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

NAME: ____________________ **GROUP:** ____________________

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

Completely fill the circles as shown: ○○●○

Answer sheet

Q1. ○ a.
○ b.
○ c.
○ d.

Q2. ○ a.
○ b.
○ c.
○ d.

Q3. ○ a.
○ b.
○ c.
○ d.

Q4. ○ a.
○ b.
○ c.
○ d.

Q5. ○ a.
○ b.
○ c.
○ d.

Q6. ○ a.
○ b.
○ c.
○ d.

Q7. ○ a.
○ b.
○ c.
○ d.

Q8. ○ a.
○ b.
○ c.
○ d.

Q9. ○ a.
○ b.
○ c.
○ d.

Q10. ○ a.
○ b.
○ c.
○ 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 = [["machine", "learning", "is", "fun"],
             ["learning", "models", "is", "important"]]
```

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

```
{
 "machine": 1,
 "learning": 2,
 "is": 3,
 "fun": 4,
 "models": 5,
 "important": 6
}
```

```
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:
            if token not in word_index:
                word_index[token] = current_index
                __________  # Fill in the blank

    return word_index
```

What should replace the blank to correctly increment the `current_index`?

- ◯ a. `current_index = current_index`
- ◯ b. `current_index -= 1`
- ◯ c. `word_index[token] += 1`
- ◯ d. `current_index += 1`

Q. 2 After applying the `word_index` dictionary to convert a corpus into sequences, each sequence represents a document as a list of integers.

Given the following corpus:

```
documents = [["machine", "learning", "is", "fun"],
             ["learning", "models", "is", "important"]]
```

And the `word_index` dictionary:

```
{
 "machine": 1,
 "learning": 2,
 "is": 3,
 "fun": 4,
 "models": 5,
 "important": 6
}
```

What can we say about the nature of the elements in the resulting sequences?

- ○ a. Each integer in the sequences corresponds to a unique word in the `word_index` dictionary.
- ○ b. Each sequence contains random integers generated independently of the `word_index`.
- ○ c. The sequences are lists of tokens instead of integers.
- ○ d. Each integer in the sequences corresponds to the frequency of a word in the corpus.

Q. 3 In the modified algorithm for computing the co-occurrence matrix, we use a weight $\alpha(i, j)$ instead of simply adding 1 when a word at index j is in the context of the center word at index i.

The intuition behind $\alpha(i, j)$ is that words closer to the center word should contribute more to the co-occurrence count, while words farther away should contribute less. This adjustment reflects the observation that words closer in proximity often have a stronger semantic relationship.

Below is the modified pseudo-code for the algorithm:

Algorithm 1 Getting the Co-Occurrence Matrix with Distance Weighting

Require: sequences (list of lists of integers), context_size
Ensure: X (the co-occurrence matrix)

Initialize matrix $X \in \mathbb{M}_{V,V}(\mathbb{R})$ with zeros
for all sequence in sequences **do**
    for all center_word with index i in sequence **do**
        for all context_word with index j in context of center_word **do**
            if $i \neq j$ **then**
                $X[\text{center\_word}, \text{context\_word}] \leftarrow X[\text{center\_word}, \text{context\_word}] + \alpha(i, j)$
            end if
        end for
    end for
end for
return X

Which of the following best defines $\alpha(i, j)$?

- ○ a. $\alpha(i, j) = |i - j|$

- ○ b. $\alpha(i,j) = \frac{1}{|i-j|}$
- ○ c. $\alpha(i,j) = \max(0, i-j)$
- ○ d. $\alpha(i,j) = 1$

Q. 4 The following function computes the co-occurrence matrix with weighted contributions based on the inverse distance between words. The formula for the left context has already been implemented using $i - j$. Complete the blank in the right context to correctly compute the inverse distance.

```
def get_occurence_matrix(sentences, context_size, vocabulary_size):
    """
    This function creates the co-occurrence matrix from the corpus
    composed of sentences.
    """
    X = np.zeros((vocabulary_size, vocabulary_size))
    N = len(sentences)
    print("number of sentences to process:", N)
    it = 0
    for sentence in sentences:
        it += 1
        if it % 10000 == 0:
            print("processed", it, "/", N)
        n = len(sentence)
        for i in range(n):
            # center word
            w_i = sentence[i]

            start = max(0, i - context_size)
            end = min(n - 1, i + context_size)

            # left context side
            for j in range(start, i):
                # context word
                w_j = sentence[j]
                # inverse of distance between w_i and w_j
                inverse_distance = 1. / (i - j)
                # Add the inverse of the distance to X[w_i, w_j]
                X[w_i, w_j] += inverse_distance

