Ellis Wright

Feedback sheet: Assignment # 5

This feedback sheet is meant to help you to improve your assignment reports, and to understand the grading criteria.

Part of the work of an applied mathematician is to communicate with experts that may not be mathematicians. Therefore, it is important to develop one's reporting skills. In this course, the quality of the reporting is an important element in grading.

Below, a list of important points concerning the reporting are given, and it also serves as a rubric for feedback.

1. The report should be self-explanatory, not an answer to a question. This means that you have to explain what the problem is that you are addressing. Preferably, do not just copy the assignment but explain in your own words what the assignment asks you to do. **A printed code is not a sufficient answer!**

   Your grade:

   ☐ Excellent: No need to improve
   ☐ Good: Acceptable, leaving space for improvement
   ☐ Below standard : Requires more work

2. The report should be carefully typeset, paying attention to the layout. Sloppy reporting gives a negative impression and raises questions about the correctness of the results, so one needs to pay attention on the visual quality.

   Your grade:

   ☐ Excellent: No need to improve
   ☐ Good: Acceptable, leaving space for improvement
   ☐ Below standard : Requires more work

3. Well planned graphs and plots are an integral part of good reporting. When you include a plot in the report, pay attention to the following:

   (a) Make sure that the fonts are of readable size. The default font in Matlab is not acceptable, so increase the font size, e.g., using the command `set(gca,'FontSize',15)`.

   (b) Lines and curves need to be thick enough to be readable, and markers must be large enough.

   (c) Pay attention to axes. For instance, if you plot a circle, make sure that it does not appear as an ellipse, using commands like `axis('square')` or `axis('equal')`. Also, think about the scaling of the axis so that they convey the information you want. Sometimes a logarithmic scale is more informative.

   (d) Select the graphical output carefully. Is a histogram better than a continuous curve? Discrete quantities may be better represented by dots or bars than by continuous curves etc.

(e) Consider marking the axes with a label, using commands `xlabel, ylabel`.

Your grade:

- ☐ Excellent: No need to improve
- ☐ Good: Acceptable, leaving space for improvement
- ☐ Below standard : Requires more work

4. Technical correctness. Each assignment will be checked for correctness. If you get a grade less than excellent, go and check the comments on the report.

   Your grade:

   - ☐ Excellent: No need to improve
   - ☐ Good: Acceptable, leaving space for improvement
   - ☐ Below standard : Requires more work

5. Code: Including a code, or a piece of it is recommended when applicable. Pay attention on your code: A code with no comments is below standard. Well documented code is a great asset for yourself, too!

   Your grade:

   - ☐ Excellent: No need to improve
   - ☐ Good: Acceptable, leaving space for improvement
   - ☐ Below standard : Requires more work

MATH 444 **Due Date: 04/17/2022**

Assignment 5

Ellis Wright (enw30)

1 Introduction

The goal of this assignment was to classify 350 images into 5 distinct patterns. An image is a 128 x 128 pixel grid represented as a 16,384 x 1 vector containing normalized gray-scale values within the interval [0,1].

