IE 529
Stats of Big Data and Clustering
Computational Assignment: 1
DUE: Monday, October 24, 2016

I. Regression analysis:

1. The data is given to you in *Comp1_IE529* contains two vectors, 'lift_kg' and 'putt_m', where 'lift_kg(i)' corresponds to a maximum weight lifted by athlete i in kilograms, and 'putt_m(i)' corresponds to a longest shot-put by athlete i in meters. Your assignment is to use regression methods to determine a model that describes the relationship between the two variables. That is, suppose x_1 = 'lift' and x_2 = 'putt'; you should find a mathematical model relating x_1 and x_2, such as
$$x_2 = 10 + 2x_1 + x_1^2 - 0.1x_1^3.$$
The relationship may be linear/affine, polynomial or logistic.
2. Write a simple program to compute a least squares solution for the case of linear or polynomial regression, and determine the lowest order model that fits the data reasonably well (order 1 is linear, and higher orders are polynomial).
3. Call an existing logistic regression function in Matlab or Python to determine if a logistic model will fit the data much better or not.
4. State which of your candidate models best describes the data, taking into account that simplicity is preferred. **Provide plots** of a linear fit, one polynomial fit (i.e., second order or higher) and one logistic fit. Compute the sum-of-square of residuals, i.e., give the actual cost $\|\epsilon_i\|_2^2$, for each of the models plotted.
5. Turn in a report including (a) clearly labeled plots with associated explicit models and costs, (b) a brief discussion explaining your model choice, and (c) your code/function calls.

See the plot in Figure 1 to note that a lower order polynomial, i.e. a first, second, or third order polynomial seems like it will be sufficient to fit the data.

(a) Consistent with the assessment from Figure 1. I tried out a first, second, or third order polynomial. I calculated the error between the data and the predicted data from the learned models by using the 2-norm of the difference. These were the results

$$\|\hat{\epsilon}_{lin}\|^2 = 94.58, \qquad \|\hat{\epsilon}_{quad}\|^2 = 51.09, \qquad \|\hat{\epsilon}_{cub}\|^2 = 44.58.$$

See the plot of the different fits on top of the data in Figure 2.

(b) While clearly the cubic function has the lowest error, it seems to overfit the data. For example if we took a 62nd order polynomial (there are 62 data points) we would have an error of zero but the function would be outrageous, not following the trend of the data at all.