            # right context side
            for j in range(i + 1, end + 1):
                # context word
                w_j = sentence[j]
                # inverse of distance between w_i and w_j
                inverse_distance = __________  # Fill in the blank
                # Add the inverse of the distance to X[w_i, w_j]
                X[w_i, w_j] += inverse_distance
    return X
```

What should replace the blank ?

- ○ a. $1./(j-i)$
- ○ b. $1./(i-j)$
- ○ c. 1.
- ○ d. $1./(i+j)$

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

```
corpus = [["dog", "barked", "at", "the", "mailman"],
          ["the", "dog", "is", "friendly"]]

word_index = {"dog": 0, "barked": 1, "at": 2, "the": 3,
              "mailman": 4, "is": 5, "friendly": 6}
```

Suppose the context window size is 3. 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 (`"dog"`).

- ○ a. $X_{0,3} = 0$
- ○ b. $X_{0,3} = 1$
- ○ c. $X_{0,3} = 2$
- ○ d. $X_{0,3} = 0$

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.

Assume that we have trained embeddings $W_{\text{equity}}, W_{\text{market}}$ using this approximation and compare the dot product of these embeddings to the dot product of one-hot vectors for the same words.

Which of the following best describes the comparison?

- ○ a. The dot product $W_{\text{equity}}^T W_{\text{market}}$ from embeddings is positive and large, while the dot product of their one-hot vectors is 0.
- ○ b. The dot product $W_{\text{equity}}^T W_{\text{market}}$ from embeddings is 0, and the dot product of their one-hot vectors is also 0.
- ○ c. The dot product $W_{\text{equity}}^T W_{\text{market}}$ from embeddings is smaller than the dot product of their one-hot vectors, which is positive.
- ○ d. The dot product $W_{\text{equity}}^T W_{\text{market}}$ from embeddings is negative, while the dot product of their one-hot vectors is 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 Which of the following equations correctly represents the gradient $\nabla_{\tilde{W}_j} J(\tilde{W}_j)$ based on its shape?

○ a. $\nabla_{\tilde{W}_j} J(\tilde{W}_j) = -2\sum_{i'=1}^{V} f(X_{i'j})\left(\log X_{i'j} - W_{i'}^T\tilde{W}_j - b_{i'} - \tilde{b}_j\right)$

○ b. $\nabla_{\tilde{W}_j} J(\tilde{W}_j) = -2\sum_{i'=1}^{V} f(X_{i'j})\left(\log X_{i'j} - W_{i'}^T\tilde{W}_j - b_{i'} - \tilde{b}_j\right) W_{i'}$

○ c. $\nabla_{\tilde{W}_j} J(\tilde{W}_j) = -2\sum_{i'=1}^{V} f(X_{i'j})\left(\log X_{i'j} - W_{i'}^T\tilde{W}_j - b_{i'} - \tilde{b}_j\right) W_{i'}W_{i'}^T$

○ d. $\nabla_{\tilde{W}_j} J(\tilde{W}_j) = 0$

Q. 9 In the iterative optimization method where gradients are set to zero, the update equations for the parameters $W, \tilde{W}, b, \tilde{b}$ are given as:

$$W_i^{(t+1)} \longleftarrow \left(\sum_{j'=1}^{V} f(X_{ij'})\tilde{W}_{j'}^{(t)}\tilde{W}_{j'}^{(t)^T}\right)^{-1}\left(\sum_{j'=1}^{V} f(X_{ij'})(\log X_{ij'} - b_i^{(t)} - \tilde{b}_{j'}^{(t)})\tilde{W}_{j'}^{(t)}\right) \quad (1)$$

$$\tilde{W}_j^{(t+1)} \longleftarrow \left(\sum_{i'=1}^{V} f(X_{i'j})W_{i'}^{(t)}W_{i'}^{(t)^T}\right)^{-1}\left(\sum_{i'=1}^{V} f(X_{i'j})(\log X_{i'j} - b_{i'}^{(t)} - \tilde{b}_{j}^{(t)})W_{i'}^{(t)}\right) \quad (2)$$

$$b_i^{(t+1)} \longleftarrow \left(\sum_{j'=1}^{V} f(X_{ij'})\right)^{-1}\left(\sum_{j'=1}^{V} f(X_{ij'})(\log X_{ij'} - W_i^{(t)^T}\tilde{W}_{j'}^{(t)} - \tilde{b}_{j'}^{(t)})\right) \quad (3)$$