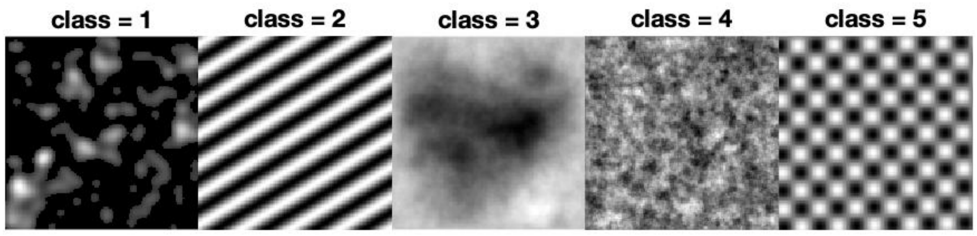
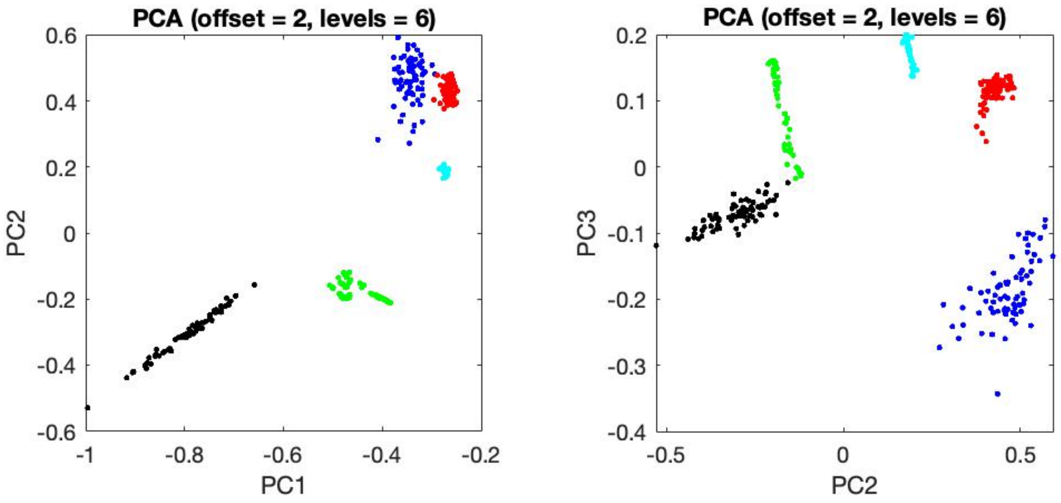


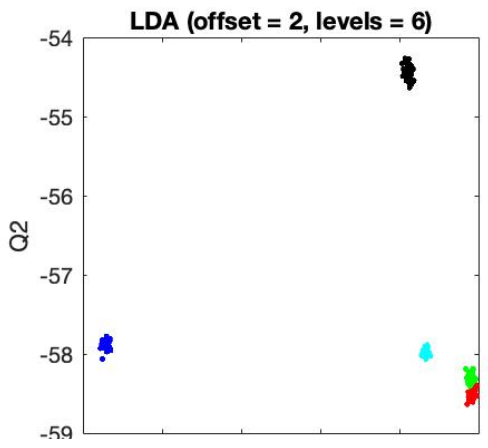
Figure 1: Patterns to classify

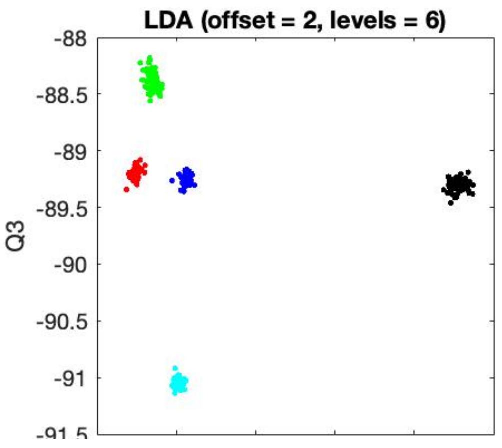
2 GLCM Algorithm

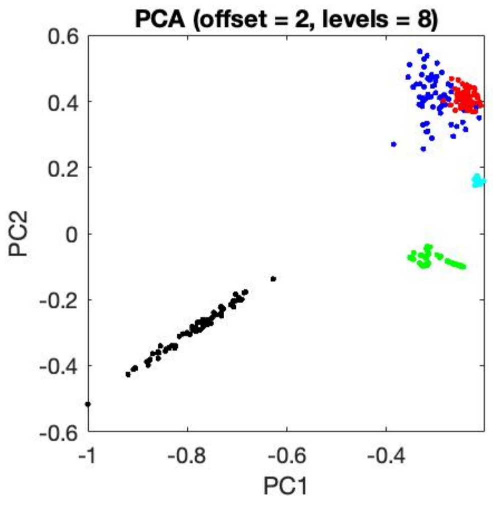
An image has a gradient of color with different patterns. To gather information about the closeness and magnitude of these gradients we use x,y offsets and a frequency matrix, respectively. Putting the values of the pixels into k bins will coarsen the images into k colors or 'levels.' Then by making a k x k matrix, we can store the frequency of each gradient with the given offsets. The ijth coordinate is the number of times there is a gradient from i to j over the distance x and y pixels. By using different offsets we can get different information- like gradients going to the right, above, diagonal, etc.. Now, we are using the GLCM matrix instead of the pixels to represent the image. We can calculate the first several principle components and linear discriminant to see how well they separate the classes. Dependng on the number of gradients I want to collect (m) I reduce the dimensional from 16,384 to 128m to 2.

