Assignment 2 & 3

Paul Thillen et Louis-Philippe Noël
IFT3395/6390 - Machine learning

November 30, 2017

1 Theoretical part A

1.1

To show:

$$sigmoid(x) = \frac{1}{2}(tanh(\frac{x}{2}) + 1)$$

Which is equivalent to showing:

$$tanh(x) = 2 \cdot sigmoid(2x) - 1$$

We have:

$$\begin{aligned} tanh(x) &= \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^x - \frac{1}{e^x}}{e^x + \frac{1}{e^x}} = \frac{\frac{e^{2x}-1}{e^x}}{\frac{e^{2x}+1}{e^x}} \\ &= \frac{e^{2x} - 1}{e^{2x} + 1} \end{aligned}$$

and:

$$\begin{aligned} sigmoid(x) &= \frac{1}{1 + e^{-x}} = \frac{1}{1 + \frac{1}{e^x}} = \frac{1}{\frac{e^x+1}{e^x}} \\ &= \frac{e^x}{e^x + 1} \end{aligned}$$

consequently:

$$\begin{aligned} 2 \cdot sigmoid(2x) - 1 &= 2 \cdot \frac{e^{2x}}{e^{2x} + 1} - 1 \\ &= \frac{2 \cdot e^{2x}}{e^{2x} + 1} - \frac{e^{2x} + 1}{e^{2x} + 1} \\ &= \frac{e^{2x} - 1}{e^{2x} + 1} = tanh(x) \end{aligned}$$

1.2

To show:

$$ln(sigmoid(x)) = -softplus(-x)$$

We have:

$$ln(sigmoid(x)) = ln(\frac{1}{1+e^{-x}}) = -ln(1+e^{-x}) = -softplus(-x)$$

1.3

To show:

$$sigmoid'(x) = sigmoid(x) \cdot (1 - sigmoid(x))$$

We have:

$$\begin{aligned} sigmoid'(x) &= (\frac{e^x}{1+e^x})' \\ &= \frac{e^x(1+e^x) - e^x \cdot e^x}{(1+e^x)^2} \\ &= \frac{e^x}{1+e^x}(1 - \frac{e^x}{1+e^x}) \\ &= sigmoid(x) \cdot (1 - sigmoid(x)) \end{aligned}$$

1.4

To show:

$$tanh'(x) = 1 - tanh^2(x)$$

We have:

$$\begin{aligned} tanh'(x) &= (\frac{e^x - e^{-x}}{e^x + e^{-x}})' \\ &= \frac{(e^x + e^{-x})^2 - (e^x - e^{-x})^2}{(e^x + e^{-x})^2} \\ &= 1 - \frac{(e^x - e^{-x})^2}{(e^x + e^{-x})^2} \\ &= 1 - tanh^2(x) \end{aligned}$$

1.5 Write sign using only indicator functions

$$sgn(x) = \mathbb{1}_{\mathbb{R}_+}(x) - \mathbb{1}_{\mathbb{R}_-}(x)$$

1.6 Derivative of abs

$$abs(x) = \begin{cases} x, \text{ if } x > 0 \\ 0, \text{ if } x = 0 \\ -x, \text{ if } x < 0 \end{cases}$$

$$abs'(x) = \begin{cases} 1, \text{ if } x > 0 \\ 0, \text{ if } x = 0 \\ -1, \text{ if } x < 0 \end{cases}$$

1.7 Derivative of rect

$$rect(x) = \begin{cases} x, \text{ if } x > 0 \\ 0, \text{ else} \end{cases} = \mathbb{1}_{\{x>0\}}(x) \cdot x$$

$$rect'(x) = \mathbb{1}_{\{x>0\}}(x)$$

1.8 L2 gradient

$$\frac{\partial ||x||_2^2}{\partial x} = \begin{pmatrix} \frac{\partial}{\partial x_1} ||x||_2^2 \\ ... \\ \frac{\partial}{\partial x_d} ||x||_2^2 \end{pmatrix} = \begin{pmatrix} 2x_1 \\ ... \\ 2x_d \end{pmatrix}$$

