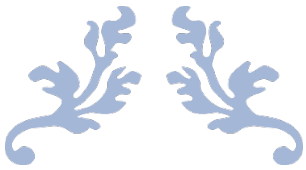


HOMEWORK 0

Jie Tang

SEPTEMBER 2, 2020

Q1. Read the textbook Chapter Page 21-40. (5 points)

Summary :

1. There are several distributed file systems of the type:
    (a): The Google File System (GFS)
    (b): Hadoop Distributed File System (HDFS)
    (c): CloudStore, an open-source DFS originally developed by Kosmix

2. If you have only small files, there is no point using a DFS for them. Files are rarely updated when we use DFS.

3. A chunk is a collection of elements, and no element is stored across two chunks. Each key is assigned to one and only one Reduce task

4. We could apply the Reduce function within the Map task, before the output of the Map tasks is subject to grouping and aggregation.

5. There is often significant variation in the lengths of the value lists for different keys, so different reducers take different amounts of time.

6. Matrix-vector and matrix-matrix calculations fit nicely into the MapReduce style of computing.

7. A relation, however large, can be stored as a file in a distributed file system. The elements of this file are the tuples of the relation.

Q2. Write your own code of K-means clustering. Do not copy online codes. Test it on iris dataset, calculate the NMI. (15 points)

1. The most common type of k-means is Lloyd's algorithm. The following is the algorithm's logic:

(a) Select K initial centroids
(b) Repeat
i. Assign each data point to its nearest centroid
ii. Update and recompute centroids and clusters
(c) Until centroid don't change

So we are essentially solving a minimization problem, where each data point gets assigned to its closest centroid each iteration, until the centroid don't change.

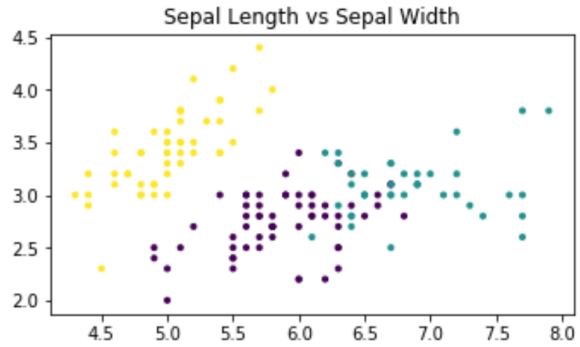
2. Normalized Mutual Information (NMI)

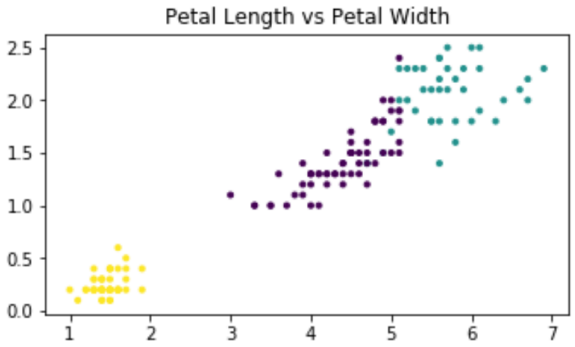
To calculate NMI, entropy is needed. So, if given the ground truth labels, denote them as Y, and the clustering labels, denote them as C, the normalized mutual information is given by:

$$NMI(Y, C) = \frac{2 * MI(Y,C)}{H(Y) + H(C)}$$

Where H(Y) = Entropy(Y), and MI is Mutual Information.

3. Below is the IRIS dataset visualization:





4**. Based on the codes, the NMI is 0.742, which is quite okay.**

Q3. Write your own code of K-nearest neighbor classifier. Do not copy online codes. Test it on iris dataset, calculate the accuracy. (15 points)

The KNN algorithm logic is like following:

(a) Initialize K to your chosen number of neighbors
(b) For each example in the data
i. Calculate the distance between the query example and the current example from the data.
ii. Add the distance and the index of the example to an ordered collection
(c) Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
(d) Pick the first K entries from the sorted collection
(e) Get the labels of the selected K entries
(f) If regression, return the mean of the K labels
(g) If classification, return the mode of the K labels

Based on the codes, I got 95% accuracy using KNN.

Q4. Conduct SVD on $A = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$. (15 points)

Singular value decomposition takes a rectangular matrix data (defined as A, where A is a n x p matrix). The SVD theorem states:

$$\mathbf{A}_{nxp} = \mathbf{U}_{nxn}\, \mathbf{S}_{nxp}\, \mathbf{V}^{\mathbf{T}}_{pxp}$$

Where:
$\mathbf{U}^T\mathbf{U} = \mathbf{I}_{nxn}$
$\mathbf{V}^T\mathbf{V} = \mathbf{I}_{pxp}$ (i.e. U and V are orthogonal)
S (the same dimensions as A) has singular values and is diagonal

Calculating the SVD consists of finding the eigenvalues and eigenvectors of AA^T and A^TA. The eigenvectors of A^TA make up the columns of V , the eigenvectors of AA^T make up the columns of U.

Also, the singular values in **S** are square roots of eigenvalues from AA^T or A^TA. The singular values are the diagonal entries of the S matrix and are arranged in descending order. The singular values are always real numbers. If the matrix A is a real matrix, then U and V are also real.

$AA^T = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$

Eigenvalues: λ1 = λ2 = 2,

Eigenvector: $K*\begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $K*\begin{pmatrix} 0 \\ 1 \end{pmatrix}$. K is not equal to 0.

$A^TA = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$

Eigenvalues: λ=0 with multiplicity of 2, λ=2 with multiplicity of 2.

Eigenvector:

λ=0, $K*\begin{pmatrix} -1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$, $k*\begin{pmatrix} 0 \\ -1 \\ 0 \\ 1 \end{pmatrix}$; **λ=2,** $K*\begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$, $K*\begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$. K is not equal to 0.

So: $U_{nxn} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $S = \begin{pmatrix} \sqrt{2} & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$, $V^T_{pxp} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix}$,