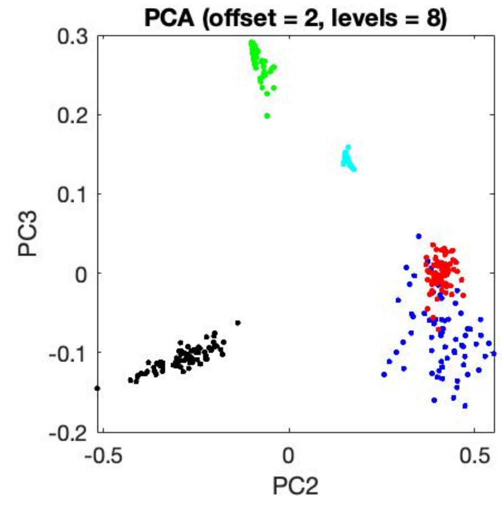
3 Experimental Results

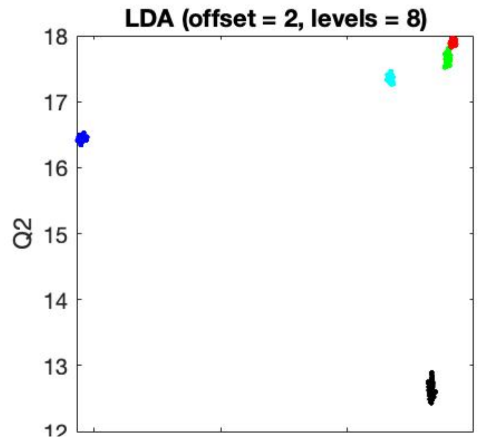


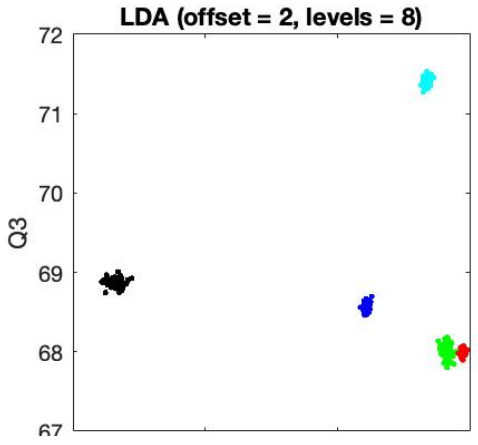

LDA (offset = 2, levels = 6)
Q2
-54
-55
-56
-57
-58
-59

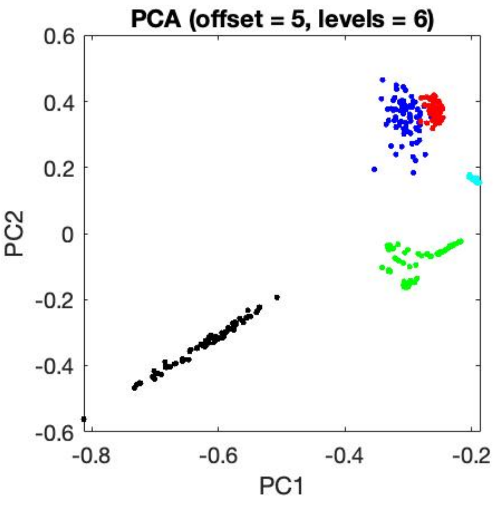

LDA (offset = 2, levels = 6)
Q3
-88
-88.5
-89
-89.5
-90
-90.5
-91
-91.5

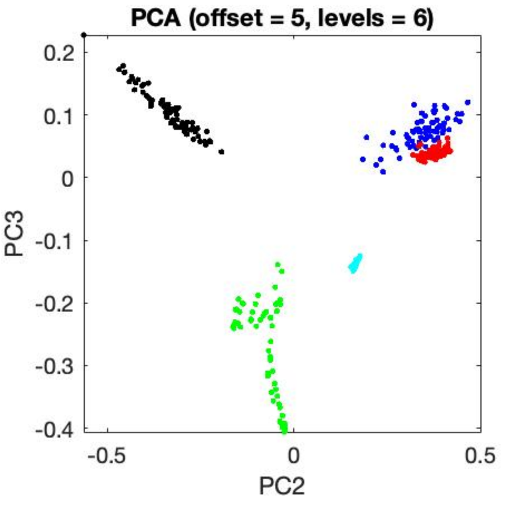

PCA (offset = 2, levels = 8)
PC2
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-1
-0.8
-0.6
-0.4
PC1