1.9 L1 gradient

$$\frac{\partial ||x||_1}{\partial x} = \begin{pmatrix} \frac{\partial}{\partial x_1} ||x||_1 \\ ... \\ \frac{\partial}{\partial x_d} ||x||_1 \end{pmatrix} = \begin{pmatrix} abs'(x_1) \\ ... \\ abs'(x_d) \end{pmatrix}$$

2 Theoretical part B

2.1

Dimensions of $W^{(1)}$ and $b^{(1)}$:

$$\begin{aligned} dim(W^{(1)}) &= d_h \times d \\ dim(b^{(1)}) &= d_h \end{aligned}$$

Preactivation vector of neurons of the hidden layer h^a where $w_j^{(1)}$ is the j-th row of $W^{(1)}$.

$$h^a = W^{(1)} \cdot x + b^{(1)}$$
$$h_j^a = w_j^{(1)} \cdot x + b_j^{(1)}$$

Ouput vector of the hidden layer h^s:

$$h^s = rect(h^a)$$
$$h_k^s = max(0, h_k^a)$$

2.2

Dimensions of $W^{(2)}$ and $b^{(2)}$:

$$dim(W^{(2)}) = m \times d_h$$
$$dim(b^{(2)}) = m$$

Preactivation vector of neurons of the output layer o^a where $w_j^{(2)}$ is the j-th row of $W^{(2)}$.

$$o^a = W^{(1)} \cdot h^s + b^{(1)}$$
$$o_j^a = w_j^{(1)} \cdot h^s + b_j^{(1)}$$

2.3

Ouput vector of the output layer o^s:

$$o^s = softmax(o^a)$$
$$o_k^s = \frac{e^{o_k^a}}{\sum\limits_{i=1}^{m} e^{o_i^a}}$$

Since exponentials are always positive and both denominator and numerator are exponentials or sum of exponentials, o_k^s has to be positive too.

If we sum over all k for o_k^s, we receive:

$$\sum_{j=1}^{m} \frac{e^{o_j^a}}{\sum\limits_{i=1}^{m} e^{o_i^a}} = \frac{\sum\limits_{j=1}^{m} e^{o_j^a}}{\sum\limits_{i=1}^{m}} = \frac{\sum\limits_{i=1}^{m} e^{o_i^a}}{\sum\limits_{i=1}^{m} e^{o_i^a}} = 1$$

These two properties are important because o^s is a probability distribution for each possible class.

2.4

Loss function given a probability $o^s_y(x)$ for a single input vector x to be of class y:

$$\begin{aligned} L(x,y) &= -log(o^s_y(x)) \\ &= -log(\frac{e^{o^a_y}}{\sum\limits_{i=1}^{m} e^{o^a_i}}) \\ &= -log(e^{o^a_y}) + log(\sum_{i=1}^{m} e^{o^a_i}) \\ &= -o^a_y + log(\sum_{i=1}^{m} e^{o^a_i}) \end{aligned}$$

2.5

What is $\hat{R}$? For a loss function L and training data D:

$$\hat{R}(L,D) = \frac{1}{|D|}\sum_d L(x^{(d)}, y^{(d)})$$

What is θ?

$$\theta = \{W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}\}$$

How many scalar parameters n_θ are there?

$$\begin{aligned} n_\theta &= |W^{(1)}| + |b^{(1)}| + |W^{(2)}| + |b^{(2)}| \\ &= d \cdot d_h + d_h + d_h \cdot m + m \\ &= (d_h+1)d + (m+1)d_h \end{aligned}$$

