Deep Learning : Theory and Practice

Midterm Exam
Date: March 8, 2019 (6:15PM)

Instructions

1. This exam is open notes. However, computers, mobile phones and other handheld devices are not allowed.
2. Notation - bold symbols are vectors, capital bold symbols are matrices and regular symbols are scalars.
3. All questions even if they are multiple choice need justification. No marks will be provided for answers without justification.
4. **Online exam instructions** - Download question paper, write name and question number and the answer in A4 sheets, scan and send the answer sheets back in email to *deeplearning.cce2019@gmail.com*. **No external online resources can be accessed during the exam.** Any notes, books or slide material from the class can be used.
5. Total Duration In Class - **60 minutes**
6. Total Duration Online Exam - **90 minutes**
7. Total Marks - **50 points**

Name - .....

Organization - .....

1. The sigmoid function is defined as

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

and the ReLU function is defined as

$$\text{ReLU}(x) = \begin{cases} 0 & ; x \leq 0 \\ x & ; x > 0 \end{cases}$$

The derivative of $\sigma(\text{ReLU}(a))$ with respect to a a is ....

(a) $\begin{cases} \sigma(a) - \sigma(a)^2 & a \geq 0 \\ 0.5 & a < 0 \end{cases}$

(b) $\begin{cases} \sigma(a) - \sigma(a)^2 & a > 0 \\ 0 & a \leq 0 \end{cases}$

(c) $\begin{cases} \sigma(a) - \sigma(a)^2 & a \geq 0 \\ 0 & a < 0 \end{cases}$

(d) $\begin{cases} \sigma(a) - \sigma(a)^2 & a > 0 \\ 0.5 & a \leq 0 \end{cases}$

(e) $\sigma(a) - \sigma(a)^2$

(**Points** 5)

Solution:

$$\sigma(\text{ReLU}(a)) = \begin{cases} 0.5 & ; a \leq 0 \\ \sigma(a) & ; a > 0 \end{cases} \tag{1}$$

$$\frac{\partial}{\partial a}\sigma(\text{ReLU}(a)) = \begin{cases} 0 & ; a < 0 \\ \sigma(a)(1 - \sigma(a)) & ; a > 0 \end{cases} \tag{2}$$

As the derivative is discontinuous at $a = 0$, both options (b) and (c) are accepted.

2. If A is constrained to be a symmetric matrix, the derivative $\frac{\partial}{\partial A}\boldsymbol{x}^T A\boldsymbol{x}$ = ...

   (a) $2\boldsymbol{x}\boldsymbol{x}^T$
   (b) $\boldsymbol{x}\boldsymbol{x}^T$
   (c) $\boldsymbol{x}\boldsymbol{x}^T - diag(\boldsymbol{x}\boldsymbol{x}^T)$
   (d) $2\boldsymbol{x}\boldsymbol{x}^T - diag(\boldsymbol{x}\boldsymbol{x}^T)$

(**Points** 5)

Solution: Assuming $\boldsymbol{x}$ is a D dimensional vector and A is a symmetric matrix of dimension $D \times D$,

$$\boldsymbol{x}^T A\boldsymbol{x} = \sum_{m=1}^{D}\sum_{n=1}^{D} x_m A_{mn} x_n$$

$$\frac{\partial}{\partial A_{ij}}(\boldsymbol{x}^T A\boldsymbol{x}) = \begin{cases} x_i x_j + x_j x_i & \text{if } i \neq j \quad \because A_{ij} = A_{ji} \\ x_i x_i & \text{if } i = j \end{cases}$$

$$= \begin{cases} 2x_i x_j & \text{if } i \neq j \\ x_i x_i & \text{if } i = j \end{cases}$$

$$\frac{\partial}{\partial A}(\boldsymbol{x}^T A\boldsymbol{x}) = \begin{pmatrix} x_1x_1 & 2x_1x_2 & \dots & 2x_1x_D \\ 2x_2x_1 & x_2x_2 & \dots & 2x_2x_D \\ \vdots & \vdots & \ddots & \vdots \\ 2x_Dx_1 & 2x_Dx_2 & \dots & x_Dx_D \end{pmatrix}$$

$$= 2\begin{pmatrix} x_1x_1 & x_1x_2 & \dots & x_1x_D \\ x_2x_1 & x_2x_2 & \dots & x_2x_D \\ \vdots & \vdots & \ddots & \vdots \\ x_Dx_1 & 2x_Dx_2 & \dots & x_Dx_D \end{pmatrix} - \begin{pmatrix} x_1x_1 & 0 & \dots & 0 \\ 0 & x_2x_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & x_Dx_D \end{pmatrix}$$

$$= 2\boldsymbol{x}\boldsymbol{x}^T - diag(\boldsymbol{x}\boldsymbol{x}^T)$$

Option (d) is the right answer.