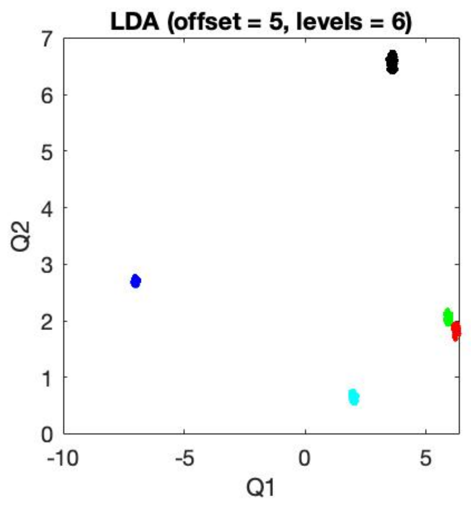

PCA (offset = 2, levels = 8)
PC3
0.3
0.2
0.1
0
-0.1
-0.2
-0.5
0
0.5
PC2

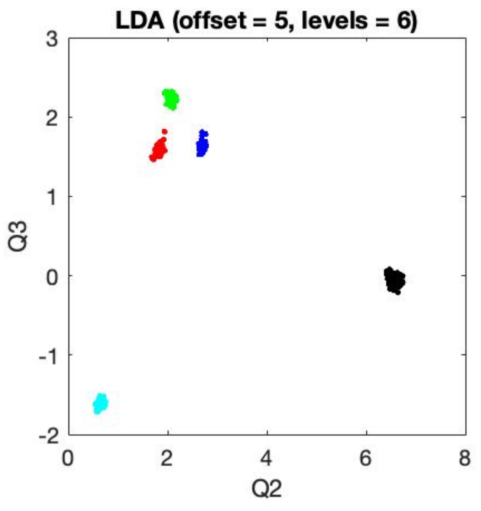

LDA (offset = 2, levels = 8)
Q2
18
17
16
15
14
13
12


LDA (offset = 2, levels = 8)
Q3
72
71
70
69
68
67

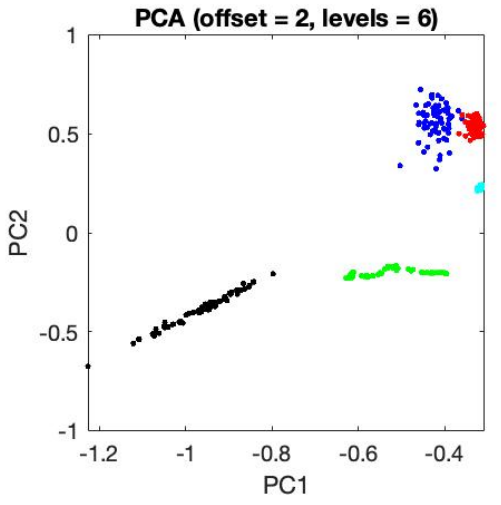

PCA (offset = 5, levels = 6)
PC2
PC1
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-0.6
-0.4
-0.2

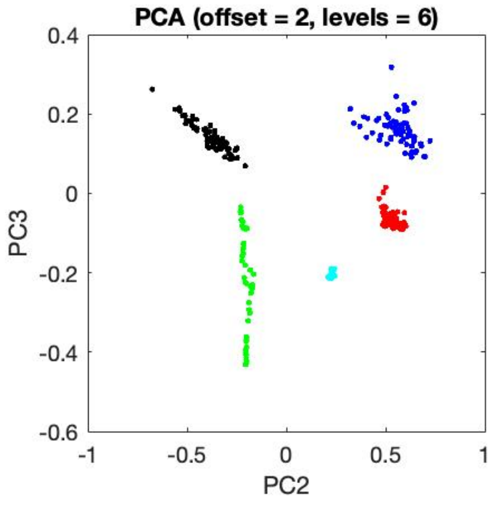

PCA (offset = 5, levels = 6)
PC3
PC2
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
0
0.5