Optimization problem:

$$argmin_\theta \hat{R}(L,D) = argmin_\theta \sum_d L(x^{(d)}, y^{(d)})$$

2.6

Batch gradient descent equation:

$$\theta \leftarrow \theta - \eta \frac{d\hat{R}}{d\theta}$$

2.7

To show:

$$\nabla L(o^a) = o^s - onehot_m(y) \qquad (1)$$

We have:

$$\nabla L(o^a) = \begin{pmatrix} ... \\ \frac{d}{do_k^a} - o_y^a + log(\sum_{i=1}^{m} e^{o_i^a}) \\ ... \end{pmatrix}$$

with

$$\frac{d}{do_k^a} - o_y^a + log(\sum_{i=1}^{m} e^{o_i^a}) = \begin{cases} -1 + \frac{e^{o_k^a}}{\sum_{i=1}^{m} e^{o_i^a}}, \text{ if } y = k \\ 0 + \frac{e^{o_k^a}}{\sum_{i=1}^{m} e^{o_i^a}}, \text{ if } y \neq k \end{cases}$$
$$= \begin{cases} -1 + softmax(o_k^a), \text{ if } y = k \\ softmax(o_k^a), \text{ if } y \neq k \end{cases}$$

so

$$\nabla L(o^a) = o^s - onehot_m(y)$$

2.8

```
onehot = np.zeros(m)
onehot[y-1] = 1
grad_oa= os - onehot
```

2.9

To compute: $\nabla L(W^{(2)}), \nabla L(b^{(2)})$. We know $\frac{d}{do_k^a}L$ and we have with W_k being the k-th line of W:

$$\frac{d}{dW_k^{(2)}}L = \frac{d}{do_k^a}L\frac{d}{dW_k^{(2)}}o_k^a$$
$$\frac{d}{db_k^{(2)}}L = \frac{d}{do_k^a}L\frac{d}{db_k^{(2)}}o_k^a$$

We have to compute:

$$\frac{d}{dW_k^{(2)}}o_k^a = \frac{d}{dW_k^{(2)}}(W_k^{(2)} \cdot h^S + b^{(2)}) = h^S$$
$$\frac{d}{db_k^{(2)}}o_k^a = \frac{d}{dW_k^{(2)}}(W_k^{(2)} \cdot h^S + b^{(2)}) = 1$$

Finally, we have:

$$\frac{d}{dW_k^{(2)}}L = (o_k^s - onehot_m(y)) \cdot h^S$$
$$\frac{d}{db_k^{(2)}}L = o_k^s - onehot_m(y)$$

2.10

In matrix form, we can write:

$$\nabla L(W^{(2)}) = \nabla L(o^a) \cdot (h^S)^T \in m \times d_h$$
$$\nabla L(b^{(2)}) = \nabla L(o^a) \in m \times 1$$

Dimensions:

$$\nabla L(o^a) \in 1 \times m$$
$$h^S \in 1 \times d_h$$
$$(h^S)^T \in d_h \times 1$$

In Python:

```
grad_b2 = grad_oa
grad_W2 = numpy.dot(grad_oa,numpy.transpose(h_s))
```

2.11

To compute:

$$\frac{d}{dh_j^s}L = \sum_{k=1}^{m} \frac{dL}{do_k^a} \frac{d}{dh_j^s} o_k^a$$

The only unknown is $\frac{d}{dh_j^s} o_k^a$. We have:

$$\frac{d}{dh_j^s} o_k^a = \frac{d}{dh_j^s}(W_k^{(2)} \cdot h^S + b^{(2)}) = w_{k,j}^{(2)}$$

$w_{k,j}$ being the j-th column of the k-th line of W. With the sum, we have:

$$\frac{d}{dh_j^s}L = \sum_{k=1}^{m} \frac{dL}{do_k^a} \frac{do_k^a}{dh_j^s} = \sum_{k=1}^{m} \frac{dL}{do_k^a} w_{k,j}^{(2)} = (W_{\bullet,j}^{(2)})^T \frac{dL}{do^a}$$

2.12

In matrix form, we can write :

$$\nabla L(h^s) = (W^{(2)})^T \nabla L(o^a) \in d_h \times 1$$

Dimensions :

$$dim((W^{(2)})^T) = d_h \times m$$
$$dim(\nabla L(o^a)) = m \times 1$$

In Python:

```
grad_hs = numpy.dot(numpy.transpose(W2), grad_oa)
```

2.13

From theoretical part A we know that :

$$rect'(x) = \begin{cases} 1, \text{ if } x > 0 \\ 0, \text{ if } x \leq 0 \end{cases} = \mathbb{1}_{\{x>0\}}(x)$$

