Data Structures and Algorithms
with applications in Machine Learning
- Mock MCQ 1 -

NAME: ____________________ **GROUP:** ____________________

Each Question: 1 Mark **Duration:** 30 Minutes

Completely fill the circles as shown: ○○●○

Answer sheet

Q1.	○	a.	Q6.	○	a.
	○	b.		○	b.
	○	c.		○	c.
	○	d.		○	d.
Q2.	○	a.	Q7.	○	a.
	○	b.		○	b.
	○	c.		○	c.
	○	d.		○	d.
Q3.	○	a.	Q8.	○	a.
	○	b.		○	b.
	○	c.		○	c.
	○	d.		○	d.
Q4.	○	a.	Q9.	○	a.
	○	b.		○	b.
	○	c.		○	c.
	○	d.		○	d.
Q5.	○	a.	Q10.	○	a.
	○	b.		○	b.
	○	c.		○	c.
	○	d.		○	d.

The Quiz

Q. 1 Complete the missing part of the function to create a `word_index` dictionary from a list of documents.

The input is a list of documents, where each document is a list of tokens. For example:

```
documents = [["the", "cat", "sat"], ["the", "dog", "barked"]]
```

The function should return a dictionary where each unique word is assigned a unique integer starting from 1. For example:

```
{
 "the": 1,
 "cat": 2,
 "sat": 3,
 "dog": 4,
 "barked": 5
}
```

Complete the missing part of the `if` statement to ensure only unique words are added to the dictionary:

```
def create_word_index(documents):
    """
    Creates a word_index dictionary mapping unique tokens to unique integers.

    Parameters:
        documents (list of list of str): List of documents, where each document
        is a list of tokens.

    Returns:
        dict: A dictionary mapping words to unique integers.
    """
    word_index = {}
    current_index = 1

    for document in documents:
        for token in document:
            # Check if the token is already in the dictionary
            if ____________:  # Fill in the blank
                word_index[token] = current_index
                current_index += 1

    return word_index
```

What should replace the blank in the `if` statement?

- ○ a. token in `word_index`
- [✓ ● b. token not in `word_index`]
- ○ c. `current_index` in `word_index`
- ○ d. token == `word_index`

Q. 2 What is the expected result of applying the `word_index` dictionary to convert a corpus into sequences of integers?

For example, given the following corpus:

```
documents = [["the", "cat", "sat"], ["the", "dog", "barked"]]
```

And the `word_index` dictionary:

```
{
 "the": 1,
 "cat": 2,
 "sat": 3,
 "dog": 4,
 "barked": 5
}
```

What would the resulting sequences look like?

[✓ a. [[1, 2, 3], [1, 4, 5]]]
○ b. [["the", "cat", "sat"], ["the", "dog", "barked"]]
○ c. [[2, 3, 1], [4, 5, 1]]
○ d. [1, 2, 3, 4, 5]

Q. 3 Below is the algorithm for computing the co-occurrence matrix:

Algorithm 1 Getting the Co-occurrence Matrix

```
Require: sequences (list of lists of integers), context_size
Ensure: $X$ (the co-occurrence matrix)
1: Initialize matrix $X \in \mathbb{M}_{V,V}(\mathbb{R})$ with zeros
2: for all sequence in sequences do
3:     for all center_word with index $i$ in sequence do
4:         for all context_word with index $j$ in context of center_word do
5:             if $i \neq j$ then
6:                 $X[\text{center\_word}, \text{context\_word}] \leftarrow X[\text{center\_word}, \text{context\_word}] + 1$
7:             end if
8:         end for
9:     end for
10: end for
11: return $X$
```