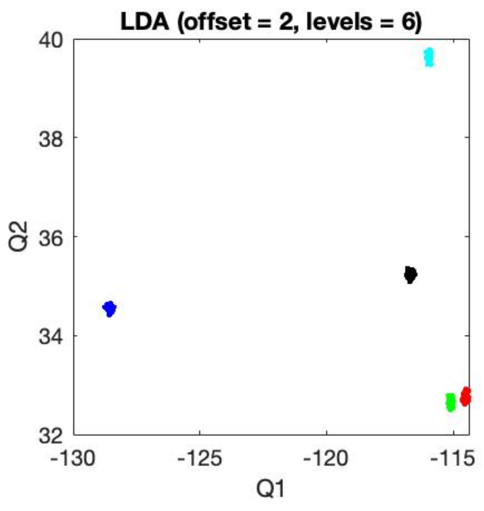


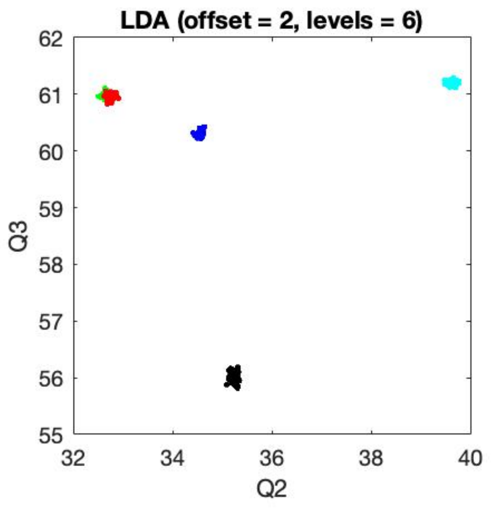


Next, I wanted to try to add more features so I tried 5 instead of 3


PCA (offset = 2, levels = 6)
PC1
PC2


PCA (offset = 2, levels = 6)
PC2
PC3





4 Conclusion

Based on the results it looks like the information among classes are well preserved all the way down to 2 dimensions for both PCA and LDA. LDA is better based off of the compactness of the clusters. It is hard to see the PCAs getting better from 6 to 8 levels but it looks like the LDA clusters get slightly smaller. Using 5 features instead of 3 makes the LDA project onto points and the PCA looks like it would be easily clustered. We can conclude that using glcm's for pattern recognition are an effective way of categorizing the data. In order to be sure an independent test set would have to be used.

