Tutorial: Applications of Random Matrix Theory to PCA

Jay Jojo Cheng, Sukyoung Cho, Mostafa Hassan

August 2019

1 Abstract

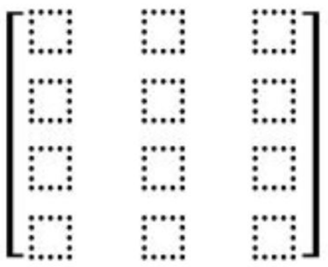
An image or other data matrix may easily be contaminated by noise, and it is still challenging process to denoise an image. In previous lectures, we have seen how truncated singular value decomposition (SVD) and principal component analysis can be used in dealing with measurements in the presence of noise. Here we introduce ideas from the theory of random matrices which can help us quantify what noise ‘looks like.’ In particular, we will see that when E is an $m \times n$ matrix with random entries, the eigenvalues of $\frac{1}{n}EE^T$ follow a particular distribution called the Marchenko-Pastur distribution. This in turn will suggest an algorithm for choosing a cutoff value for the truncated SVD.

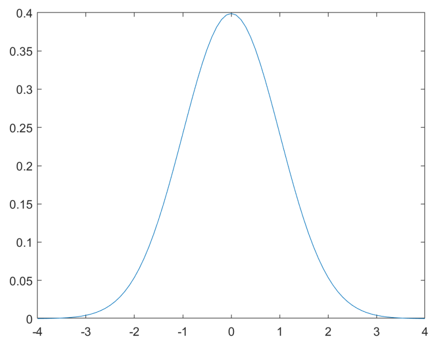
1.1 Learning objectives

1. Understand what a probability distribution is and generate random matrices in MATLAB
2. Understand through simulations how the eigenvalues of noise (a random matrix) take on the Marchenko-Pastur distribution
3. Learn to use a provided script for generating noise for an image denoising it
4. Compare distribution of singular values in the original image, corrupted image, and denoised image
5. Understand the effect of the level of noise of recovery
6. Students will understand the basic strategy of the denoising algorithm
7. Compute the largest and smallest expected eigenvalues in the Marchenko-Pastur distribution
8. Use the Marchenko-Pastur law to compute the appropriate cutoff value for truncated SVD

2 Introduction to basic probability

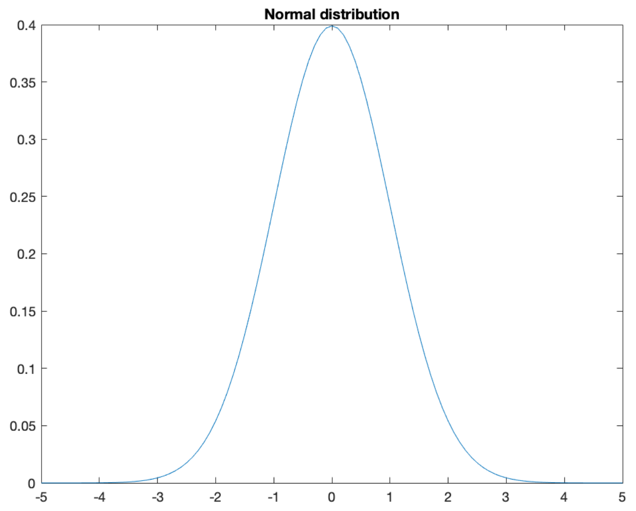
Welcome. In this tutorial we will discuss what random matrices are, how they can be used as a model for noise, and how the distribution of their singular values help us denoise PCA. What is a random matrix? Recall that a matrix is a $m \times n$ array of numbers.