3. **Neural Networks - Cost Function** - Let us define a NN with $\{\boldsymbol{x}_i\}_{i=1}^N$ and $\{\boldsymbol{t}_i\}_{i=1}^N$ denote the input and targets to the NN for N training data points. The task is classification with hard targets $\boldsymbol{t} \in \mathcal{B}^{C\times 1}$, where $\mathcal{B}$ denotes boolean variable (0 *or* 1), and C is the number of classes. Note that every data point $\boldsymbol{x}_i$ is associated with only one class label c_i where $c_i \in \{1, 2, .., C\}$ classes. The output of the network is denoted as $\boldsymbol{y}_i$ where $\boldsymbol{y}_i = \{y_i^1, y_i^2, .., y_i^C\} \in \mathcal{R}^{C\times 1}$ and $0 < \boldsymbol{y}_i < 1$. The NN cost function is defined using mean square error (MSE).

$$E_{MSE} = \sum_{i=1}^{N} ||\boldsymbol{t}_i - \boldsymbol{y}_i||^2$$

Which of the following statements are true regarding the MSE loss function (justify your answer).

(a) $\sum_{i=1}^{N} \left[1 - y_i^{c_i}\right] > E_{MSE}$

(b) $\sum_{i=1}^{N} \left[1 - y_i^{c_i}\right]^2 \leq E_{MSE}$

(c) $\sum_{i=1}^{N} \left[1 - y_i^{c_i}\right]^2 \geq E_{MSE}$

(d) None of the above

(**Points** 10)

Solution:

Option (a) **need not always be true**. Here is a counter example:
Let $N = 1$, $C = 2$, $c_1 = 1$, $c_2 = 0$, $y_1^1 = 0.2$, $y_1^2 = 0.8$.

$$\begin{aligned} 1 - y_1^{c_1} &= 1 - 0.2 \\ &= 0.8 \\ E_{MSE} &= (1 - 0.2)^2 + (0 - 0.8)^2 \\ &= 1.28 \end{aligned}$$

Option (b) **is always true**. Proof:

$$\begin{aligned} E_{MSE} &= \sum_{i=1}^{N} ||\boldsymbol{t}_i - \boldsymbol{y}_i||^2 \\ &= \sum_{i=1}^{N} \sum_{c=1}^{C} (t_i^c - y_i^c)^2 \\ &= \sum_{i=1}^{N} \left[(1 - y_i^{c_i})^2 + \underbrace{\sum_{\substack{c=1 \\ c\neq c_i}}^{C} (-y_i^c)^2}_{\text{A non-negative quantity}} \right] \\ &\geq \sum_{i=1}^{N} \left[1 - y_i^{c_i}\right]^2 \end{aligned}$$

Options (c) and (d) **are false** by contradiction.

4. A binary linear classifier is trained using mean square error loss and is given by $y = \boldsymbol{w}^T\boldsymbol{x} + b$, where $y > 0$ is associated with class C_1 and $y \leq 0$ with class C_2. Separately a logistic regression classifier is trained using maximum likelihood criterion and it outputs $y = \sigma(\boldsymbol{w}^T\boldsymbol{x} + b)$ with $y > 0.5$ associated with class C_1 and $y \leq 0.5$ associated with class C_2. Check whether the following statements are true or false (justify your choice in each case).

   (a) If the weight $\boldsymbol{w}$ and the bias b are the same for both classifiers, the logistic regression performs better classification than the linear classifier.

   (b) When trained using the same training data, both the linear classifier and the logistic regression classifier have the same parameters $\boldsymbol{w}$ and b.

