Homework 6

Due May 21st 2021 by 11:59pm

Instructions: This homework consists of a reading assignment and two coding exercises. Please submit your solutions via Gradescope. Solutions should consist of three files: a PDF containing your solutions to the *non-coding questions*; a Jupyter notebook (`.ipynb` file) and an html print-out (`.html` file) with your solution to the *coding exercise*. **All coding exercises must be completed in Python.** Please be sure to comment the code appropriately. Students are encouraged to discuss homework problems, particularly on Canvas and in the TA hours, but must submit their own solutions.

Reading Assignment

- Review Lecture 6.
- Review Lab 6 (available from the Course Materials page on Canvas).

Exercise 1

In this exercise you will implement your own version of the *power iteration algorithm* to compute the top principal component of a feature matrix X.

Algorithm 1 Power Iteration

---

require matrix $\Sigma \in \mathbb{R}^{d \times d}$, max iteration number M.
initialization random vector $v_0 \in \mathbb{R}^d$.
repeat for $t = 0, 1, 2, \ldots M$

- Compute $z = Av_{t-1}$
- Update $v_t = \frac{z}{\|z\|_2}$

compute $\lambda = v_M^T \Sigma v_M$
return top eigenvector v_M and corresponding eigenvalue λ

(a) Download the data and separate into features X and target Y with the following code:

```
file = 'default_plus_chromatic_features_1059_tracks.txt'
dat = pd.read_csv(file, header=None)
```

```
X = dat.iloc[:,:-2]
y, orig = pd.factorize(dat.iloc[:,-1])
```

You can download the data from Canvas using this link.

(b) Perform an 80-20 train-test split. Standardize the train and test X. Construct the empirical covariance matrix from the train X via

$$V_X = \frac{1}{n-1} X^T X.$$

(c) Implement the power iteration algorithm. Run it on V_X with `max_iter`$= 100$.

(d) Compare the top eigenvalue found by your power iteration method to either scikit-learn's `PCA` or `np.linalg.eig`. Compare the corresponding eigenvectors by computing the norm of their difference. Note that eigenvectors are only identified up to the sign, so you may wish to compare instead to the negative of your eigenvector.

(e) Compute the *deflated* covariance matrix

$$V_{\text{deflate}} = V_X - \lambda_1 v_1 v_1^T,$$

where (λ_1, v_1) are the top eigenvalue and eigenvector found above. Run the power iteration algorithm on V_{deflate}. Compare the result to the *second* largest eigenvector and eigenvalue obtained by `sklearn` or `numpy`.

(f) Given the deflation method demonstrated above, propose an approach to compute the k^{th} largest eigenvector of X.

(g) Compute a reduced-dimensionality version of the training data by projecting it onto the top 20 principal components. You may use `sklearn` or `numpy` for this part, though you are also welcome to find the PCs by power iteration.

(h) Using the `KNearestNeighbor` class from `sklearn.neighbors`, fit *two* 1-nearest neighbor models: one on the original train data, and one on the projected train data.

(i) Generate class predictions and compute prediction accuracy on the test data (for the 1-NN model trained on the original data) and projected test data (for the model trained on projected data). How do the results compare?

(j) (Bonus) The power iteration algorithm iteratively updates the estimate of the top eigenvector v. Propose a stopping criterion for the algorithm more sophisticated than the max iteration limit implemented above.

Exercise 2

Repeat the above exercise on an Azure virtual machine. To show that your work was done on the Azure VM, please include the following: (1) a screenshot of your running VM in the Azure dashboard; (2) a screenshot of your command line after you `ssh` to the VM; (3) a screenshot of a running Python notebook on your VM, where you have run the following code in a cell block:

```
import os
os.getcwd()
```

The screenshots may either be embedded in your notebook or submitted as additional files.