$$\tilde{b}_j^{(t+1)} \longleftarrow \left(\sum_{i'=1}^{V} f(X_{i'j})\right)^{-1}\left(\sum_{i'=1}^{V} f(X_{i'j})(\log X_{i'j} - W_{i'}^{(t)^T}\tilde{W}_{j}^{(t)} - b_{i'}^{(t)})\right) \quad (4)$$

Which of the following best describes the interdependence of these update equations?

○ a. The update equations for W and $\tilde{W}$ are independent of b and $\tilde{b}$, so these parameters can be updated in parallel.

○ b. The parameter updates depend only on the values of $f(X)$ and $\log X$, making the updates independent of each other.

○ c. Each parameter update depends on the values of the other parameters at the current iteration t, making it necessary to update all parameters iteratively until convergence.

○ d. All parameters can be updated simultaneously in a single step without the need for iteration, as the update equations guarantee immediate convergence.

Q. 10 The following pseudo-code implements the alternating least squares method to optimize the loss function by iteratively updating the parameters $W, \tilde{W}, b, \tilde{b}$. Complete the missing part of the pseudo-code for updating $W_i^{(t+1)}$.

Algorithm 2 Training by Alternating Least Squares

Require: $\log X$, $f(X)$, number of epochs N_{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)

Initialize randomly the parameters $W^{(0)}$, $\tilde{W}^{(0)}$, $b^{(0)}$, $\tilde{b}^{(0)}$
costs $\leftarrow$ []
for $t = 0$ **to** $N_{\text{epochs}} - 1$ **do**
    Calculate the cost as a function of $W^{(t)}$, $\tilde{W}^{(t)}$, $b^{(t)}$, $\tilde{b}^{(t)}$ and append to costs
    for $i = 0$ **to** $V - 1$ **do**
        $W_i^{(t+1)} \leftarrow \ldots$ ▷ Fill in the blank
    end for
    for $j = 0$ **to** $V - 1$ **do**
        $\tilde{W}_j^{(t+1)} \leftarrow \left(\sum_{i'=1}^{V} f(X_{i'j}) W_{i'}^{(t)} W_{i'}^{(t)\top}\right)^{-1} \left(\sum_{i'=1}^{V} f(X_{i'j})(\log X_{i'j} - b_{i'}^{(t)} - \tilde{b}_j^{(t)}) W_{i'}^{(t)}\right)$
    end for
    for $i = 0$ **to** $V - 1$ **do**
        $b_i^{(t+1)} \leftarrow \left(\sum_{j'=1}^{V} f(X_{ij'})\right)^{-1} \left(\sum_{j'=1}^{V} f(X_{ij'})(\log X_{ij'} - W_i^{(t)\top} \tilde{W}_{j'}^{(t)} - \tilde{b}_{j'}^{(t)})\right)$
    end for
    for $j = 0$ **to** $V - 1$ **do**
        $\tilde{b}_j^{(t+1)} \leftarrow \left(\sum_{i'=1}^{V} f(X_{i'j})\right)^{-1} \left(\sum_{i'=1}^{V} f(X_{i'j})(\log X_{i'j} - W_{i'}^{(t)\top} \tilde{W}_j^{(t)} - b_{i'}^{(t)})\right)$
    end for
end for
return $W^{(N_{\text{epochs}}-1)}$, $\tilde{W}^{(N_{\text{epochs}}-1)}$, $b^{(N_{\text{epochs}}-1)}$, $\tilde{b}^{(N_{\text{epochs}}-1)}$

Which of the following correctly fills the blank for $W_i^{(t+1)}$?

○ a. $W_i^{(t+1)} \leftarrow W_i^{(t)} - \eta \cdot \nabla_{W_i} J(W_i^{(t)})$

○ b. $W_i^{(t+1)} \leftarrow \left(\sum_{i'=1}^{V} f(X_{i'j}) W_{i'}^{(t)} W_{i'}^{(t)T}\right)^{-1} \left(\sum_{i'=1}^{V} f(X_{i'j})(\log X_{i'j} - b_{i'}^{(t)} - \tilde{b}_j^{(t)}) W_{i'}^{(t)}\right)$

○ c. $W_i^{(t+1)} \leftarrow \left(\sum_{j'=1}^{V} f(X_{ij'}) \tilde{W}_{j'}^{(t)} \tilde{W}_{j'}^{(t)T}\right)^{-1} \left(\sum_{j'=1}^{V} f(X_{ij'})(\log X_{ij'} - b_i^{(t)} - \tilde{b}_{j'}^{(t)}) \tilde{W}_{j'}^{(t)}\right)$

○ d. $W_i^{(t+1)} \leftarrow W_i^{(t)} + \eta \cdot f(X_{ij}) \cdot \tilde{W}_j$