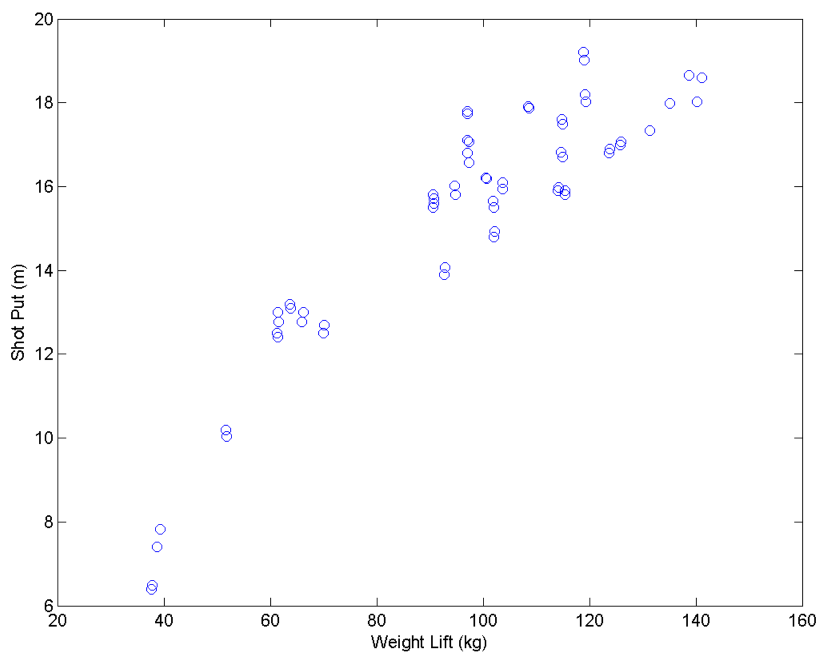

Figure 1: Shot Put Data

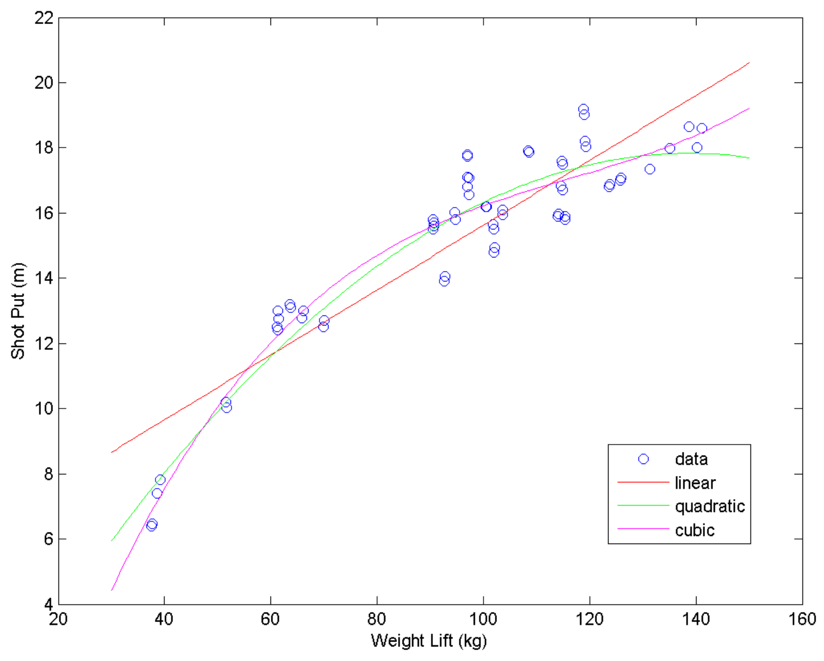

Figure 2: Regression Fits on the Shot Put Data

(c)

```
% comp1_1.m
% Philip Pare
% 10/17/16
% I use pinv(A) but this is equivalent to inv(A'*A)*A' or (A'*A)\A'

load('Comp1_IE529_data1.mat')

% 1. Pick Model
figure(1)
clf;
plot(lift_kg,putt_m,'o')
hold on
xlabel('Weight Lift (kg)')
ylabel('Shot Put (m)')
saveas(figure(1),'data_1.png')

% 2. Learn (Estimate) Model Parameters

% Linear
A = [ ones(length(lift_kg),1), lift_kg'];
mb = pinv(A)*putt_m';
error_lin = norm(A*mb-putt_m')^2;

lift_hat = 30:150;
put_hat = [ones(length(lift_hat),1) lift_hat']*mb;
plot(lift_hat,put_hat,'r')

% Quadratic
A2 = [ ones(length(lift_kg),1), lift_kg' (lift_kg.^2)'];
mb2 = pinv(A2)*putt_m';
error_2 = norm(A2*mb2-putt_m')^2;

put_hat2 = [ones(length(lift_hat),1) lift_hat' (lift_hat.^2)']*mb2;
plot(lift_hat,put_hat2,'g')

% Cubic
A3 = [ ones(length(lift_kg),1), lift_kg' (lift_kg.^2)' (lift_kg.^3)'];
mb3 = pinv(A3)*putt_m';
error_3 = norm(A3*mb3-putt_m')^2;

put_hat3 = [ones(length(lift_hat),1) lift_hat' (lift_hat.^2)' (lift_hat.^3)']*mb3
plot(lift_hat,put_hat3,'m')
legend('data','linear','quadratic', 'cubic', 'Location', 'SouthEast')
saveas(figure(1),'fits.png')
```

II. Principal Component Analysis: real data

1. The data for this portion of the computational assignment consists of 4 vectors of data, each with 150 entries (NOTE: each row is a sample with 4 entries; the number of rows is the number of samples). Each column represents a measurement, in centimeters, of one specific feature, taken from a sample of 150 flowers. The four features are sepal length and width, and petal length and width. We would like to distill this data down to a lower dimension (2 or 3), and try to determine how many species might be represented by the data.
2. For the given data, de-mean each entry, using column sample means. Perform a PCA on the de-meaned data. Clearly explain how you performed the PCA (submit pseudo-code for your own PCA code, or reference and describe any function you called, i.e, from what library, how it works, what it takes as inputs and produces as outputs).
3. State what portion of the variance in the data is contained in each of the 4 principal components. From these values, state how many *true* components are needed to represent the data.
4. On a 2D graph with the horizontal axis given by the first PC, and the vertical plot given by the second PC, plot the 2D representation of all the data points (i.e., find the 2D projection of each row). Discuss: (a.) are there any visually apparent clusters, if so, how many, and (b.) from this do you think you can conjecture anything about the number of species represented by the data?
5. Repeat parts 2.-4. for *standardized* data: in addition to de-meaning the entries by column means, you should also scale by the inverse of the sample standard deviation, i.e., for each element x_{ij} in the original matrix X, normalize the element to $\tilde{x}_{ij}$ where

$$\tilde{x}_{ij} = \frac{x_{ij} - \bar{x}_j}{s_j}.$$

   Note whether or not this scaling changes the outcome of your analysis.
6. Turn in a report including (a) a matrix with the 4 components for the data and their associated variances, and (b) plots for part 4. for both the de-meaned and the standardized data sets.

(a) The principal components are the columns of the matrix:

$$\begin{bmatrix} 0.3616 & 0.6565 & 0.5810 & -0.3173 \\ -0.0823 & 0.7297 & -0.5964 & 0.3241 \\ 0.8566 & -0.1758 & -0.0725 & 0.4797 \\ 0.3588 & -0.0747 & -0.5491 & -0.7511 \end{bmatrix},$$

with variances

$$\begin{bmatrix} 4.2248 & 0.2422 & 0.0785 & 0.0237 \end{bmatrix}.$$

(b) See Figure 3 for the 2D representation of all the demeaned data points. Note there are two distinct clusters. This would suggest there are two species.

