i) = \frac{t_k \exp(y_i)}{\sum_j \exp(y_j)} & \text{if } i \neq k \\ \frac{t_k \exp(y_k)}{\sum_j \exp(y_j)} - t_k & \text{if } i = k \end{cases}$$

$$\frac{\partial E}{\partial y_i} = \sum_k \frac{\partial e_k}{\partial y_i} = \left(\sum_k \frac{t_k \exp(y_i)}{\sum_j \exp(y_j)}\right) - t_i = \frac{\exp(y_i)}{\sum_j \exp(y_j)}\left(\sum_k t_k\right) - t_i$$

In the case in which the target is a one-hot vector s.t. $t_h = 1$ and $t_k = 0$ for each $k \neq h$, we have that $e_k = 0$ for each $k \neq h$. Therefore, the desired quantity becomes:

$$\frac{\partial E}{\partial y_i} = \frac{\exp(y_i)}{\sum_j \exp(y_j)} t_h - t_i = \frac{\exp(y_i)}{\sum_j \exp(y_j)} - t_i = \begin{cases} \frac{\exp(y_i)}{\sum_j \exp(y_j)} & \text{if } i \neq h \\ \frac{\exp(y_i)}{\sum_j \exp(y_j)} - 1 & \text{if } i = h \end{cases}$$

Question 3 (15 points).

The function $f(x) = 5x^2 + 2$ is a convex and positive function which corresponds to the equation of a parabola. The vertex of the parabola is the point $(0, 2)$, which is also the minimum of f: this is a standard result of analytic geometry, however it is also easy to check that the first and the second derivatives of f are $10x$ and 10 respectively. In particular, $x = 0$ is a stationary point and the positive second derivative imply that it is a minimum (it also implies that the function is convex). Hence, we want x_n to converge

to 0.
Let x_0 be our starting point. By recursion, we have that, for each $n \geq 1$:

$$x_n = x_{n-1} - 10\eta f'(x_{n-1}) = x_{n-1} - 10\eta x_{n-1} = (1-10\eta)x_{n-1} = (1-10\eta)^n x_0$$

It follows that $\lim_{n\to\infty} x_n = 0$ if and only if $|1-10\eta| < 1$. We want that:

$$\begin{cases} 1-10\eta < 1 & \Leftrightarrow \eta > 0 \\ 1-10\eta > -1 & \Leftrightarrow \eta < 1/5 \end{cases}$$

We obtain that the range of values that η can take so that gradient descent converges to the minimum from any starting point is $(0, 1/5)$.
We also observe that if $\eta = 1/10$, then the convergence is achieved in one single iteration. If $\eta \neq 1/10$, the speed of convergence is higher when the learning rate η is around $1/10$ and lower when it is close to the extremes of the range.