CS168, Spring 2019

Mini-Project #4

Due by 11:59 PM on Tuesday, April 30th.

Instructions

- You can work individually or with one partner. If you work in a pair, both partners will receive the same grade.
- If you've written code to solve a certain part of a problem, or if the part explicitly asks you to implement an algorithm, you must also include the code in your pdf submission. The code for all parts should go in an appendix. **This is a change from previous assignments**, where the code was embedded in each part.
- Make sure the plots that you submit are easy to read at a normal zoom level.
- Detailed submission instruction can be found on the course website (`https://web.stanford.edu/class/cs168`) under the "Coursework - Assignment" section. If you work in pairs, only one member should submit all of the relevant files.
- **Reminder:** No late assignments will be accepted, but we will drop your lowest mini-project grade when calculating your final grade.

Part 1: Principal Component Analysis (PCA)

Goal: In this part of the mini-project, you will run PCA on a real data set, and interpret the output.

Description: Download the `p4dataset2019.txt` file from the course Web site. The data represented there is from the 1000 genomes project. Each of the 995 lines in the file represents an individual. The first three columns represent respectively the individual's unique identifier, his/her sex (1=male, 2=female) and the population he or she belongs to[1]. The subsequent 10101 columns of each line are a subsample of nucleobases from the individual's genome.

We will be looking at the output of PCA on this dataset. PCA can refer to a number of related things, so to be explicit, in this section when we say "PCA" we mean

- The data should be centered (i.e., the sample mean subtracted out) but *not* normalized.
- The output should be the normalized principal components (i.e., unit-length eigenvectors).

Feel free to use a library implementation of PCA for the following questions. For python users, we recommend scikit learn's implementation. Matlab's built-in `pca` function can also be used. Note that with both python scikit and Matlab, you can specify how many principal components you want (this can save on computation time).

Exercises: First convert the data from the text file of nucleobases to a real-valued matrix (PCA needs a real-valued matrix). Specifically, convert the genetic data into a binary matrix X such that $X_{i,j} = 0$ if the i^{th} individual has column j's mode nucleobase[2] for his or her j^{th} nucleobase, and $X_{i,j} = 1$ otherwise. Note that all mutations appear as a 1, even if they are different mutations, so if the mode for column j is "G", then if individual i has an "A","T", or "C", then $X_{i,j}$ would be 1.

The first 3 columns of the data file provide meta-data, and should be ignored when creating the binary matrix X. We will examine genotypes to extract phenotype information.

[1] See `http://www.1000genomes.org/faq/which-populations-are-part-your-study/` for decodings

[2] By "mode nucleobase", we just mean the most frequently occurring nucleobase in that position (across the 995 data points).

(a) (Warm-up, do not submit.) Say we ran PCA on the binary matrix X above. What would be the dimension of the returned vectors?

(b) (6 points) We will examine the first 2 principal components of X. These components contain lots of information about our data set. Create a scatter plot with each of the 995 rows of X projected onto the first two principal components. In other words, the horizontal axis should be v_1, the vertical axis v_2, and each individual should be projected onto the subspace spanned by v_1 and v_2. Your plot must use a different color for each population and include a legend.

(c) (7 points) In two sentences, list 1 or 2 basic facts about the plot created in part (b). Can you interpret the first two principal components? What aspects of the data do the first two principal components capture? Hint: think about history and geography.

(d) (5 points) We will now examine the third principal component of X. Create another scatter plot with each individual projected onto the subspace spanned by the first and third principal components. After plotting, play with different labeling schemes (with labels derived from the meta-data) to explain the clusters that you see. Your plot must include a legend.

(e) (5 points) Something should have popped out at you in the plot above. In one sentence, what information does the third principal component capture?

(f) (4 points) In this part, you will inspect the third principal component. Plot the nucleobase index vs the absolute value of the third principal component. What do you notice? What's a possible explanation? Hint: think about chromosomes.

Deliverables: Scatter plot for part (b). Short discussion for part (c). Scatter plots for parts (d) and (f). One sentence answers for (e) and (f). Code for the whole section in the Appendix (which doesn't have to be separated into parts).

Bonus questions (up to 10 points): These questions suggest ways to explore the dataset in more detail. If you submit interesting answers, we will add a small amount of extra credit. Your work here could be aided with this mapping from each unique identifier to more data about the individual

(g) For this problem we simplified our dataset by capturing all deviations from the mode value with an indicator variable. This loses some information relative to the original data set. How would you create a real-valued matrix Y suitable for PCA analysis such that there is a bijection between our input data (minus the first three columns) and Y? The matrix Y should be a useful input to PCA. Explain the reasoning behind your choice of Y. Your answer to this question should not take more than a few sentences.

(h) Perform PCA on the matrix Y from part (g). Recreate the plot from part (b). What added value (if any) does this more complex representation add?

(i) Can you uncover what information is captured in the fourth principal component of X?

