深圳大学期末考试试卷

开/闭卷 开卷 A/B 卷 _____

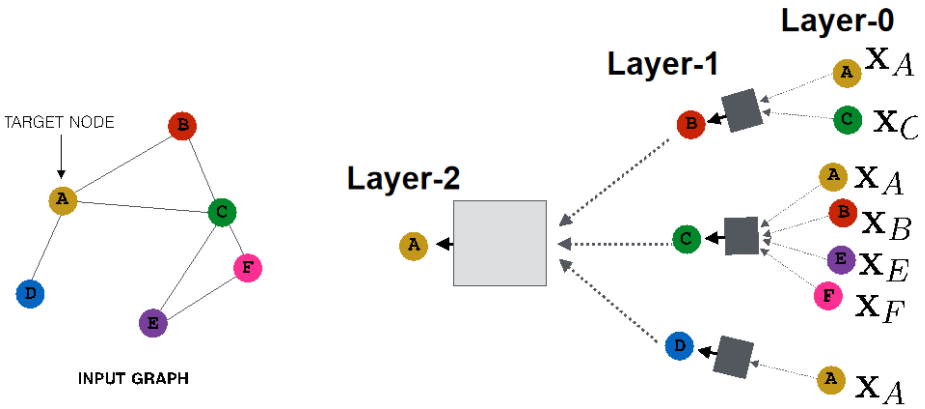
课程编号 1504660001 课序号 01 课程名称 自然语言处理 学分 2.5

命题人(签字)_____ 审题人(签字)_____ _____年____月____日

题号	一	二	三	四	五	六	七	八	九	十	基本题总分	附加题
得分												
评卷人												

一、随堂测试 4：基于 GNN 信息聚合机制的文本建模计算案例 (共 5 题，每道 20 分，共 100 分)

（姓名： 学号：）



提示图

文档： "香蕉和樱桃是我最喜欢的水果之一。"

(1) 对文档 1："香蕉和樱桃是我最喜欢的水果之一。"，定义滑动窗口，获得单词的邻接矩阵

```
import numpy as np
import jieba

# 1. 文本分词
text = "香蕉和樱桃是我最喜欢的水果之一。"
words = list(jieba.lcut(text))
words = [w for w in words if w not in ['。']]
n_words = len(words)
print("分词结果：", words)
print("单词数量：", n_words)
# 2. 定义滑动窗口（窗口大小=2，左右各 1 个邻居）
window_size = 2  # 窗口大小=左右邻居数+1（自身）
adj = np.zeros((n_words, n_words), dtype=int)  # 邻接矩阵 n×n

for i in range(n_words):
    # 窗口范围：[max(0, i-1), min(n_words-1, i+1)]（左右各 1 个邻居）
    start = max(0, i - (window_size // 2))
    end = min(n_words - 1, i + (window_size // 2))
    for j in range(start, end + 1):
```

```
            if i != j:  # 排除自身（可根据需求调整是否包含）
                adj[i][j] = 1
print("邻接矩阵：\n", adj)

cherry_idx = words.index("樱桃")
print("樱桃的索引：", cherry_idx)
print("樱桃的邻居索引：", np.where(adj[cherry_idx] == 1)[0])
```

(2) 随机初始化所有参数，包括 X、W_1、W_2、B_1, B_2，参数的维度自拟。

```
# 3. 初始化参数
embed_dim = 5  # 单词嵌入维度
hidden1_dim = 8  # Layer1 隐藏层维度
hidden2_dim = 10  # Layer2 隐藏层维度

# 单词嵌入矩阵 X: (n_words, embed_dim)
np.random.seed(42)
X = np.random.randn(n_words, embed_dim)  # 正态分布初始化

# 权重矩阵与偏置（线性变换参数）
W1 = np.random.randn(embed_dim, hidden1_dim)  # Layer1 权重: (embed_dim, hidden1_dim)
B1 = np.random.randn(hidden1_dim)  # Layer1 偏置: (hidden1_dim,)
W2 = np.random.randn(hidden1_dim, hidden2_dim)  # Layer2 权重: (hidden1_dim, hidden2_dim)
B2 = np.random.randn(hidden2_dim)  # Layer2 偏置: (hidden2_dim,)

print("X（单词嵌入）形状：", X.shape)
print("W1 形状：", W1.shape, " B1 形状：", B1.shape)
print("W2 形状：", W2.shape, " B2 形状：", B2.shape)
```

