HW2

High-dimensional data analysis

Due: 18/6/2021

Question 1 (30 points)

1. A matrix A is given. We look at $\|A-\sigma_1 u_1 v_1^T\|$, when $\sigma_1 u_1 v_1^T$ is the largest rank-1 piece of A.

   - What is the ℓ_2 norm of $\|A-\sigma_1 u_1 v_1^T\|$?
   - What is the Nuclear norm of $\|A-\sigma_1 u_1 v_1^T\|$?
   - What the Frobenius norm of $\|A-\sigma_1 u_1 v_1^T\|$?
     This one is a bit tricky.. Use these two facts.
     1: $\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{i=1}^{n}|a_{ij}|} = \sqrt{\text{trace}(A^T A)}$.
     2: the trace of similar matrices is identical: If $A = MBM^{-1}$, then trace(A) = trace(B).

2. A=S is a symmetric matrix, thus $A = S = Q^T \Lambda Q$. What is the closest A_1 matrix to A, in terms of the Eckard-Young Theorem. Use the ℓ_2 matrix norm for $\|A - A_1\|$.

3. The Geometry of SVD: We saw in class that for $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ the 4 parameters a,b,c, and d are translated into two rotation parameters for the matrices U and V: θ_1 and θ_2, and the two stretching parameters σ_1^2 and σ_2^2 in Σ.

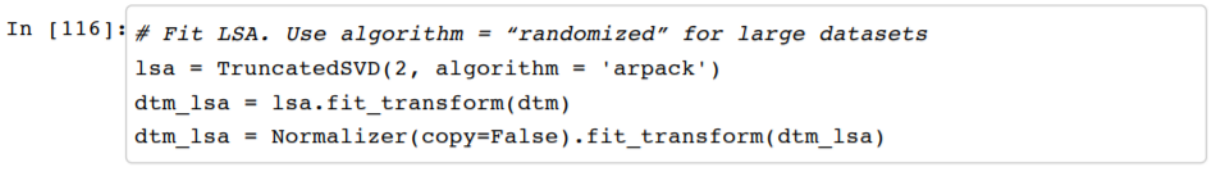
   Now, consider A of size 2×3.

   - What are the sizes of U, Σ and V?
   - How are the θs and σs distributed between U, V and Σ in this case? Explain your answer. (Tip: think how many parameters/angles are needed to define a 3×3 rotation.)

Question 2 (30 points)

Implement Latent Semantic Indexing (LSI).

- Review the example we saw in class (LSI SVD Example in the Moodle).
- Use the example in the file named **LSICodeExample.pdf** as a base for this question.
- Replace the lines that are shown in Figure 1 with a standard SVD function (such as scipy.sparse.linalg.svds, for example). This function returns the three matrices U S and V.
- Use the output from the SVD to embed the words and the documents in the same 2-dimensional embedding, like presented in Figure 2.
- Write a function that receives 2 words query (out of the words in the example), locates the query in the embedded space (as a red x) and returns the name of the closest documents. See Figure 3, in which the 2 query words are circled in green, the embedding of the quarry is a red x and the closest document is circled in red.



```
In [116]: # Fit LSA. Use algorithm = "randomized" for large datasets
          lsa = TruncatedSVD(2, algorithm = 'arpack')
          dtm_lsa = lsa.fit_transform(dtm)
          dtm_lsa = Normalizer(copy=False).fit_transform(dtm_lsa)
```

