CS168, Spring 2015

# Mini-Project #4

Due by 11:59 PM on Tuesday, April 28th.

## 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 plots you submit are easy to read at a normal zoom level.
- Detailed submission instruction can be found on the course website (`http://cs168.stanford.edu`) 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: Principle 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:**

Take a look at `http://web.stanford.edu/class/cs168/p4_dataset.csv`. The file lists 368 car models from 2004, along with the car's category (`Sedan, Sports, SUV, Wagon, Minivan`) and 11 real valued features (`Retail, Dealer, Engine, Cylinders, Horsepower, CityMPG, HighwayMPG, Weight, Wheelbase, Length` and `Width`).

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 normalized principle components (unit eigenvectors) rather than their associated eigenvalues, and
- We will be scaling the data by dividing by the standard deviation in each coordinate (after subtracting the mean).

A simple python implementation follows. Feel free to use it directly.

```
import numpy as np
import numpy.linalg as la

# data[0] has the real valued features of the first car, etc
def PCA(data, scale = True):
    data -= np.mean(data, 0)
    if scale:
        data /= np.std(data, 0)
    C = np.cov(data, rowvar = 0, bias = 1) # covariance matrix
    w,V = la.eigh(C) # w = eigenvalues, V[:,w] = corresponding normalized eigenvectors
    # return the eigenvectors ordered by w (in decreasing order)
    return [V[:,i] for i,e in sorted(enumerate(w), key = lambda x: x[1], reverse = True)]
```

Note that the `pca` function in matlab does not scale or normalize the data by default. You can mimic the behavior above in matlab with `inv(diag(std(data))) * pca(data, 'VariableWeights', 'variance')`.

**Exercises:**

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

(b) Run PCA on the car dataset. Find the first two components $v_0$ and $v_1$. As a sanity check, the first component $v_0$ should have two positive coordinates, and the absolute values of all its coordinates should be roughly equal. If you're getting two negative coordinates, use $-v_0$ instead (both are equally valid as normalized eigenvectors, this is just to fix an orientation for the rest of the section). The second component $v_1$ should have four positive coordinates.

(c) What do the two positive coordinates of $v_0$ correspond to? List 1 or 2 basic facts about cars that are suggested by the first component. What might we expect cars with {high, low} projections onto $v_0$ to look like?

(d) The second component also tells us a lot, but is easier to see graphically. Draw a scatter plot with the Sports cars, SUVs, and Minivans projected onto the first two components of the PCA, using three different colors (you can leave out the Sedans and Wagons). In other words, the horizontal axis should be $v_0$, the vertical axis $v_1$, and each car should be projected onto the subspace spanned by $v_0$ and $v_1$.

(e) Something should have popped out at you in the plot above. Explain why the Sports cars, SUVs, and Minivans are clustered where they are.

**Deliverables:** Two vectors for part (b). One plot for part (d). Discussion for (c) and (e). Code for the whole section in the Appendix (doesn't have to be separated into parts).

**Bonus questions:** These are largely to help your understanding, but if you submit answers, we will add a small amount of extra credit.

(f) How would $v_0$ change if we

- Duplicated the `Retail` column?
- Doubled every value in the `Retail` column?
- Did not scale the data (i.e. `scale = False` in the reference implementation)?

(g) Currently the first eigenvalue is 7.2, the second is 1.8, and the rest of the eigenvalues sum to 2 (the sum of all the eigenvalues is 11 ... why?). How would you interpret the data set if instead

- The first eigenvalue were 10 and the rest were 0.1.
- All the eigenvalues were roughly 1.

# 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 variabls, 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.

We looked at PCA in Part 1. Note that in this part, we will be using the *unscaled* version of PCA (why?). More concretely, if you are using the python `PCA` implementation from Part 1, your code should now call it with `scale = False`. If you are using the matlab `pca`, you can just use `pca(data)` directly.

As a reminder, least squares regression tries to minimize 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: $\langle X - \overline{X}, Y - \overline{Y} \rangle / \|X - \overline{X}\|_2^2$, 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.
- Say the elements of $X$ and $Y$ were chosen completely at random (e.g. every element is independently and uniformly distributed in $[0, 1]$). What would `pca-recover` return, and what would `ls-recover` return?

(b) 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 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)*`$\alpha$ generates an array of 1000 independent samples from $\mathcal{N}(0, \alpha^2)$. In python you'll also need to add `from numpy.random import randn`.

(c) 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.

(d) For (b) and (c), evaluate the quality of the relationship recovered by PCA, and that recovered by least squares. Why does PCA do better in one, and least squares in the other?

**Deliverables:** One plot for (b), one plot for (c), and discussion for (d). Code for (b) and (c) in the Appendix (can be mingled with code from Part 1).