Homework 5

Wednesday, October 14, 2020 8:18 PM

Homework 5

Chapter 15

Homework 5: Data, Correlation, and Smile Detection

Contents

Learning Objectives

Concepts

- Describe the physical significance of the mean and standard deviation of a data set.
- Describe the physical significance of correlation, anti-correlation or non-correlation of two variables.
- Approximate the mean and standard deviation from a histogram of the data.
- Interpret the meaning of a pair of images that has a Pearson Correlation Coefficient of: about 0; or 0.5; or 0.9.
- Interpret the physical/mathematical meaning of the diagonal and off-diagonal elements in a 2 x 2 correlation matrix, $\mathbf{C} = \mathbf{A}^T\mathbf{A}$, if given the equation for the Pearson Correlation Coefficient.

MATLAB skills

- Compute the dot product of two vectors
- Set up the appropriate matrices to compute the correlation coefficient between two variables.

15.1 Correlation

Now let's consider that we have N measurements of two different associated quantities and want to test whether these are linearly correlated (if one goes up, the other also goes up), anti-correlated (if one goes up, the other goes down) or uncorrelated (the behavior of one cannot be predicted by watching the behavior of the other). (Please note that correlation has nothing to do with causality!). There are many different measures of correlation, but we will discuss here one of the most common, the Pearson Correlation Coefficient.

For a pair of associated datasets $X = \{x_i\}$ and $Y = \{y_i\}$, each with N elements, we define the Pearson Correlation Coefficient to be:

$$\rho(X,Y) = \frac{1}{N-1}\sum_{i=1}^{N}\frac{(x_i-\mu_x)}{\sigma_x}\frac{(y_i-\mu_y)}{\sigma_y} \tag{15.1}$$

where μ_x, μ_y, σ_x and σ_y are the means and standard deviations of the datasets. Essentially, for each pair of values, we take the product of the variations from the mean, then sum these products up over all pairs of values and normalize by the expected variation as characterized by the standard deviation. If the two values are consistently always on the same side of the mean, then each term in the sum will contribute positively, and the total value will be close to $+1$, indicating positive correlation. If the two values are consistently on the opposite sides of the mean, then each term in the sum will contribute negatives, and the total value will be close to -1, indicating anticorrelation. If, for every pair, it is just as likely that the two values will be on opposite sides of the mean as on the same side of the mean, then the sum will go to 0, and the two values are uncorrelated.

Consider the following data:

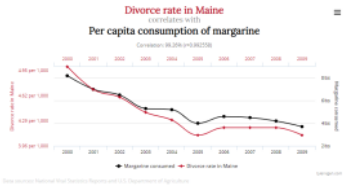
	A	B	C	D	E	F	G	H	I	J
3		Poverty	Infant Mort	White	Crime	Doctors	Traf Deaths	University	Unemployed	Income
4	Alabama	15.7	9.0	71.0	448	218.2	1.81	22.0	5.0	42,666
5	Alaska	8.4	6.9	70.6	661	228.5	1.63	27.3	6.7	68,460
6	Arizona	14.7	6.4	86.5	483	209.7	1.69	25.1	5.5	50,958
7	Arkansas	17.3	8.5	80.8	529	203.4	1.96	18.8	5.1	38,815
8	California	13.3	5.0	76.6	523	268.7	1.21	29.6	5.8	61,021
9	Colorado	11.4	5.7	89.7	348	259.7	1.14	35.6	4.9	56,993
10	Connecticut	9.3	6.2	84.3	256	376.4	0.86	35.6	5.7	68,595
11	Delaware	10.0	8.3	74.3	689	250.9	1.23	27.5	4.8	57,989
12	Florida	13.2	7.3	79.8	723	247.9	1.56	25.8	6.2	47,778
13	Georgia	[illegible]	8.1	65.4	493	217.4	1.46	27.5	6.2	50,861
14	Hawaii	[illegible]	5.6	[illegible]	273	317.0	1.33	29.1	3.9	67,214
15	Idaho	12.5	6.[illegible]	[illegible]	239	168.8	1.60	24.0	4.9	47,576

Exercise 15.1

1. Look over the data. By eye, which columns look correlated? Anticorrelated? Uncorrelated?
2. Choose your two favorite columns of data from this dataset. Input these into vectors in Matlab. For each of these vectors, subtract off the mean, and then divide out the standard deviation.
3. With these vectors, how would you directly compute the correlation coefficient between them? Go ahead and do this in MATLAB, and reflect on your result. Don't forget to normalize by $1/(N-1)$.