To compute:

$$\frac{d}{dh_j^a} L = \frac{dL}{dh_j^s} \frac{d}{dh_j^a} h_j^s$$

We know $\frac{d}{dh_j^s} L$ and we have:

$$\frac{d}{dh_j^a} h_j^s = \frac{d}{dh_j^a} rect(h_j^a) = \begin{cases} 1, \text{ if } h_j^a > 0 \\ 0, \text{ if } h_j^a \leq 0 \end{cases}$$
$$= \mathbb{1}_{\{h_j^a>0\}}(h_j^a)$$

Finally, we have:

$$\frac{d}{dh_j^a} L = (W_{\bullet,j}^{(2)})^T \nabla L(o^a) \mathbb{1}_{\{h_j^a>0\}}(h_j^a)$$

2.14

In matrix form, we can write with $\odot$ being element-wise multiplication:

$$\nabla L(h^a) = \nabla L(h^s) \odot \mathbb{1}_{\{h_j^a>0\}}(h_j^a)$$

Dimensions :

$$dim(\nabla L(h^s)) = d_h \times 1$$
$$dim(\mathbb{1}_{\{h_j^a>0\}}(h_j^a)) = d_h \times 1$$

In Python:

```
grad_ha = numpy.multiply(grad_hs, indicator(ha))
```

2.15

To compute: $\nabla L(W^{(1)}), \nabla L(b^{(1)})$. We know $\frac{d}{dh^a_j}L$ and we have:

$$\frac{d}{dW^{(1)}_j}L = \frac{d}{dh^a_j}L\frac{d}{dW^{(1)}_j}h^a_j$$
$$\frac{d}{db^{(1)}_j}L = \frac{d}{dh^a_j}L\frac{d}{db^{(1)}_j}h^a_j$$

We have to compute:

$$\frac{d}{dW^{(1)}_j}h^a_j = \frac{d}{dW^{(1)}_j}(W^{(1)}_j \cdot x + b^{(1)}) = x$$
$$\frac{d}{db^{(1)}_j}h^a_j = \frac{d}{db^{(1)}}(W^{(1)}_j \cdot x + b^{(1)}) = 1$$

Finally, we have:

$$\frac{d}{dW^{(1)}_j}L = \frac{d}{dh^a_j}L \cdot x$$
$$\frac{d}{db^{(1)}_j}L = \frac{d}{dh^a_j}L$$

2.16

In matrix form, we can write :

$$\nabla L(W^{(1)}) = \nabla L(h^a) \cdot x^T \in d^h \times d$$
$$\nabla L(b^{(1)}) = \nabla L(h^a)$$

Dimensions :

$$dim(\nabla L(h^a)) = d_h \times 1$$
$$dim(x) = d \times 1$$

In Python:

```
grad_b1 = grad_ha
grad_W1 = numpy.dot(grad_ha,numpy.transpose(x))
```

2.17

Following the same logic than in question 2.13,we have:

$$\frac{dL}{dx} = \sum_{k=1}^{d_h} \frac{dL}{dh_k^a} \frac{dh_k^a}{dx_j} = (W_{\bullet,j}^{(1)})^T \frac{dL}{dh^a}$$

And in matrix form:

$$\nabla L(x) = (W^{(1)})^T \nabla L(h^a)$$

In Python:

```
grad_b2 = numpy.dot(numpy.tranpose(W1),grad_ha)
```

2.18

We have two parameters: W and b. The gradient of b is unchanged. The gradient of W will be affected by the deduction of its sign (L^1) and by the addition of two times its value (L^2) :

$$\nabla L(W^{(2)}) = \nabla L(o^a) \cdot (h^S)^T + 2 \cdot W^{(2)} - sign(W^{(2)})$$
$$\nabla L(W^{(1)}) = \nabla L(h^a) \cdot x^T + 2 \cdot W^{(1)} - sign(W^{(1)})$$

```
grad_w2_el = grad_w2+2*w2-np.sign(w2)
grad_w1_el = grad_w1+2*w1-np.sign(w1)
```