IFT6390-fundamentals of machine learning
Assignment 3

Jonathan Guymont, Marzieh Mehdizadeh

$$x \tag{1}$$

Question 1

(a)

$$\begin{aligned} 0.5(\tanh(0.5x)+1) =& 0.5(\frac{e^{0.5x}-e^{-0.5x}}{e^{0.5x}+e^{-0.5x}}+1) \\ =& 0.5(\frac{1-e^{-x}}{1+e^{-x}}+\frac{1+e^{-x}}{1+e^{-x}}) \\ =& 0.5\frac{2}{1+e^{-x}} \\ =& \frac{1}{1+e^{-x}} \end{aligned} \tag{2}$$

(b)

$$\begin{aligned} \log \text{sigmoid}(x) =& \log(1+e^{-x})^{-1} \\ =& -\log(1+e^{-x}) \\ =& -\text{softmax}(-x) \end{aligned} \tag{3}$$

(c)

$$\begin{aligned} \frac{d}{dx}\text{sigmoid}(x) =& \frac{d}{dx}(1+e^{-x})^{-1} \\ =& -(1+e^{-x})^{-2}\frac{d}{dx}1+e^{-x} \\ =& (1+e^{-x})^{-2}e^{-x} \\ =& (1+e^{-x})^{-1}\frac{e^{-x}}{1+e^{-x}} \\ =& (1+e^{-x})^{-1}(\frac{1+e^{-x}-1}{1+e^{-x}}) \\ =& (1+e^{-x})^{-1}(1-\frac{1}{1+e^{-x}}) \end{aligned} \tag{4}$$

(4)

$$\begin{aligned}\tanh'(x) =&(\frac{e^x - e^{-x}}{e^x + e^{-x}})' \\ =&\frac{(e^x + e^{-x})(e^x + e^{-x}) - (e^x - e^{-x})(e^x - e^{-x})}{(e^x + e^{-x})^2} \\ =&\frac{(e^x + e^{-x})^2 - (e^x - e^{-x})^2}{(e^x + e^{-x})^2} \\ =&1 - \frac{(e^x - e^{-x})^2}{(e^x + e^{-x})^2} \\ =&1 - \tanh^2(x)\end{aligned} \tag{5}$$

(5) $\text{sign}(x) = -1 + 2 \cdot \mathbb{1}_{x>0}$

(6) $\text{abs}'(x) = \text{sign}(x)$

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

(8) Let $f(\boldsymbol{x}) = \sum_{x_i \in \boldsymbol{x}} x_i^2$, then $f'(\boldsymbol{x}) = (f'_{x_1}, ..., f'_{x_{|\boldsymbol{x}|}}) = (2x_1, ..., 2x_{|\boldsymbol{x}|})$

(9) Let $f(\boldsymbol{x}) = \sum_{x_i \in \boldsymbol{x}} |x_i|$, then $f'(\boldsymbol{x}) = (f'_{x_1}, ..., f'_{x_{|\boldsymbol{x}|}}) = (\text{sign}(x_1), ..., \text{sign}(x_{|\boldsymbol{x}|}))$

Gradient Computation for Parameters optimizations in a neural net for multiclass classification

(1) The dimension of $\boldsymbol{b}^{(1)}$ is $d_h \times 1$. The formula of the preactivation vector is

$$\boldsymbol{h}^a = \boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}$$

and the formula for obtaining the value of the element j is

$$\boldsymbol{h}_j^a = \boldsymbol{W}_{\boldsymbol{j},\cdot}^{(1)}\boldsymbol{x} + \boldsymbol{b}_j^{(1)} = \boldsymbol{b}_j^{(1)} + \sum_{i=1}^{d} \boldsymbol{W}_{\boldsymbol{j},\boldsymbol{i}}^{(1)}\boldsymbol{x}_i.$$

The output vector of the activation is given by

$$\boldsymbol{h}^s = \text{relu}(\boldsymbol{h}^a)$$

where relu$(\cdot)$ is applied element wise, i.e. $\boldsymbol{h}_j^s = \max(0, \boldsymbol{h}_j^a)$, $j = 1, ..., d_h$.

(2) The dimension of $\boldsymbol{W}^{(2)}$ is $m \times d_h$ and the dimension of $\boldsymbol{b}^{(2)}$ is $m \times 1$.

$$\boldsymbol{o}^a = \boldsymbol{W}^{(2)}\boldsymbol{h}^s + \boldsymbol{b}^{(2)}$$

$$\boldsymbol{o}_k^a = \boldsymbol{W}_{k,\cdot}^{(2)}\boldsymbol{h}^s + \boldsymbol{b}_k^{(2)} = \boldsymbol{b}_k^{(2)} + \sum_{i=1}^{d_h} \boldsymbol{W}_{\boldsymbol{k},\boldsymbol{i}}^{(2)}\boldsymbol{h}_i^s$$

for $k = 1, .., m$.