(**Points** 10)

Answer:

(a)

$$\begin{aligned}
\sigma(\boldsymbol{w}^T\boldsymbol{x} + b) = 0.5 \quad &\Longleftrightarrow \quad \boldsymbol{w}^T\boldsymbol{x} + b = 0 \qquad \text{(Boundary condition)} \\
\sigma(\boldsymbol{w}^T\boldsymbol{x} + b) < 0.5 \quad &\Longleftrightarrow \quad \boldsymbol{w}^T\boldsymbol{x} + b < 0 \qquad \text{(Decided class } C_1\text{)} \\
\underbrace{\sigma(\boldsymbol{w}^T\boldsymbol{x} + b) > 0.5}_{\text{Logistic regression decision rule}} \quad &\Longleftrightarrow \quad \underbrace{\boldsymbol{w}^T\boldsymbol{x} + b > 0}_{\text{Linear classifier decision rule}} \qquad \text{(Decided class } C_2\text{)}
\end{aligned}$$

If the weight $\boldsymbol{w}$ and the bias b are the same for both classifiers, the decision made on a test feature vector $\boldsymbol{x}$ is the same by both logistic regression and linear classifiers. Hence, statement (a) **is FALSE.**

(b) The linear classifier minimizes the mean squared error function

$$\begin{aligned}
E_{MSE} &= \sum_{i=1}^{N} (y_i - t_i)^2 \\
&= \sum_{i=1}^{N} (\boldsymbol{w}^T\boldsymbol{x}_i + b - t_i)^2
\end{aligned}$$

and the logistic regression classifier maximizes the likelihood

$$\begin{aligned}
L &= \prod_{i=1}^{N} \mathrm{p}(t_i = 1|\boldsymbol{x}_i) \\
&= \prod_{i=1}^{N} (y_i)^{t_i}(1 - y_i)^{(1-t_i)} \\
&= \prod_{i=1}^{N} \sigma(\boldsymbol{w}^T\boldsymbol{x}_i + b)^{t_i}(1 - \sigma(\boldsymbol{w}\boldsymbol{x}_i + b))^{(1-t_i)}
\end{aligned}$$

The two classifier are optimized on two different functions, which do not necessarily have the same optimal point. Hence, statement (b) is **FALSE**.

Moreover, the MSE of a linear classifier is a convex function in $\boldsymbol{w}$ and b, whereas the likelihood function is non convex. It could have several local minima. When trained using the same training data, it can not be guaranteed that the linear classifier and the logistic regression classifier have the same parameters $\boldsymbol{w}$ and b.

5. A deep neural network with 1 hidden layer is described as follows. The input is denoted as $\boldsymbol{x}$, and the output of layer 1 as $\boldsymbol{z}^{(1)} = \boldsymbol{\phi}(W^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)})$, where $\boldsymbol{\phi}$ is the non-linearity applied. The output activation of the second layer is given as $\boldsymbol{a}^{(2)} = (W^{(2)}\boldsymbol{z}^{(1)} + \boldsymbol{b}^{(2)})$, and the network output is $\boldsymbol{y} = \boldsymbol{\psi}(\boldsymbol{a}^{(2)})$ where $\boldsymbol{\psi}$ can be of two types, linear activation ($\boldsymbol{\psi}(\boldsymbol{a}^{(2)}) = \boldsymbol{a}^{(2)}$) and soft-max activation $\boldsymbol{\psi}(\boldsymbol{a}^{(2)}) = softmax(\boldsymbol{a}^2)$. The error function in the case of linear activation is the mean square loss $E_{MSE} = ||\boldsymbol{t} - \boldsymbol{y}||^2$, while we use a cross entropy loss $E_{CE} = -\sum_{i=1}^{C} t_i \log y_i$ for the soft-max activation. Here $\boldsymbol{t}$ is the target vector.

   Check whether the following statement is true or false.

   "The derivative of cross entropy loss $\left(\frac{\partial E_{CE}}{\partial \boldsymbol{a}^{(2)}}\right)$ with softmax target activation is equivalent to derivative of MSE loss $\left(\frac{\partial E_{MSE}}{\partial \boldsymbol{a}^{(2)}}\right)$ with linear target activation."

