CS365 Written Assignment 4

Khoa Cao

Due December 5, 2025

Question 1

Consider the PageRank transform defined below, for $(0 \leq \alpha \leq 1)$:

$$\mathbf{M'} = \alpha\mathbf{M} + (1-\alpha)\frac{1}{|V|}\mathbf{1}$$

To augment this so that webpages containing a keyword k are ranked higher, we can modify the transform to:

$$\mathbf{M}' = \alpha\mathbf{M} + (1-\alpha)\mathbf{K}$$

where $\mathbf{K}$ is a matrix defined as follows:

$$\mathbf{K}_{ij} = \begin{cases} \frac{2}{N_k+N} & \text{if } k \in v_j \\ \frac{1}{N_k+N} & \text{otherwise} \end{cases}$$

where N is the total number of webpages, and N_k is the number of webpages containing keyword k.

First, we will prove that $\mathbf{K}$ is a stochastic matrix:

$$\begin{aligned}
\sum_{i=1}^{N} \mathbf{K}_{ij} &= \sum_{i:k\in v_j} \frac{2}{N_k+N} + \sum_{i:k\notin v_j} \frac{1}{N_k+N} && \text{definition of } \mathbf{K} \\
&= N_k \cdot \frac{2}{N_k+N} + (N-N_k)\cdot\frac{1}{N_k+N} && \text{definition of } N_k \\
&= \frac{2N_k+N-N_k}{N_k+N} && \text{algebra} \\
&= \frac{N_k+N}{N_k+N} = 1 && \text{algebra} \\
&\Rightarrow \mathbf{K} \text{ is stochastic}
\end{aligned}$$

Question 2

To use a binary classifier to perform multi-class classification, we can use a one-vs-rest approach. For each class C_i, define a indicator variable y_i:

$$y_i = \begin{cases} 1 & \text{if } C_i \text{ is the correct class} \\ 0 & \text{otherwise} \end{cases}$$

Then, we can train a binary classifier for each class C_i using the indicator variable y_i as the label. For n classes, we train n binary classifiers, one for each class i. Each binary classifier, h_i, will learn the probability

$$Pr[y_i = 1|X]$$

To classify a new instance, we can run it through all n classifiers and choose the class with the highest probability. This approach works for any number of classes because each binary classifier is independent of the others and we can train as many classifiers as needed.

Question 3

Given the following class distribution:

Class c	$Pr[c]$
Red	0.05
Blue	0.87
Green	0.08

To determine what metric we would avoid when evaluating a model on this dataset, we first formally define when a metric is bad. A metric is bad if it does not reflect the true performance of the model on the dataset, formally defined as follows:

> A metric m is bad if there exists a model M such that $m(M)$ is high, but the true performance of M on the dataset is low.

Using this definition, we can analyze common metrics used in multi-class classification:

- **Accuracy**: This metric is bad for this dataset because a model that always predicts the majority class (Blue) will have an accuracy of 87%, but will have poor performance on the minority classes (Red and Green).
- **Precision, Recall, F1-Score**: These metrics are not bad for this dataset because they take into account the performance on each class individually and penalize models that perform poorly on minority classes.

Therefore, the metric we would avoid when evaluating a model on this dataset is **Accuracy**.

Question 4

This problem is analogous to finding the maximum k such that the k-core of a graph is non-empty. We can use the following algorithm to find the maximum k-core of a graph:

Algorithm 1: Find Maximum k-core of a Graph

```
Input: Graph G = (V, E) as adjacency list adj
Output: Maximum k-core of G
k ← 0;
deg ← {};
for each vertex v ∈ V do
    deg[v] ← degree(v);
end
max_deg ← maximum degree in deg;
res ← ∅;
for k ← 0 to max_deg do
    adj' ← adj;
    deg' ← deg;
    while adj ≠ ∅ do
        removed ← false;
        for each vertex v in adj do
            if deg'[v] < k then
                for u ∈ adj[v] do
                    deg'[u] ← deg'[u] − 1;
                end
                remove v from adj';
                removed ← true;
            end
        end
        if !removed then
            break;
        end
    end
    if adj' ≠ ∅ then
        res ← adj';
    end
end
return res;
```

Claim. *The algorithm above runs in $O(|V| \cdot (|V| + |E|))$ time.*

Proof. The algorithm has an outer loop that runs at most max_deg times, which is at most $|V| - 1$. Inside the outer loop, there is a while loop removes vertices with degree less than k until no more verticies can be removed. This loop iterates over all vertices and edges in the graph, taking $O(|V| + |E|)$ time. Thus, the total time complexity of the algorithm is $O(|V| \cdot (|V| + |E|))$. In a connected graph, $|E| \geq |V| - 1$, so the time complexity can be simplified to $O(|V| \cdot |E|)$. □

Claim. *The algorithm above correctly finds the maximum k-core of a graph.*

Proof. By definition, a k-core is a maximal subgraph in which every vertex has degree at least k. Therefore, in a k-core,

$$\min_{vinG_k} deg(v, G_k) \geq k$$

The algorithm iteratively removes vertices with degree less than k until no more vertices can be removed. At this point, the remaining graph is a k-core. We increment k until $k = max\_deg$. Since the maximum degree of a subgraph cannot exceed the maximum degree of the original graph, the last non-empty subgraph found is the maximum k-core of the graph. □