(3)

$$\boldsymbol{o}_k^s = \frac{\exp(\boldsymbol{o}_k^a)}{\sum_{k=1}^m \exp(\boldsymbol{o}_k^a)} \tag{6}$$

They are all positive because $\exp\colon \mathbb{R} \mapsto \mathbb{R}^+$. Also a sum of positive number is positive. And the ratio of a posive number over a positive number is also positive.

$$\sum_{k=1}^m \boldsymbol{o}_k^s = \frac{1}{\sum_{k=1}^m \exp(\boldsymbol{o}_k^a)} \sum_{k=1}^m \exp(\boldsymbol{o}_k^a) = 1$$

(4) Let $Z = \sum_{k=1}^m \exp(\boldsymbol{o}_k^a)$. Then $\boldsymbol{o}^s = \frac{1}{Z}(\exp(\boldsymbol{o}_1^a), ..., \exp(\boldsymbol{o}_m^a))^\top$ and

$$L(\boldsymbol{x}, y) = -\log \text{onehot}_m(y)(\exp(\boldsymbol{o}_1^a(\boldsymbol{x}))/Z, ..., \exp(\boldsymbol{o}_m^a(\boldsymbol{x}))/Z)^\top = -\log \text{onehot}_m(y)\boldsymbol{o}^s(\boldsymbol{x})$$

where $\text{onehot}_m(y)$ is a $1 \times m$ onehot representation for y.

(5) $\hat{R}$ is an estimation of the expected value of the loss function (minus the loglikelihood in our case)

$$\hat{R} = \frac{1}{n} \sum_{i=1}^n L(x^{(i)}, y^{(i)}) = \frac{1}{n} \sum_{i=1}^n -\log \text{onehot}_m(y^{(i)})\boldsymbol{o}^s(\boldsymbol{x}^{(i)})$$

The set of trainable parameters is $\boldsymbol{\theta} = \{W^{(1)}, W^{(2)}, b^{(1)}, b^{(2)}\}$. The number of scalar parameters is $n_\theta = d_h \cdot d + d_h + m \cdot d_h + m = d_h(d+1) + m(d_h+1)$.

Optimization problem. First we need to initialize the parameters properly. To find the parameters that minimize the loss, we need to compute the derivative of the loss function w.r.t each parameters. Then we update each parameters by moving them in the opposite direction of their gradient (since we minimize). We repeat this step until a stopping criterion is met (e.g. maximum number of iteration is reached when using early stopping).

(6)

Algorithm 1 Pseudocode for Batch Gradient Descent

Require: Step size η
Require: Initial parameter $\boldsymbol{\omega}_0$
Require: Number of iterations T
  for $i = 1$ to T **do**
    Compute gradient $\boldsymbol{g}_t = \frac{1}{m}\nabla_{\boldsymbol{\omega}} \sum_i L(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$
    Apply update: $\boldsymbol{\omega}_t = \boldsymbol{\omega}_{t-1} - \eta \boldsymbol{g}_t$
  end for

(7)

$$
\begin{aligned}
\frac{\partial}{\partial \boldsymbol{o}^a} L =& \frac{\partial}{\partial \boldsymbol{o}^a} - \log \text{onehot}_m(y) \boldsymbol{o}^s \\
=& - \frac{1}{\text{onehot}_m(y) \boldsymbol{o}^s} \frac{\partial}{\partial \boldsymbol{o}^a} \text{onehot}_m(y) \boldsymbol{o}^s \\
=& - \frac{1}{\text{onehot}_m(y) \boldsymbol{o}^s} \frac{\partial}{\partial \boldsymbol{o}^a} \text{onehot}_m(y) \text{softmax}(\boldsymbol{o}^a) \\
=& - \frac{1}{\text{onehot}_m(y) \boldsymbol{o}^s} \text{onehot}_m(y) \text{softmax}(\boldsymbol{o}^a) (\text{onehot}_m(y) - \text{softmax}(\boldsymbol{o}^a)) \\
=& - (\text{onehot}_m(y) - \text{softmax}(\boldsymbol{o}^a))
\end{aligned}
\tag{7}
$$

(8)

```
import numpy as np

def grad_oa(x, y):
    onehot = np.zeros(m)
    onehot[y] = 1
    return os(x) - onehot
```

Neural Network Implementation and Experiment