(j) The provided dataset for part 1 represents approximately 0.6% of the dataset found at the top of this directory: `http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/supporting/hd_genotype_chip/`. What other information can you uncover from this much larger file? For this question we encourage you to look at populations beyond Africa. You will notice that the file is a compressed .vcf file, you may want to look into installing `vcftools` to work with the file.

Part 2: Understanding the difference between PCA and Least Squares

Goal: Both PCA and least squares regression can be viewed as algorithms for inferring (linear) relationships among data variables. In this part of the mini-project, you will develop some intuition for the differences

between these two approaches, and an understanding of the settings that are better suited to using PCA or better suited to using the least squares fit.

Description: The high level bit is that PCA is useful when there is a set of *latent* (hidden/underlying) variables, and all the coordinates of your data are linear combinations (plus noise) of those variables. The least squares fit is useful when you have direct access to the independent variables, so any noisy coordinates are linear combinations (plus noise) of known variables.

We will consider a simple example with two variables, x and y, where the true relationship between the variables is $y = 2x$. Our goal is to recover this relationship—namely, recover the coefficient "2". In subpart (b), we consider the setting where our data consists of the actual values of x, and noisy estimates of y. In subpart (c), we consider the case where our data consists of noisy measurements of both x and y. For each part, we will evaluate the quality of the relationship recovered by PCA, and that recovered by standard least squares regression.

As a reminder, least squares regression minimizes the squared error of the dependent variable from its prediction. Namely, given (x_i, y_i) pairs, least squares returns the line $l(x)$ that minimizes $\sum_i (y_i - l(x_i))^2$.

Exercises:

(a) Warm-up (do not submit):

- Write a routine `pca-recover` that takes a vector X of x_i's and a vector Y of y_i's and returns the slope of the first component of the PCA (namely, the second coordinate divided by the first).
- Write a routine `ls-recover` that takes X and Y and returns the slope of the least squares fit. (Hint: since X is one dimensional, this takes a particularly simple form[3]: $\langle X - \overline{X}, Y - \overline{Y} \rangle / \|X - \overline{X}\|_2^2$, where $\overline{X}$ is the mean value of X.)
- Set $X = [.001, .002, .003, \ldots, 1]$ and $Y = 2X$. Make sure both routines return 2.

(b) (4 points) Say the elements of X and Y were chosen identically and independently at random (e.g. every element is uniformly distributed in the square $[0, 1] \times [0, 1]$). What would PCA recover, and what would LS recover?

(c) (5 points) We first consider the case where x is an independent (a.k.a. explanatory) variable, and we get noisy measurements of y. Fix $X = [x_1, x_2, \ldots, x_{1000}] = [.001, .002, .003, \ldots, 1]$. For a given noise level c, let $\hat{y}_i \sim 2x_i + \mathcal{N}(0, c) = 2i/1000 + \mathcal{N}(0, c)$, and $\widehat{Y} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{1000}]$. Make a scatter plot with c on the horizontal axis, and the output of `pca-recover` and `ls-recover` on the vertical axis. For each c in $[0, 0.05, 0.1, \ldots, .45, .5]$, take a sample $\widehat{Y}$, plot the output of `pca-recover` as a red dot, and the output of `ls-recover` as a blue dot. Repeat 30 times. You should end up with a plot of 660 dots, in 11 columns of 60, half red and half blue.

Hint: in both numpy and matlab, `randn(1000)*`σ generates an array of 1000 independent samples from $\mathcal{N}(0, \sigma^2)$. In python you'll also need to add `from numpy.random import randn`.

(d) (5 points) We now examine the case where our data consists of noisy estimates of both x and y. For a given noise level c, let $\hat{x}_i \sim x_i + \mathcal{N}(0, c) = i/1000 + \mathcal{N}(0, c)$ and $\hat{y}_i \sim y_i + \mathcal{N}(0, c) = 2i/1000 + \mathcal{N}(0, c)$. Similar to (b), for each c in $[0, 0.05, 0.1, \ldots, .45, .5]$, take a sample $\widehat{X}$ and $\widehat{Y}$, plot the output of `pca-recover` as a red dot, and the output of `ls-recover` as a blue dot. Repeat 30 times. You should have a plot with 330 red dots and 330 blue dots.

(e) (9 points) Why does PCA do better in one, and least squares in the other? (No need to repeat the discussion above about latent vs. known independent variables.) Split this into three questions: (i) Why does PCA do poorly with noise in only Y? (ii) Why does PCA do well with noise in X and Y? (iii) Why does LS do poorly with noise in X and Y?

Deliverables: Short answer for (b). One plot for (c), one plot for (d), and discussion for (e). Code for (c) and (d) in the Appendix (which can be combined with the code from Part 1).

[3] Recall that angle brackets just mean inner product of two vectors, i.e. if a, b are two $d \times 1$ vectors, then $\langle a, b \rangle = \sum_{i=1}^{d} a_i b_i$.