See Figure 4 for the 2D representation of all the demeaned, normalized data points. Note there are still two clusters. This still suggests there are two species.

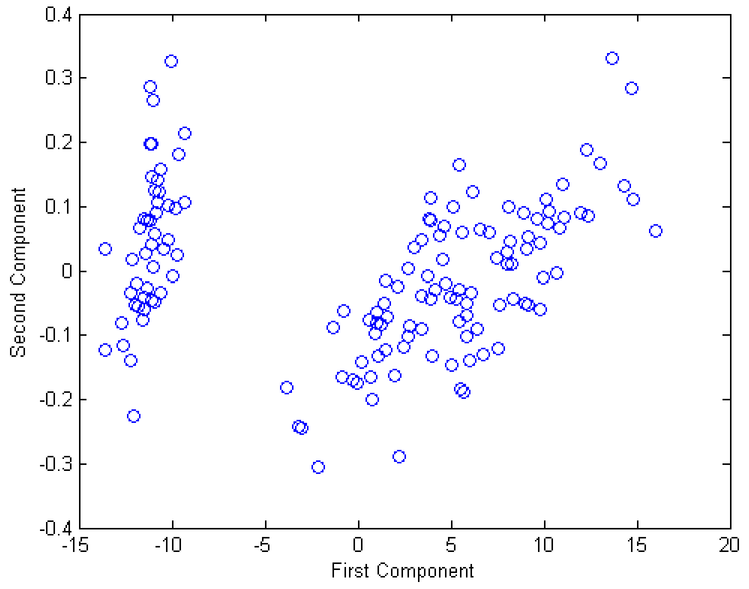

Figure 3: Demeaned data plotted against the two principal components

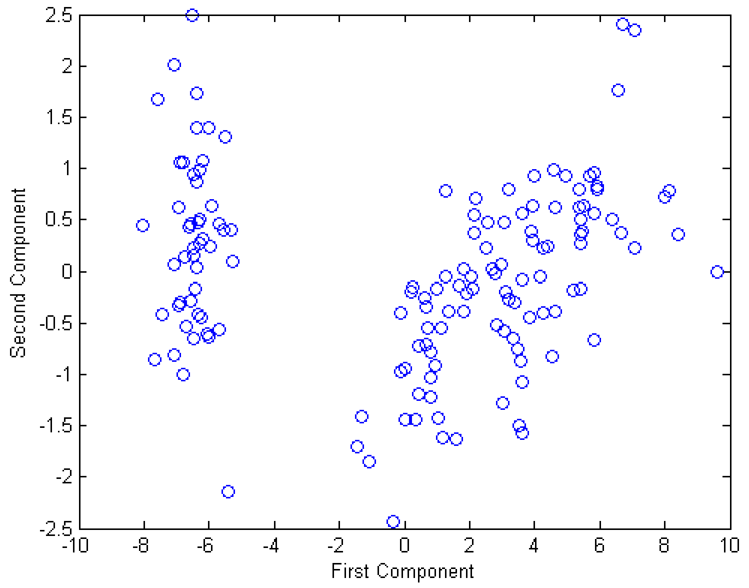

Figure 4: Demeaned, normalized data plotted against the two principal components

```
% Comp1_2.m

[n, m] = size(flowers);

% 1. Compute and subtract mean
meanflowers = mean(flowers,1);
flowersmean0 = flowers - repmat(meanflowers,n,1);

% 2. Compute the covariance matrix
Cova = flowersmean0'*flowersmean0/(n-1);

% 3. Compute an eigenvalue decomposition and sort both sort both the
%    eigenvalues and associated eigenvectors in descending order
[U, D] = eig(Cova);
[eigvals,Ind] = sort(diag(D),'descend');
Us = U(:,Ind);

% 4. % (i) Plot of the reduced, un-biased, and scaled data
data2d=D*U'*flowersmean0';
data2d = data2d(Ind,:);
data2d(3,:) = [];

figure(2)
clf
plot(data2d(1,:),data2d(2,:),'o')
hold on
xlabel('First Component')
ylabel('Second Component')

% 5. Repeat for Normalized Data

flowersNormalized = flowersmean0/diag(diag(Cova).^(.5));

% 2. Compute the covariance matrix
CovaNorm = flowersNormalized'*flowersNormalized/(n-1);

% 3. Compute an eigenvalue decomposition and sort both sort both the
%    eigenvalues and associated eigenvectors in descending order
[Unorm, Dnorm] = eig(CovaNorm);
[eigvalsNorm,Indnorm] = sort(diag(Dnorm),'descend');
UsNorm = Unorm(:,Indnorm);

% (i) Plot of the reduced, un-biased, and scaled data
dataNorm2d=Dnorm*Unorm'*flowersNormalized';
dataNorm2d = dataNorm2d(Indnorm,:);
```

```
dataNorm2d(3,:) = [];

figure(3)
clf
plot(dataNorm2d(1,:),dataNorm2d(2,:),'o')
hold on
xlabel('First Component')
ylabel('Second Component')
```