The University of Texas at Austin
Department of Electrical and Computer Engineering

EE379K: Data Science Lab — Spring 2018

Lab Four

Caramanis/Dimakis Due: Monday, February 19, 3:00pm 2018.

Submit a Lab report pdf file that shows your name, your lab partner's name, your code, your results and a discussion of your results. Include your Kaggle names, the Team Kaggle name and a screenshot of your public submission score in the pdf of the lab report.

Also submit all your full code in separate files. For this lab, you can submit in either .ipynb format or .py format. If you choose to submit .py files, submit them in the format problemX.py or if you need, problemXa.py, problemXb.py, and so on.

Problem 1: PCA.

1. Generate 20 random points in $d = 3$, from a Gaussian multivariate distribution with mean $[0, 0, 0]$ and covariance matrix

$$\Sigma_1 = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.7 \end{bmatrix}.$$

   Let's call this data with label 1. Also generate 20 random points in $d = 3$ from another Gaussian with mean $[1, 1, 1]$ and covariance

$$\Sigma_2 = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}.$$

   Let's call that data with label 2. Create a three dimensional plot of the clouds of data points, labeled with the two labels.

2. What do the points look like ?

3. Concatenate all the points and ignore the labels for now. You have created an X matrix with 40 data points in $d = 3$ dimensions. Find the covariance matrix of this dataset. **Do not simply use np.cov, build the covariance matrix from the definition using linear algebra operations in python.**

4. Let's do PCA on this dataset using $k = 2$ dimensions: Find the two eigenvectors of the covariance matrix with the largest eigenvalues. Project the data points on these two vectors and show the two dimensional plot with the clouds of points. Also show the labels of the points. Did PCA make it easier to distinguish the two labels in two dimensions ? **Again, do not simply use sklearn PCA. You are only allowed to use matrix operations and np.linalg.eig to find eigenvalues and eigenvectors of a matrix.**

Problem 2: Low rank approximation of Mona Lisa.

1. Load the Mona Lisa image (in grayscale) and treat it as a matrix M. Perform a singular value decomposition on this matrix using **linalg.svd**. You can perform a low-rank approximation by zeroing out singular values and keeping only the top k. Show the best rank $k = 2$, $k = 5$ and $k = 10$ approximation to Mona Lisa.

2. If each pixel is represented by two bytes, how many bits is your compressed Mona Lisa for each of those k rank approximations?

Problem 3: Using Low Rank Structure for Corrupted Entries.

Download files `CorrMat1.csv` and `CorrMat3.csv` from Canvas. These are each 100 by 100 matrices. Look at the data and find which entries are corrupted. Then try to correct these corrupted entries. Explain your approach. (Hint: The corrupted entries have values that are completely out of the range of the others. This should help you identify which are the corrupted ones. For completing them, the hint is that we have been talking about PCA, low rank matrices and low-rank approximations.)

Problem 4: Getting more into Kaggle.

1. Goal: Get the best score you can in the Housing prices competition that you started in the previous lab. https://www.kaggle.com/c/house-prices-advanced-regression-techniques/

2. Train a ridge regression and a lasso regression model. Optimize the alphas using cross validation. What is the best score you can get from a single ridge regression model and from a single lasso model?

3. Plot the l_0 norm (number of nonzeros) of the coefficients that lasso produces as you vary alpha.

4. Add the outputs of your models as features and train a ridge regression on all the features plus the model outputs (This is called Ensembling and Stacking). Be careful not to overfit. What score can you get? (We will be discussing ensembling more, later in the class, but you can start playing with it now).

5. Install XGBoost (Gradient Boosting) and train a gradient boosting regression. What score can you get just from a single XGB? (you will need to optimize over its parameters). We will discuss boosting and gradient boosting in more detail later. XGB is a great friend to all good Kagglers!

6. Do your best to win. Try feature engineering and stacking many models. You are allowed to use any public tool in python. No non-python tools allowed.

7. Read (and post) in the Kaggle forums. Include in your report if you find something in the forums you like, or if you made a post or code, especially if other Kagglers used it afterwards.

8. Be sure you do not violate the rules of Kaggle! No sharing of code or data outside the Kaggle forums. Every student should have their own individual Kaggle account and teams can be formed in the Kaggle submissions with your Lab partner.

9. You will be graded based on your public score (include that in your report) and also on the creativity of your solution. In your report (**that you will submit as a pdf file**), explain what worked and what did not work. Many creative things will not work, but you will get partial credit for developing them. We will invite teams with interesting solutions to present them in class.