Figure 1: Code to replace

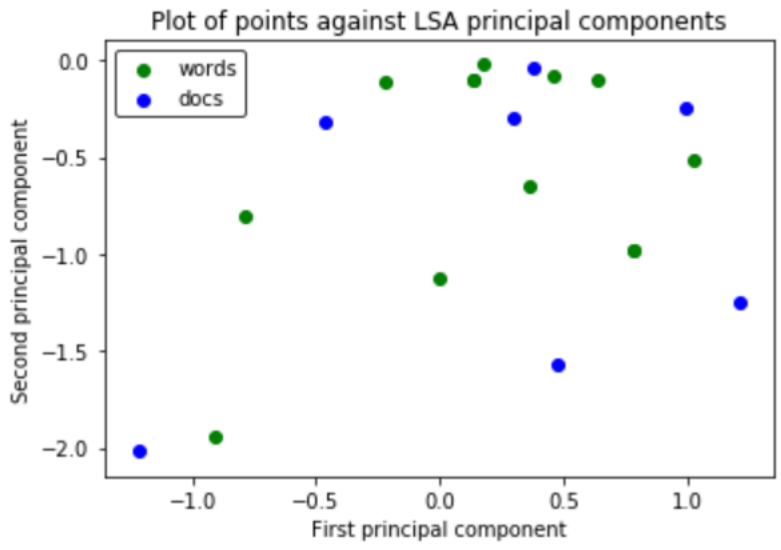


Figure 2: Joint word-doc embedding

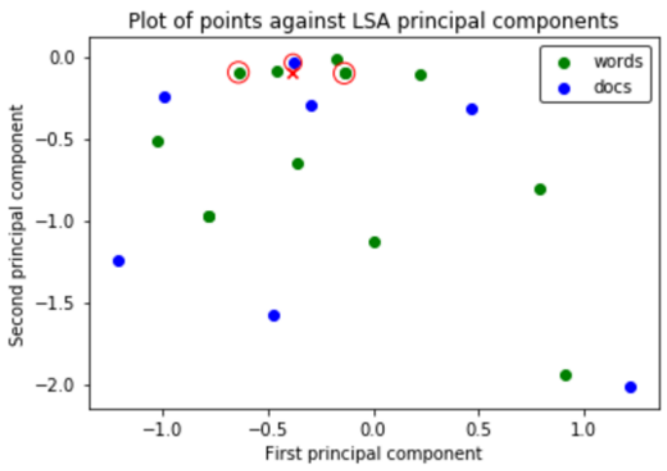


Figure 3: Embedding of the quarry and result

Question 3 (20 points)

Diffusion maps: Diffusion maps is easy to implement. The first step will be to write a **'myDM'** function. Then, run it on a simple dataset.

- Write a function named **'myDM'** that receives as input: a dataset **data**, a parameter **C** that will be used to multiply ϵ and the reduced desired dimension **d**.
- Use the functions pdist and squareform (from scipy) to create a square matrix of Euclidean distances between all of your data points (squareform(pdist(data))).
- Compute ϵ for the Gausian kernel using the max-min heuristic.
  - Use the dist matrix for this calculation.
  - The diagonal of dist is 0's. Replace it with the maximum value of the matrix.
  - Then, find the minimum value of each row in the modified dist.
  - Last, set ϵ to be the maximum value of the minimums.
- Compute $W = exp(\frac{-\text{dist}^2}{C\epsilon})$.
- Make W a Markov matrix. Denote the result by P.
- Compute the spectral decomposition of P with SVD.
- Sort the singular values and vectors by descending order.
- Return the first **d** Diffusion maps coordinates (ignore the first singular vector) by multiplying the the left singular vectors with their associated singular values.

(a) Load the dataset **two circles data.csv** and plot it.

(b) Apply your DM function to this dataset and plot the result of the first 3 diffusion maps coordinates.

(c) Apply K-means with K=2 to the original dataset and plot the result (each cluster plotted with a different color).

(d) Read about the **Fiedler vector** of a graph. Use the Diffusion maps coordinate that corresponds to the Fiedler vector to cluster the data into two clusters. In particular, the sign (+/-) of each point in this vector, defines its cluster.

(e) Plot the clustering result according to the Fiedler vector. (This is similar to the idea of **Spectral Clustering**).

Question 4 (20 points)

Random Projections

(a) Load the dataset named **Arcene data** and its associated labels **Arcene label** .

- This dataset is taken from here: https://archive.ics.uci.edu/ml/datasets/Arcene
- The task is to distinguish cancer versus normal patterns from mass-spectrometric data.
- The set you have has 100 data points, each point in dimension $D = 10,000$.

(b) Use the numpy function **randn** to create the random projection matrix **R**. Set the dimension of the reduced space **d** to be between 500 and 1000 (you can try different values for d).

(c) Normalize each row of **R** to have norm()=1.

(d) Project the data to the reduced space.

(e) Pick two points, for example point num. 22 and point num. 43 and compute the Euclidean distance between them in the original space and in the reduced space. Display (print) your results.

(f) Also compute the ratio between these two distances, and display it.

(g) SVM classification:

- Use a 10-fold cross validation to split the original data, the projected data and the label into a 90-10 split at each fold.
- Create two SVM models: one with the original (90%) train data and one with the projected (90%) train data.
- Use the two models to predict the class of the (10%) test data in the original space and in the reduced space.
- Save the correct classification rate for both models at each fold.
- Finally, display the mean correct classification rate for both models.
- Are the results of the two models the same? different? You can try to run your code with different values of d and see how it behaves.