   (**Points** 10)

   Answer:

   For the first case, we have

$$\begin{aligned} E_{MSE} &= ||\boldsymbol{t} - \boldsymbol{y}||^2 \\ &= (\boldsymbol{t} - \boldsymbol{y})^T(\boldsymbol{t} - \boldsymbol{y}) \end{aligned}$$

   Taking derivative wrt $\boldsymbol{a}^{(2)}$

$$\begin{aligned} \frac{\partial E_{MSE}}{\partial \boldsymbol{a}^{(2)}} &= \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{a}^{(2)}} \frac{\partial E_{MSE}}{\partial \boldsymbol{y}} \\ &= I\left\{-2(\boldsymbol{t} - \boldsymbol{y})\right\} \qquad [\text{since } \boldsymbol{y} = \boldsymbol{a}^{(2)}] \\ &= 2(\boldsymbol{y} - \boldsymbol{t}) \end{aligned}$$

   For the second case,

$$\begin{aligned} E_{CE} &= -\sum_{j=1}^{C} t_j \log y_j \\ &= -\sum_{j=1}^{C} t_j \log \frac{e^{a_j^{(2)}}}{\sum_{c=1}^{C} e^{a_c^{(2)}}} \\ &= -\sum_{j=1}^{C} t_j a_j^{(2)} + \sum_{j=1}^{C} t_j \log \sum_{c=1}^{C} e^{a_c^{(2)}} \end{aligned}$$

   Taking derivative wrt $a_i^{(2)}$

$$\begin{aligned} \frac{\partial E_{CE}}{\partial a_i^{(2)}} &= -t_i + \sum_{j=1}^{C} t_j \underbrace{\frac{e^{a_i^{(2)}}}{\sum_{c=1}^{C} e^{a_c^{(2)}}}}_{softmax(a_i^{(2)})} \\ &= y_i - t_i \end{aligned}$$

$$\frac{\partial E_{CE}}{\partial \boldsymbol{a}^{(2)}} = \boldsymbol{y} - \boldsymbol{t}$$

Though the derivatives are equivalent in terms of the network output $\boldsymbol{y}$, the network outputs itself are different in both cases. (The first network has a linear activation and the second network has softmax activation. Hence they are not equivalent.)

6. Assume a neural network with 1 hidden layer, where all the weights and bias parameters of the model are initialized to 0 with *Tanh* as the activation function in the hidden layer and linear activation at the output layer. Check whether the following statement is true or false.

   "The weight matrix in the first layer W^1 does not get updated during the backpropagation"

   (**Points** 10)

   Answer:

   The neural network can be expressed as follows:

$$\boldsymbol{x} \xrightarrow{W^{(1)}\boldsymbol{x}+\boldsymbol{b}^{(2)}} \boldsymbol{a}^{(1)} \xrightarrow{tanh(\boldsymbol{a}^{(1)})} \boldsymbol{y}^{(1)} \xrightarrow{W^{(2)}\boldsymbol{y}^{(1)}+\boldsymbol{b}^{(2)}} \boldsymbol{a}^{(2)} \xrightarrow{\text{(linear activation)}} \boldsymbol{y}^{(2)}$$

   The backpropogated derivatives of the error function wrt $W^{(1)}$ and $W^{(2)}$ are given by

$$\frac{\partial E}{\partial W^{(2)}} = \underbrace{\frac{\partial \boldsymbol{a}^{(2)}}{\partial W^{(2)}}}_{=\boldsymbol{y}^{(1)T}=\mathbf{0}} \frac{\partial \boldsymbol{y}^{(2)}}{\partial \boldsymbol{a}^{(2)}} \frac{\partial E}{\partial \boldsymbol{y}^{(2)}}$$

$$\frac{\partial E}{\partial W^{(1)}} = \frac{\partial \boldsymbol{a}^{(1)}}{\partial W^{(1)}} \frac{\partial \boldsymbol{y}^{(1)}}{\partial \boldsymbol{a}^{(1)}} \underbrace{\frac{\partial \boldsymbol{a}^{(2)}}{\partial \boldsymbol{y}^{(1)}}}_{=W^{(2)}=0} \frac{\partial \boldsymbol{y}^{(2)}}{\partial \boldsymbol{a}^{(2)}} \frac{\partial E}{\partial \boldsymbol{y}^{(2)}}$$

   When the weights and biases are initialized to zero, the gradients wrt $W^{(2)}$ is zero, and hence $W^{(2)}$ do not get updated. The derivative wrt $W^{(1)}$ will remain zero, and hence $W^{(1)}$ will also never get updated.

   So the statement is **TRUE.**