The task is to implement this algorithm in Python. Complete the missing part of the following function to handle the update of the co-occurrence matrix.

```
def get_cooccurrence_matrix(sequences, context_size, vocab_size):
    """
    Creates a co-occurrence matrix from sequences of word indices.

    Parameters:
        sequences (list of list of int): List of sentences represented
        as lists of word indices.
        context_size (int): The size of the context window.
        vocab_size (int): The size of the vocabulary.
```

```
Returns:
    numpy.ndarray: A co-occurrence matrix of shape (vocab_size, vocab_size).
"""
import numpy as np
X = np.zeros((vocab_size, vocab_size), dtype=np.float32)

for sequence in sequences:
    for i, center_word in enumerate(sequence):
        # Define start and end of the context window
        start = max(0, i - context_size)
        end = min(len(sequence), i + context_size + 1)

        for j in range(start, end):
            if i != j:
                # Update the co-occurrence matrix
                X[center_word, sequence[j]] = __________  # Fill in the blank

return X
```

What should replace the blank to correctly implement the algorithm?

- ○ a. X[`center_word`, sequence[j]] - 1
- ● b. X[`center_word`, sequence[j]] + 1
- ○ c. X[sequence[j], `center_word`] + 1
- ○ d. X[`center_word`, `center_word`] + 1

Q. 4 What does the element X_{ij} in the co-occurrence matrix represent based on the algorithm for constructing the matrix?

- ● a. The number of times the word represented by index j appears in the context of the word represented by index i within the specified context window
- ○ b. The frequency of the word represented by index i in the entire corpus
- ○ c. The cosine similarity between the words represented by indices i and j
- ○ d. The total number of words in the sentence containing i and j

Q. 5 Given the following small corpus and `word_index`:

```
corpus = [["cat", "sat", "on", "the", "mat"],
          ["the", "cat", "is", "cute"]]

word_index = {"cat": 0, "sat": 1, "on": 2, "the": 3, "mat": 4, "is": 5, "cute": 6}
```

Suppose the context window size is 1. Calculate $X_{0,3}$, where X_{ij} is the number of times the word corresponding to index j (`"the"`) appears in the context of the word corresponding to index i (`"cat"`).

- ● a. $X_{0,3} = 2$, because the word `"the"` appears twice in the context of `"cat"` in the corpus
- ○ b. $X_{0,3} = 1$, because the word `"the"` appears only once in the context of `"cat"` in the corpus
- ○ c. $X_{0,3} = 0$, because the word `"the"` does not appear in the context of `"cat"`
- ○ d. $X_{0,3} = 3$, because the word `"the"` appears three times in the corpus

Q. 6 Recall the desired approximation:

$$\log X_{ij} \approx W_i^T \tilde{W}_j + b_i + \tilde{b}_j$$

The term $W_i^T \tilde{W}_j$ represents the relationship between the word indexed by i and the word indexed by j through their embeddings.

If we manage to achieve this approximation, what would we expect regarding the following comparisons of the dot products for the words `"king"`, `"queen"`, and `"apple"` with embeddings $W_{\text{king}}, W_{\text{queen}}, W_{\text{apple}}$?

- ● a. $W_{\text{king}}^T W_{\text{queen}} > W_{\text{king}}^T W_{\text{apple}}$
- ○ b. $W_{\text{king}}^T W_{\text{queen}} < W_{\text{king}}^T W_{\text{apple}}$
- ○ c. $W_{\text{king}}^T W_{\text{queen}} = W_{\text{king}}^T W_{\text{apple}}$
- ○ d. $W_{\text{king}}^T W_{\text{queen}} = 0$

Q. 7 Recall the cost function J:

$$J(\theta) = \sum_{i=1}^{V} \sum_{j=1}^{V} f(X_{ij})(\log X_{ij} - W_i^T \tilde{W}_j - b_i - \tilde{b}_j)^2$$

The parameters to optimize are:

- $W \in \mathcal{M}_{V,D}(\mathbb{R})$, the first embedding matrix,
- $\tilde{W} \in \mathcal{M}_{V,D}(\mathbb{R})$, the second embedding matrix,
- $b \in \mathbb{R}^V$, the bias vector for W,
- $\tilde{b} \in \mathbb{R}^V$, the bias vector for $\tilde{W}$.