Exercise 15.2

A note of warning. Correlation does not imply causation! To drive this point home, visit the Spurious Correlation Website. Follow the link at the bottom of the site to discover and plot a spurious correlation of your very own.



This just in: Science **proves** margarine causes divorces for Mainers!

Correlation: The Idea, the Matrices, and the MATLAB

Correlation: The Idea, the Matrices, and the MATLAB

Now we are going to use matrix mathematics to construct correlation coefficients in an efficient manner. Let's first consider a data matrix $\mathbf{B}$ which has two columns of data, each of which has N samples:

$$\mathbf{B} = \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ \dots & \dots \\ x_N & y_N \end{pmatrix}$$

If we subtract out the means and divide out the standard deviation and a factor of $\sqrt{N-1}$, we get the matrix $\mathbf{A}$:

$$\mathbf{A} = \frac{1}{\sqrt{N-1}} \begin{pmatrix} \frac{x_1-\mu_x}{\sigma_x} & \frac{y_1-\mu_y}{\sigma_y} \\ \frac{x_2-\mu_x}{\sigma_x} & \frac{y_2-\mu_y}{\sigma_y} \\ \frac{x_3-\mu_x}{\sigma_x} & \frac{y_3-\mu_y}{\sigma_y} \\ \dots & \dots \\ \frac{x_N-\mu_x}{\sigma_x} & \frac{y_N-\mu_y}{\sigma_y} \end{pmatrix}$$

where μ_x, μ_y and σ_x, σ_y are the mean and standard deviations of each column, and N is the number of samples (rows).

The *correlation matrix* $\mathbf{C} = \mathbf{A}^T\mathbf{A}$ has elements of the self and cross correlations between the datasets.

Exercise 15.3

1. Before we start to use this idea, let's think it through a bit...
   (a) What is the size of the matrix $\mathbf{C}$? 2x2 ✓
   (b) What do the elements on the *diagonal* of this matrix represent? What must their values be? Not sure? Maybe 1 ✓
   (c) What do the elements of the off-diagonal represent? What is element $\mathbf{C}_{12}$ of this matrix? What is element $\mathbf{C}_{21}$? What do you notice? Is this always going to be true? What about if we had three datasets? What can you say about the elements $\mathbf{C}_{13}$ vs $\mathbf{C}_{31}$?
   (d) If you create a data matrix that has completely identical columns of data, what should the correlation matrix look like? All 1s
   (e) If you create a data matrix that has completely uncorrelated datasets, what should the correlation matrix look like? [10] Identity

Self-correlations. Not 100% sure, but it smells like the diagonals are 1

C12 = Cov(x,y)
C13 is correl w/ x&z?

Exercise 15.4

In this exercise we want you to calculate the correlation matrix in MATLAB using the 2 favorite data vectors you chose earlier.

1. In MATLAB, construct the correlation matrix $\mathbf{C}$ using the method described above.
2. Check your results by using the MATLAB function `corrcoef`. Please note that the input to

this function is the original data vectors.

15.2 Correlation in Facial Recognition

Kinds of Correlation in an Image Set

If we think about photos now, we can think about two different correlations: the correlation between a given pair of pixels (across all the pictures in a data set), and the correlation between photos (across all the pixels in those images). In order to compute an accurate correlation coefficient, you need to have multiple data points in each set being correlated, e.g., many pixels in each picture being correlated, or many pictures across which a pair of pixels (pixel locations) can be correlated.

Think about what each of these correlations *means*. What would a high correlation between a given pair of images mean? What about a high correlation between a given pair of pixels (e.g., the upper-left-most pixel and the upper-right-most pixel)? It might help to open a few face images or draw some face sketches to think about.

Exercise 15.5

Consider six grayscale pictures, each with a resolution of m x n pixels.

1. What is the size of the data matrix containing these six pictures as the columns? mn x 6
2. What is the expression for the correlation matrix between the pictures? What size is this correlation matrix? corrcoef(images)
3. What is the expression for the correlation matrix between different pixels? Pay careful attention to the mean and standard deviation you are using. What is the size of this correlation matrix? corrcoef(transpose(images))
4. People's faces are approximately left-right symmetric. How would you expect this to affect the entries in the correlation matrix between different pixels? I think the edges would be close to 1

Test your understanding...

