FOR INTERNAL SCRUTINY (date of this version: 23/2/2025)

UNIVERSITY OF EDINBURGH

COLLEGE OF SCIENCE AND ENGINEERING

SCHOOL OF INFORMATICS

INFR10086 MACHINE LEARNING

May 2025

00:00 to 00:00

INSTRUCTIONS TO CANDIDATES

1. **Note that ALL QUESTIONS ARE COMPULSORY.**
2. **DIFFERENT QUESTIONS MAY HAVE DIFFERENT NUMBERS OF TOTAL MARKS. Take note of this in allocating time to questions.**
3. **This is a NOTES PERMITTED, CALCULATORS NOT PERMITTED examination.**

   Candidates may consult up to THREE A4 pages (6 sides) of notes.

   CALCULATORS MAY NOT BE USED IN THIS EXAMINATION.

Year 3 Courses

Convener: P.Series
External Examiners: D.Parker, A.Pocklington, R.Mullins

THIS EXAMINATION WILL BE MARKED ANONYMOUSLY

1. In this question, we will look at a loss function for training a linear classifier for binary classification. As a reminder, a linear classifier for binary classification is a function $\mathbb{R}^d \to \{+1, -1\}$ that takes the form

$$f(\mathbf{x}) = \begin{cases} +1 & \text{if } \mathbf{w}^\top \mathbf{x} \geq 0 \\ -1 & \text{otherwise} \end{cases} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^d$.

(a) Show that for all $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{w} \in \mathbb{R}^d$, and $y \in \{+1, -1\}$, the loss function

$$L(\mathbf{x}, y; \mathbf{w}) = (y\mathbf{w}^\top \mathbf{x} - 1)^2 \tag{2}$$

is an upper bound of the zero-one loss

$$L_{01}(\mathbf{x}, y; \mathbf{w}) = \mathbb{1}_{y\mathbf{w}^\top \mathbf{x} < 0}. \tag{3}$$

[5 marks]

(b) Before we run an optimization algorithm, let's look at the properties of L.

i. Show that L is convex in $\mathbf{w}$. *[4 marks]*

ii. Show that $\nabla_{\mathbf{w}} L(\mathbf{w}^*) = 0$ is a sufficient condition for $\mathbf{w}^*$ to be the optimal solution of L when L is convex. *[4 marks]*

(c) Suppose we want to optimize the loss function with gradient descent. Given a data set $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, the loss of the classifier on the data set is

$$F(\mathbf{w}) = \sum_{i=1}^{n} L(\mathbf{x}_i, y_i; \mathbf{w}). \tag{4}$$

Derive $\nabla_{\mathbf{w}} F(\mathbf{w})$. *[4 marks]*

2. In this question, we are going to implement our own attention layer in a neural network library. The forward computation of the attention layer is defined as

$$f(\mathbf{q}, \mathbf{k}_1, \ldots, \mathbf{k}_T, \mathbf{v}_1, \ldots, \mathbf{v}_T) = \sum_{t=1}^{T} \frac{\exp(\mathbf{q}^\top \mathbf{k}_t)}{\sum_{t'=1}^{T} \exp(\mathbf{q}^\top \mathbf{k}_{t'})} \mathbf{v}_t, \tag{5}$$

where $\mathbf{q}$, $\mathbf{k}_1, \ldots, \mathbf{k}_T$, and $\mathbf{v}_1, \ldots, \mathbf{v}_T$ are all in $\mathbb{R}^D$. The backward computation is needed for computing the gradient with backpropagation.

(a) We first seek to derive $\frac{\partial}{\partial \mathbf{q}} f$ with the following steps. Recall that

$$\frac{\partial}{\partial \mathbf{q}} f = \begin{bmatrix} \frac{\partial}{\partial q_1} f \\ \frac{\partial}{\partial q_2} f \\ \vdots \\ \frac{\partial}{\partial q_D} f \end{bmatrix}. \tag{6}$$

[QUESTION CONTINUES ON NEXT PAGE]

[QUESTION CONTINUES FROM PREVIOUS PAGE]

i. Let $p_t = \mathbf{q}^\top \mathbf{k}_t$. Derive $\frac{\partial}{\partial q_d} p_t$ for $d = 1, \ldots, D$. If you need to index a dimension d in $\mathbf{k}_t$, you can write $[\mathbf{k}_t]_d$. *[3 marks]*

ii. Let

$$\alpha_t = \frac{\exp(p_t)}{\sum_{t'=1}^{T} \exp(p_{t'})}. \tag{7}$$

Derive $\frac{\partial}{\partial p_j} \alpha_i$ for any $i = 1, \ldots, T$ and $j = 1, \ldots, T$. *[5 marks]*

iii. Now we can rewrite f as

$$f(\mathbf{q}, \mathbf{k}_1, \ldots, \mathbf{k}_T, \mathbf{v}_1, \ldots, \mathbf{v}_T) = \sum_{t=1}^{T} \alpha_t \mathbf{v}_t. \tag{8}$$

Use chain rule to derive $\frac{\partial}{\partial q_d} f$ in terms of $\frac{\partial}{\partial p_j} \alpha_i$ and $\frac{\partial}{\partial \mathbf{q}} p_t$. Note that you don't need to plug in the results from i. and ii. and won't be penalized if you did not get i. and ii. right. *[4 marks]*

(b) Now that we have an analytical implementation to compute $\frac{\partial}{\partial \mathbf{q}} f$. Explain how you would use the finite difference method to verify your implementation. *[3 marks]*

3. Suppose we have a 768-dimensional data set consisting of 39 data points. We are going to use PCA to analyze the data. The tool we use to run PCA will do centering for us, so that we do not need to worry about whether the data is centered or not.

	dimension 1	dimension 2	dimension 3	$\cdots$	dimension 768
sample 1	5.4	0.01	3.3	$\cdots$	4.6
sample 2	8.2	0.03	4.2	$\cdots$	3.7
sample 3	7.3	0.02	2.4	$\cdots$	5.4
sample 4	6.6	0.05	3.1	$\cdots$	6.5
sample 5	4.2	0.02	4.7	$\cdots$	3.3
$\vdots$					
sample 39	5.8	0.04	3.2	$\cdots$	7.2

(a) Before running PCA, you realize the second dimension is consistently smaller than the other dimensions. You are worried that the scale of the second dimension is too small and decide to multiply that by 100.0. Does multiplying the second dimension by a large constant change the principal components of the subsequent PCA? Detail your reasons. *[6 marks]*

[QUESTION CONTINUES ON NEXT PAGE]

[QUESTION CONTINUES FROM PREVIOUS PAGE]

(b) At most how many principal components would we get if we run PCA on this data set? Explain why. In what circumstance would we get fewer principal components? [*6 marks*]

(c) Suppose you run PCA and keep the top 25 principal components, and project the data from 768 dimensions down to 25. You later realize 25 dimensions are still too many and decide to do a second PCA on top of the 25-dimensional data and reduce the dimensions down to 10. Would the resulting 10-dimensional data set be any different if we do PCA once (as opposed to doing PCA twice) and reduce 768 dimensions directly to 10 dimensions? Detail your reasons. [*6 marks*]