假设以“樱桃”单词为例的聚合计算

(3) 写出 Layer-0 中，h0_`樱桃` 是多少

```
# 4. Layer0：输入层，节点表示=单词嵌入（无聚合）
h0 = X  # h0: (n_words, embed_dim)
h0_cherry = h0[cherry_idx]  # 樱桃的 Layer0 表示

print("Layer0 樱桃的表示 h0_樱桃：\n", h0_cherry)
print("h0_樱桃形状：", h0_cherry.shape)
```

(4) 计算 Layer-1 中相关的节点的表示向量，聚合函数的方式可以为 Mean 或其它。

```
# 5. Layer1：Mean 聚合 + 线性变换 + Sigmoid 激活
def mean_aggregate(h_prev, adj, node_idx):
    """Mean 聚合函数：计算节点邻居的前一层表示均值"""
    # 找到节点的邻居索引
    neighbor_idxs = np.where(adj[node_idx] == 1)[0]
    # 聚合邻居表示（均值）
    aggregated = np.mean(h_prev[neighbor_idxs], axis=0)
    return aggregated

def sigmoid(x):
    """Sigmoid 激活函数（提示图指定）"""
```

```
    return 1 / (1 + np.exp(-x))

# 计算樱桃的 Layer1 表示 h1_cherry
aggregated_h0 = mean_aggregate(h0, adj, cherry_idx)  # 邻居 h0 均值聚合
h1_cherry = sigmoid(np.dot(aggregated_h0, W1) + B1)  # 线性变换+激活

# 计算所有节点的 Layer1 表示（可选，用于 Layer2 聚合）
h1 = np.zeros((n_words, hidden1_dim))
for i in range(n_words):
    agg = mean_aggregate(h0, adj, i)
    h1[i] = sigmoid(np.dot(agg, W1) + B1)

print("Layer1 樱桃的聚合向量（邻居 h0 均值）：\n", aggregated_h0)
print("Layer1 樱桃的表示 h1_cherry：\n", h1_cherry)
print("h1_cherry 形状：", h1_cherry.shape)
```

(5) 计算 Layer-2 中，节点‘樱桃’的表示向量 h2_`樱桃` 是多少？

```
# 6. Layer2：Mean 聚合 + 线性变换 + Sigmoid 激活
# 聚合 Layer1 中樱桃的邻居表示
aggregated_h1 = mean_aggregate(h1, adj, cherry_idx)  # 邻居 h1 均值聚合
h2_cherry = sigmoid(np.dot(aggregated_h1, W2) + B2)  # 线性变换+激活

print("Layer2 樱桃的聚合向量（邻居 h1 均值）：\n", aggregated_h1)
print("Layer2 樱桃的表示 h2_cherry：\n", h2_cherry)
print("h2_cherry 形状：", h2_cherry.shape)
```

提示内容：

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right)$$

$$\sigma = \frac{1}{1+e^{-x}}$$

- **Mean:**

$$\text{AGG} = \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|}$$

- **Pool**
  - Transform neighbor vectors and apply symmetric vector function.

element-wise mean/max

$$\text{AGG} = \gamma\left(\{\mathbf{Q}\mathbf{h}_u^{k-1}, \forall u \in N(v)\}\right)$$

- **LSTM:**
  - Apply LSTM to random permutation of neighbors.

$$\text{AGG} = \text{LSTM}\left([\mathbf{h}_u^{k-1}, \forall u \in \pi(N(v))]\right)$$