- Pull in **six images** from the class data matrix from last year's QEA (current second years). The data should be in the (`test_images` variable in `face_bases.mat` file). Take the six images and put them in a variable called `faces`. Each of these images should come from different people - there are 8 images per person stored in the data matrix.
- Use the `resample` command to bring them down to a smaller resolution (e.g., 25 x 25) using
  `dfaces = imresize(faces,[25 25]);`
  his should be a 25 x 25 x 6 matrix.
- Now reshape them appropriately to create a matrix in which each column is a (reshaped) face using
  `rdfaces = reshape(dfaces,size(dfaces,1) * size(dfaces,2),size(dfaces,3));`

3 & 2, 4 & 5, 5 & 6

Exercise 15.6

1. Find the correlation between six different images. Which images have the highest correlation?
2. Now find the correlation between pixels across images. Try taking a single column of this matrix and reshape that column into an image. What does that image tell you? You may want to repeat this reshape and visualization for columns 1, 25, 400, and 625 to get a feel for what is happening.

Darker pixels are the pixels that are more common.

15.3 Smile Detection—Concepts

In this section we are going to use our toolbox of linear algebra skills to "detect" whether or not a person is smiling in a photograph. The approach that we will take is very common in **machine learning** - we will use a dataset to **train** our algorithm, and we will use a different dataset to **test** our algorithm. We will first develop the conceptual framework and then implement the approach in MATLAB.

The Big Idea

Let's assume that we have 100 training photos of faces, each consisting of a 5 by 5 grid of pixels. Let's pack these into a matrix $\mathbf{A}$ with 100 rows and 25 columns, i.e. every row is a different face and every column is a different pixel.

Let's also assume that we have already classified every training face as "smiling" or "not-smiling". Let's create a column vector $\mathbf{b}$ with 100 rows (corresponding to each face) which has either 1 (smiling) or 0 (not-smiling).

Let's now develop a linear system of algebraic equations by trying to express the vector $\mathbf{b}$ as a linear combination of the columns of $\mathbf{A}$, i.e.

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

Notice that the vector $\mathbf{x}$ is a column vector with 25 rows - one row for each pixel. Since there are more rows than columns we know that an **exact** solution does not exist, so we will find the **approximate** solution by orthogonal projection, i.e. we will solve

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}$$

for the unknown vector $\mathbf{x}$, which on paper takes the form

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$$

Now that we have the vector $\mathbf{x}$, let's use it to detect whether a test image is smiling. Assuming that the test image is packed into a single row vector $\mathbf{t}$ (with 25 columns) then the product

$$\mathbf{t}\mathbf{x}$$

will return a scalar. If this scalar is close to "1" then we predict the face is smiling. If this scalar is close to "0" then we predict the face is not smiling.

Exercise 15.7

In this exercise you will be carefully reading and interpreting this big idea. We are including these

questions as a scaffold, pointing out interesting features along the way.

1. Read "The Big Idea" again!
2. Interpret what it means to write down the linear system of equations $\mathbf{Ax} = \mathbf{b}$ and give a meaning to the vector $\mathbf{x}$.

X is the weighting of the pixels of A that best predicts what b should be.

3. Interpret the product $\mathbf{A}^T\mathbf{A}$ and the product $\mathbf{AA}^T$.

No idea--checking answer: I accept that the things listed are true, but don't understand the implications.

4. The vector $\mathbf{x}$ does not satisfy $\mathbf{Ax} = \mathbf{b}$ exactly. What does the expression $\mathbf{Ax} - \mathbf{b}$ tell you?

The difference

5. How would you decide whether your "trained" algorithm was worth using on a test dataset?

Look at Ax - b to see if the error is low

6. Assume you had 40 test images with 25 pixels each and that you pack them into a matrix $\mathbf{T}$ with 40 rows and 25 columns. Write down the matrix-vector product you would use for smile detection on this test dataset.

Tx

7. How would you measure the accuracy of your predictions if we also provided you with the data on whether each test image was smiling or not?

Mean(Tx - CorrectAnswers)

15.4 Smile Detection—Implementation

Please download the file `smiles.mat` from the canvas site. If you load this file in MATLAB, you will then have access to the following variables in your workspace.

```
train_data       - a 3D array containing 19685 24 x 24 pixel
                   images of faces
smile_flag_train - a vector of the same length as the number
                   of images in train_data, with 1s indicating
                   which images are smiling
test_data        - 500 24 x 24 pixel images of faces
smile_flag_test  - a vector of the same length as the number
                   of images in test_data, with 1s indicating
                   which images are smiling
```

The 'train data' and the associated 'smile flag train' are the sets of data you should use to develop your mathematical model. The 'test data' and its associated 'smile flag test' are the sets of data you should use to test your algorithm when you are finished!

Exercise 15.8

Now you are going to implement a smile detector in MATLAB. You should consider following the procedure below to implement the smile detector.