5 Code

Documentation of new code used

```
%CODE TO GENERATE FIGURES
load('TestImages.mat')
```

```
%set parameters
offset = 2;
num_levels = 8;

%get principle component and linear discriminant data transformations
[PC,LD] = image_mine(X,I,num_levels,offset);

%plot them
plot_pcas(PC,I,offset,num_levels);
plot_lda(LD,I,offset,num_levels)



%FUNCTION TO GET PRINCIPLE COMPONENTS AND LINEAR DISCRIMINANT FROM GLCM MATRIX
function [PC,LD] = image_mine(data,labels,num_levels,offset)

    %coarsen the images
    C = coarsen_mat(data,8);

    %get the size
    [s1,s2] = size(data);

    %intialize to 0
    glcm = zeros(3*num_levels^2,s2);

    %for each image, transform it into a glcm vector k^2 x 1
    for i = 1:length(data(1,:))

        img = reshape(C(:,i),128,128)';

        %try with 3 or 5 different lengths
        glcm_1 = reshape(GLCM(img,offset,0,num_levels,sqrt(s1)),1,num_levels^2);
        glcm_2 = reshape(GLCM(img,0,offset,num_levels,sqrt(s1)),1,num_levels^2);
        glcm_3 = reshape(GLCM(img,offset,offset,num_levels,sqrt(s1)),1,num_levels^
       % glcm_4 = reshape(GLCM(img,2*offset,offset,num_levels,sqrt(s1)),1,num_leve
       % glcm_5 = reshape(GLCM(img,offset,2*offset,num_levels,sqrt(s1)),1,num_leve

        %use this dimenstionality instead
        glcm(:,i) = [glcm_1,glcm_2,glcm_3];%,glcm_4,glcm_5]';

    end
    glcm = glcm ./ sum(glcm,'all');

    %get feature vectors
    [U,~,~] = svd(glcm);
    %get principle components
    Z = U'*glcm;

    PC = Z(1:4,:);

    %get LDA space
    q = LDA(glcm,labels);

    LD = q'*glcm;

end


%FUNCTION TO PLOT PCs
function [] = plot_pcas(PC4,labels,offset,num_levels)

    figure(1)
    tiledlayout(1,2)
    nexttile
    t = (labels == 1);
    plot(PC4(1,t),PC4(2,t),'k.')
    xlabel('PC1')
    ylabel PC2
    title('PCA (offset = '+ string(offset) + ', levels = ' + string(num_levels)+')
    hold on
    t = (labels == 2);
    plot(PC4(1,t),PC4(2,t),'g.')
    t = (labels == 3);
    plot(PC4(1,t),PC4(2,t),'b.')
    t = (labels == 4);
    plot(PC4(1,t),PC4(2,t),'r.')
    t = (labels == 5);
    plot(PC4(1,t),PC4(2,t),'c.')
    axis square

    hold off
    nexttile
    t = (labels == 1);
    plot(PC4(2,t),PC4(3,t),'k.')
    xlabel('PC2')
    ylabel PC3
    title('PCA (offset = '+ string(offset) + ', levels = ' + string(num_levels)+')
    hold on
    t = (labels == 2);
```

```
        plot(PC4(2,t),PC4(3,t),'g.')
        t = (labels == 3);
        plot(PC4(2,t),PC4(3,t),'b.')
        t = (labels == 4);
        plot(PC4(2,t),PC4(3,t),'r.')
        t = (labels == 5);
        plot(PC4(2,t),PC4(3,t),'c.')
        axis square
    end


    %FUNCTION TO PLOT LDs
    function [] = plot_lda(Z,labels,offset,num_levels)

        figure(2)
        tiledlayout(1,2)
        nexttile
        t = (labels == 1);
        plot(Z(1,t),Z(2,t),'k.')
        xlabel Q1
        ylabel Q2
        title('LDA (offset = '+ string(offset) + ', levels = ' + string(num_levels)+')
        hold on
        t = (labels == 2);
        plot(Z(1,t),Z(2,t),'b.')
        t = (labels == 3);
        plot(Z(1,t),Z(2,t),'g.')
        t = (labels == 4);
        plot(Z(1,t),Z(2,t),'r.')
        t = (labels == 5);
        plot(Z(1,t),Z(2,t),'c.')
        hold off
        axis square

        nexttile
        t = (labels == 1);
        plot(Z(2,t),Z(3,t),'k.')
        xlabel Q2
        ylabel Q3
        title('LDA (offset = '+ string(offset) + ', levels = ' + string(num_levels)+')
        hold on
        t = (labels == 2);
        plot(Z(2,t),Z(3,t),'b.')
        t = (labels == 3);
        plot(Z(2,t),Z(3,t),'g.')
        t = (labels == 4);
        plot(Z(2,t),Z(3,t),'r.')
        t = (labels == 5);
        plot(Z(2,t),Z(3,t),'c.')
        axis square
    end

    %FUNCTION TO GENERATE GLCM
    %function to create glcm matrix taken from slides
    function G = GLCM(mat,mu,nu,k,size)

        %offset will be nans
        C_shift= NaN(size,size);

        %take only the numbers that will exist 1 to end - offset
        C_shift(1:size-mu,1:size-nu) = mat(1+mu:size,1+nu:size);

        %initialize frequnecy to 0
        G = zeros(k,k);

        for i = 1:k

            %indices for pixels in ith layer
            Ii = (mat == i);

            for j = 1:k

                %indices for pixels in jth layer
                Ij = (C_shift == j);

                %set G(i,j) to the total times this is true in an image
                G(i,j) = sum(sum(Ii .* Ij));

            end
        end

        %normalize
        G = G * (1/sum(G,'all'));
    end

    %CODE TO COARSEN MATRIX
    %function that discretizes matrix into k boxes
    function C = coarsen_mat(mat,k)

        %find interval
        step = 1/k;

        %for each interval, update their values
        for i = 1:k
            mat((mat < step*i) & (mat >= step*(i-1))) = i;
        end
        C = mat;
    end
```