Math 425   Spring 2023   Project 1   due: 5PM on Fri April 28

Project Directions

- Include a report on every group member's contribution.
- Submit the group's well commented code used for the project with instructions on how to compile and run.
- Make a **15** to **20** minute video presentation of your results.

The project consists of three problems

Problem 1

The file `top250movies.txt` contains data from the 250 top-rated movies according to IMDB.[1] Each line in the file lists a movie title and its cast as `title/actor1/actor2/...`, with the actors listed mostly in billing order (stars first), though some casts are listed alphabetically or in order of appearance.

Create a network with a node for each actor in the file. The weight from actor a to actor b should be the number of times that actor a and b were in a movie together but actor b was listed first. That is, **edges point to higher-billed actors**. A portion of the network is shown in Figure 1 below which shows that Michael Caine was in four movies with Christian Bale where Christian Bale was listed first in the cast. Compute the ordered PageRank vector for the network and return the list of the ranked actors. (Use `encoding="utf-8"` as an argument to open and read the file, since several actors and actresses have nonstandard characters in their names such as ø and æ.)

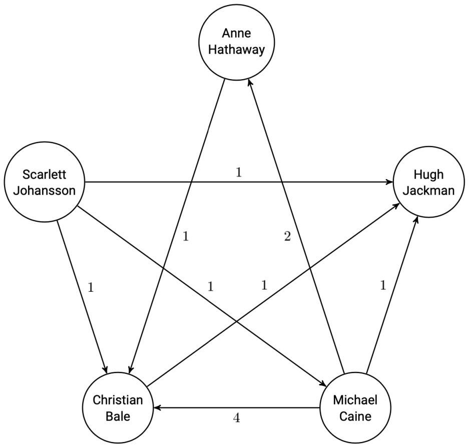

Figure 1: A portion of the network for actors in the top 250 movies according to IMDB. Michael Caine was in four movies with Christian Bale where Christian Bale was listed first in the cast.

[1] https://www.imdb.com/search/title/?groups=top_250&sort=user_rating

Problem 2

You are given part of the Wisconsin Diagnostic Breast Cancer (WDBC) dataset[2]. For each patient, you are given a vector **a** giving features computed from digitized images of a fine needle aspirate (FNA) of a breast mass for that patient. The features describe characteristics of the cell nuclei present in the image. The goal is to decide whether the cells are malignant or benign.

Here is a brief description of the way the features were computed. Ten real-valued quantities are computed for each cell nucleus:

- radius (mean of distances from center to points on the perimeter)
- texture (standard deviation of gray-scale values)
- perimeter
- area
- smoothness (local variation in radius lengths)
- compactness (perimeter^2 / area - 1.0)
- concavity (severity of concave portions of the contour)
- concave points (number of concave portions of the contour)
- symmetry
- fractal dimension ("coastline approximation" - 1)

The mean, standard error (stderr), and a measure of the largest (worst) (mean of the largest values) of each of the features were computed for each image. Thus each specimen is represented by a vector **a** with thirty entries. The domain D consists of thirty strings identifying these features, e.g. `‘‘radius (mean)"`, `‘‘ radius (stderr)"`, `‘‘radius (worst)"`, `‘‘area (mean)"`, and so on. Two files are provided containing data, `train.data` and `validate.data`. Also provided is the module `efficient_cancer_data`.
The procedure in `read_training_data` in the `efficient_cancer_data` module takes a single argument, a string giving the pathname of a file. It reads the data in the specified file and returns a pair $(A, \mathbf{b})$ where:

- A is a matrix whose rows correspond to the data for each patient in the data set. The elements in a row correspond to the 30 features measured for a patient.
- $\mathbf{b}$ is a vector whose domain is the set of patients and $\mathbf{b}[\mathbf{r}]$ is 1 if the specimen of patient r is malignant and it's -1 if the specimen is benign.

Use `read_training_data` to read the data in the file `train.data` into the variables A, $\mathbf{b}$.

(a) Use the QR algorithm to find the least-squares linear model for the data.

[2] (https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic))

(b) Apply the linear model from (a) to the data set `validate.data` and predict the malignancy of the tissues. You will have to define a classifier function

$$C(\mathbf{y}) = \begin{cases} +1 & \text{if the prediction is non-negative} \\ -1 & \text{otherwise} \end{cases}$$

(c) What is the percentage of samples that are incorrectly classified? Is it greater or smaller than the success rate on the training data?

Problem 3

Classification of Handwritten Digits

On iLearn you will find the following data files of handwritten digits[3]:

- `handwriting_training_set.txt`: 4000 training examples of handwritten digits. Each training example is a 20 pixel by 20 pixel grayscale image of a digit reshaped into a 400-dimensional vector. Each pixel is represented by a floating point number that indicates the grayscale intensity at that location. Thus the set is a 4000 by 400 matrix.
- `handwriting_training_set_labels.txt`: This data set contains the labels of the corresponding digits in the training set. The digits "1" to "9" are labeled as they are. However, because MATLAB has no zero index, the digit zero is represented as the value ten, i.e. "0" is labeled as "10."
- `handwriting_test_set.txt`: 1000 test set of handwritten digits with the same format as the training set. Thus this set is a 1000 by 400 matrix.
- `handwriting_test_set_labels.txt`: The labels for the test set.

A. Construct an algorithm for classification of handwritten digits. Use the training set and compute the SVD of each class/digit matrix. Note that in the training set, the first 400 are examples of the digit 0, the next 400 are examples of the digit 1, etc.. Identify the unknown test digits by using the singular value decomposition of the each digit matrix. Do the classification using 5, 10, 15 and 20 singular vectors as a basis.

SPECIFIC TASKS:

i. Give a table or graph of the percentage of correctly classified digits as a function of the number of basis vectors.

ii. Check if all digits are equally easy or difficult to classify. Also look at some of the difficult ones, and see that in may cases they are very badly written.

iii. Check the singular values of the different classes. Is there evidence to support using different number of basis for different digits?

[3] This is a subset of the MNIST handwritten digit dataset (http://yann.lecun.com/exdb/mnist)

B. Implement the following **two-stage** algorithm:

In the first stage compare the unknown digit only to the first singular vector in each class. If for one class/digit the residual is significantly smaller than for the others, classify as that digit. Otherwise perform the algorithm above. Is it possible to get as good a result for this version? How frequently is the second stage necessary?