1. Sketch out a set of steps you would take in order to implement smile detection. (Just words here - no code. e.g. we will have to pack all images into a single matrix)
2. Turn this set of steps into MATLAB pseudo-code. Identify important coding elements without implementing, e.g. we will use **reshape** to pack the given dataset into a matrix.
3. Review the documentation for MATLAB functions that will be used and be clear on how to use them before implementation, e.g. » help reshape

4. Methodically implement smile detection in MATLAB, testing as you go.

Alternatively, you can use our walkthrough notebook. The notebook has embedded solutions or you can try it with minimal scaffolding using the suggested process above. Even if you decide not to use the walkthrough notebook, it's worth running the embedded solutions to pickup some techniques for visualizing your smile detector model.

15.5 Conceptual Quiz

Please see Canvas for the questions.

Solution 15.1

1. We are interested in the relationship between poverty and infant mortality. Generally speaking it looks like high values of one correspond to high values of the other, and vice versa, so they would seem to be correlated.

2. We are going to define these as column vectors in MATLAB as follows (just using first 6 observations for simplicity)

```
>> X = [15.7;8.4;14.7;17.3;13.3;11.4];
>> Y = [9.0;6.9;6.4;8.5;5.0;5.7];
```

   Now we need to normalize each one - it is easier to use built-in MATLAB functions to find the mean and standard deviation.

```
Xn = (X - mean(X))./std(X);
Yn = (Y - mean(Y))./std(Y);
```

3. To find the correlation coefficient we need to multiply every entry in Xn and Yn together and then add them up. That sounds like a matrix operation that can be implemented as follows

```
>> coeff = transpose(Xn)*Yn./5
```

   where we have normalized by N-1. The result is 0.5191, which indicates a substantial correlation between poverty and infant mortality.

Solution 15.3

1. (a) Since $\mathbf{A}$ has size $N \times 2$, we know $\mathbf{A}^T$ has size $2 \times N$. Then $\mathbf{C} = \mathbf{A}^T\mathbf{A}$ has size 2×2.
   (b) The elements on the diagonal represent the self-correlation of each data column. Each element of the diagonal will be 1.
   (c) The elements $\mathbf{C}_{12} = \mathbf{C}_{21}$ is the correlation between the two columns of data. Regardless of size, the correlation matrix will be symmetric.
   (d) If the data matrix had identical columns of data, the correction matrix would be all 1s.
   (e) If the data is uncorrelated, the off-diagonal entries in the correlation matrix will be all 0s, so it will be the identity matrix. (Note: real data is unlikely to have 0 correlation, just by accident, so it will just have numbers that are close to 0.)

Solution 15.4

1. There are a few ways to do this. Here is one of them, where we use "mean" and "std" on the original data vectors

```
X = [15.7;8.4;14.7;17.3;13.3;11.4];
Y = [9.0;6.9;6.4;8.5;5.0;5.7];
Xn = (X - mean(X))./std(X);
Yn = (Y - mean(Y))./std(Y);
A = [Xn Yn]./sqrt(5);
C = transpose(A)*A

C =
    1.0000    0.5190
    0.5190    1.0000
```

An alternative is to use "mean" and "std" on a data matrix. This works because both of these functions will return a row vector containing the mean and standard deviation alone each column.

```
X = [15.7;8.4;14.7;17.3;13.3;11.4];
Y = [9.0;6.9;6.4;8.5;5.0;5.7];
B = [X Y];
A = (B-mean(B))./std(B)./sqrt(5);
C = transpose(A)*A

C =
    1.0000    0.5190
    0.5190    1.0000
```

2. The function `corrcoef` accepts the original data vectors as input

```
X = [15.7;8.4;14.7;17.3;13.3;11.4];
Y = [9.0;6.9;6.4;8.5;5.0;5.7];
C = corrcoef(X,Y)

C =
    1.0000    0.5190
    0.5190    1.0000
```

It would be worthwhile reviewing the documentation for this function by typing » `doc corrcoef`.

Solution 15.5

1. Each picture is represented by mn data points. So the data matrix containing these six pictures as columns is $mn \times 6$.
2. To find the correlation we can either use the MATLAB code we developed earlier or the command `corrcoef`. The correlation matrix will be 6×6.
3. To find the correlation between pixels, we need to take the transpose of our data matrix. This new data matrix will be $6 \times mn$, since we have mn variables (each pixel) and 6 observations (within in picture). The correlation matrix will then be $mn \times mn$.
4. High correlation between pixels equidistant from centerline.