What is the total number of parameters to train in the model, assuming the vocabulary size is V and the embedding dimension is D?

- ● a. $2VD + 2V$
- ○ b. $VD + V$
- ○ c. $V^2 + D^2$
- ○ d. $2V + D$

Q. 8 Consider the gradient of the cost function J with respect to W_i, given by:

$$\nabla_{W_i} J(W_i) = -2 \sum_{j'=1}^{V} f(X_{ij'}) \left(\log X_{ij'} - W_i^T \tilde{W}_{j'} - b_i - \tilde{b}_{j'} \right) \tilde{W}_{j'}.$$

What is the shape of $\nabla_{W_i} J(W_i)$?

- ● a. $\mathbb{R}^D$
- ○ b. $\mathbb{R}^V$

○ c. $\mathbb{R}^{V \times D}$

○ d. $\mathbb{R}$

Q. 9 The following pseudo-code implements gradient descent for optimizing the loss function J. Fill in the blank to correctly update the embedding matrix W during training.

Algorithm 2 Optimizing the Loss Function with Gradient Descent

```
Require: $\log X$, $f(X)$, learning rate $\eta$, number of epochs $N_{\text{epochs}}$
Ensure: $W^{(N_{\text{epochs}}-1)}, \tilde{W}^{(N_{\text{epochs}}-1)}, b^{(N_{\text{epochs}}-1)}, \tilde{b}^{(N_{\text{epochs}}-1)}$ (The trained parameters)
 1: Initialize parameters $W^{(0)}, \tilde{W}^{(0)}, b^{(0)}, \tilde{b}^{(0)}$
 2: for $t = 0$ to $N_{\text{epochs}} - 1$ do
 3:     Compute the cost $J(W^{(t)}, \tilde{W}^{(t)}, b^{(t)}, \tilde{b}^{(t)})$
 4:     for $i = 0$ to $V - 1$ do
 5:         $W_i^{(t+1)} \leftarrow \dots$                    ▷ Update $W_i$ using gradient descent
 6:     end for
 7:     for $j = 0$ to $V - 1$ do
 8:         $\tilde{W}_j^{(t+1)} \leftarrow \tilde{W}_j^{(t)} - \eta \cdot \nabla_{\tilde{W}_j} J(\tilde{W}_j^{(t)})$
 9:     end for
10:     for $i = 0$ to $V - 1$ do
11:         $b_i^{(t+1)} \leftarrow b_i^{(t)} - \eta \cdot \nabla_{b_i} J(b_i^{(t)})$
12:     end for
13:     for $j = 0$ to $V - 1$ do
14:         $\tilde{b}_j^{(t+1)} \leftarrow \tilde{b}_j^{(t)} - \eta \cdot \nabla_{\tilde{b}_j} J(\tilde{b}_j^{(t)})$
15:     end for
16: end for
```

What should replace the blank in the update step for $W_i^{(t+1)}$?

● a. $W_i^{(t)} - \eta \cdot \nabla_{W_i} J(W_i^{(t)})$

○ b. $W_i^{(t)} + \eta \cdot \nabla_{W_i} J(W_i^{(t)})$

○ c. $W_i^{(t)} - \nabla_{W_i} J(W_i^{(t)})$

○ d. $W_i^{(t)} + \nabla_{W_i} J(W_i^{(t)})$

Q. 10 After training the model using gradient descent, the algorithm outputs the embedding matrices W and $\tilde{W}$.

What do these embedding matrices represent?

● a. W and $\tilde{W}$ are matrices of word embeddings where each row corresponds to a vector representation of a word, capturing its semantic relationships with other words in the vocabulary.

○ b. W and $\tilde{W}$ are co-occurrence matrices where each element represents the number of times two words co-occur in the corpus.

○ c. W and $\tilde{W}$ are matrices of random values, used only for initializing the optimization process.

○ d. W and $\tilde{W}$ are matrices of word frequencies where each row corresponds to the total number of occurrences of a word in the corpus.