Solution 15.6

1. To find the correlation matrix between the six images, enter
   » `corrcoef(rdfaces)`.
2. To find the correlation across pixels, enter
   » `pixels=corrcoef(transpose(rdfaces));`. We can reshape the first column of this matrix and convert it into an image using
   » `pixels1=reshape(pixels(:,1),25,25);`
   » `imagesc(pixels1)`.
   The (i, j) entry of this image tells us how similar the top-left pixel is to the (i, j) pixel. (Note: the pixels are "numbered" 1–625 going down the first column, then down the second column, and so on.) Repeating this reshape and visualization procedure on column 400 will give you the correlation between pixel 400 and each of the other pixels, for example.

Solution 15.7

1. Read, read, read ....
2. We are trying to take a linear combination of the data in order to predict whether each image is smiling or not. The vector $\mathbf{x}$ is the magic set of weights we have to use. Its size is the same as the number of pixels, so maybe it should look like a mask that we can place over an image to tell us whether it is smiling.
3. The product $\mathbf{A}^T\mathbf{A}$ is like a pixel to pixel correlation matrix, except we haven't scaled the data matrix $\mathbf{A}$. The product $\mathbf{A}^T\mathbf{b}$ is the sum of the images that are smiling.
4. The expression $\mathbf{Ax} - \mathbf{b}$ tells us the error in predicting whether a training image is smiling or not.
5. We could add up how often the predictor is correct and divide by the number of images to get an estimate of the accuracy. We would decide on a cut-off before we used it on a test dataset.
6. It is simply $\mathbf{Tx}$.
7. As before. Determine how many we got correct and average it.

Solution 15.8

You can use the solutions that are embedded in the walkthrough notebook.

```
% Excercise 15.1
poverty = [15.7; 8.4; 14.7; 17.3; 13.3; 11.4; 9.3; 10.0; 13.2; 14.7; 9.1; 12.6];
infmort = [9; 6.9; 6.4; 8.5; 5;5.7;6.2;8.3;7.3;8.1;5.6;6.8];

poverty_2 = (poverty - mean(poverty)) / std(poverty);
infmort_2 = (infmort - mean(infmort)) / std(infmort);

p = sum((1 / (length(poverty) - 1)) * (poverty_2 .* infmort_2))
```

```
p = 0.4773
```

```
% Excercise 15.3
B = [poverty_2 infmort_2]
```

```
B = 12×2
    1.1265    1.5793
   -1.4233   -0.0653
    0.7772   -0.4568
    1.6853    1.1877
    0.2882   -1.5532
   -0.3755   -1.0050
   -1.1090   -0.6134
   -0.8645    1.0311
    0.2532    0.2480
    0.7772    0.8745
```

```
A = (1 / sqrt(length(poverty) - 1)) * B
```

```
A = 12×2
    0.3396    0.4762
   -0.4292   -0.0197
    0.2343   -0.1377
    0.5081    0.3581
    0.0869   -0.4683
   -0.1132   -0.3030
   -0.3344   -0.1850
   -0.2607    0.3109
    0.0764    0.0748
    0.2343    0.2637
```

```
% Excercise 15.4
C = transpose(A) * A
```

```
C = 2×2
    1.0000    0.4773
    0.4773    1.0000
```

```
corrcoef(poverty, infmort)
```

```
ans = 2×2
    1.0000    0.4773
    0.4773    1.0000
```

```
load("face_bases.mat")
```

```
faces = test_images(:, :, 342:347);
dfaces = imresize(faces, [25 25]);
rdfaces = reshape(dfaces, size(dfaces, 1) * size(dfaces, 2), size(dfaces, 3));
size(rdfaces)
correl_imgs = corrcoef(rdfaces);
```

```
correl_pixels = corrcoef(transpose(rdfaces));
```

```
colormap('gray'); imagesc(reshape(correl_pixels(500, :), [25 25]));
```

```
% Machine Learning Now
load('smiles.mat')
```

```
% Training
training_dat = reshape(train_data, [size(train_data, 1) * size(train_data, 2) size(train_data, 3)])';
magic = (training_dat' * training_dat) \ (training_dat' * smile_flag_train);
```

```
test_dat = reshape(test_data, [size(test_data, 1) * size(test_data, 2) size(test_data, 3)])';
predicted_smiles_float = test_dat * magic;
predicted_smiles_bool = predicted_smiles_float > 0.5
```

```
predicted_smiles_bool = 500×1 logical array
   0
   1
   1
   1
   0
   1
   0
   0
   1
   1
```

```
disp("Success rate on test dataset:")
```

```
Success rate on test dataset:
```

```
mean(predicted_smiles_bool == smile_flag_test)
```

```
ans = 0.9080
```