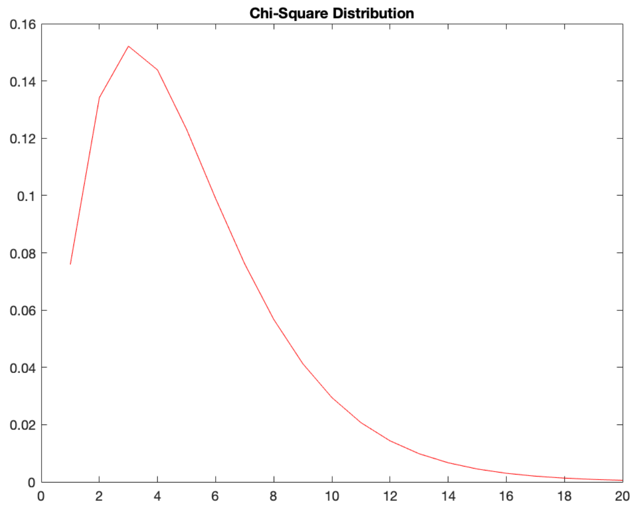


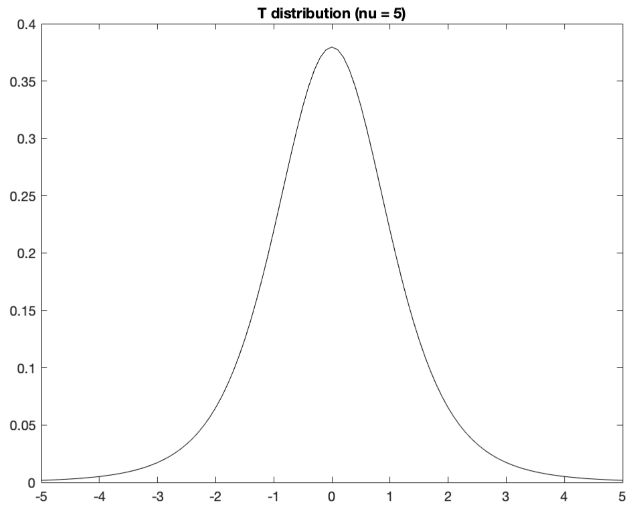


In its simplest form, a random matrix is a matrix where the entries come from a random distribution and are independent of each other, meaning that the value of one entry has no bearing on the value of another. The most familiar random distribution is the **normal distribution**, also known as a Gaussian or bell curve. The bell curve shown above is the **probability distribution function** of a normal distribution. To benefit from this module, all you need to understand about probability distribution is to imagine that it determines the properties of a random number generator. This generator generates a value along the horizontal axis, with probability proportional to the height of the graph on the vertical axis. For example, since the curve is highest close to 0, values close to zero are most likely to be drawn, while values on the left or right are less likely. The further away from the center, the less likely it is to be. Since this curve is symmetric about the origin, the average value will be 0.

Warm-up question 1: Here we provide you code for drawing points from distributions. Run each code snippet below this box and match it to one of the following three distributions.







```
%% Snippet 1
% Create Data
V = 10;
R = chi2rnd(V, 1, 100);
X = histcounts(R,20);
% Residual Norm Cost Function
RNCF = @(v) norm(X - chi2pdf((1:length(X)),v));
% Estimate Parameter
Ve = fminsearch(RNCF, rand);
figure(1)
bar((1:length(X)), X/sum(X))

%% Snippet 2
r = trnd(10*ones(1,100));
hist(r);

%% Snippet 3
r = normrnd(0,1,100,1);
hist(r);
```

In the above examples, we generated 100 values from each distribution. Now let's pick 25 points, but this time we will place the points into a 5x5 random matrix. Please type and run the following in your MATLAB console.

```
pd = makedist('Normal');

r = random(pd,[5,5]);
```

Verify for yourself, by running the running the second line several times, that the numbers change and are indeed random. The values in these entries follow the normal distribution that we have seen above. That is, if we generate a large number of these matrices and create a histogram of the entries, we will be able to approximate the normal distribution.

3 Optional historical note

Random matrices were first studied by the Scottish statistician John Wishart in the 1920s in estimating large covariance matrices - big data in the early days! [5] But much progress and interest in the mathematical theory was not made until the 1950s, when Nobel Laureate Eugene Wigner studied it as a model for the repulsion in the spectra of heavy nuclei. [3, 4]

Since then, random matrices have found applications in other areas of physics (quantum entanglement, gauge theory, cosmology, statistical physics), mathematics (Riemann hypothesis and number theory, free probability, combinatorics), statistics (multivariate statistics, principal component analysis, Bayesian model selection), information theory (signal processing, wireless communications), and biology (sequence matching, RNA folding). This is by no means an exhaustive list, and the curious reader can refer to [1] for more of the history and applications.

4 Random matrices as a model for noise

We have learned from the previous lectures that the data could easily be contaminated by the noise. For example, if the original data was very structured with a clear pattern, the addition of noise make the pattern more diffuse. The general strategy that we have seen for minimizing the effect of noise has been to regularize the data with truncated SVD, by choosing a threshold for the rank of the matrix, p. The main issue is that this p is generally not known and must be found through cross-validation; however in many situations, such as unsupervised learning, the ground truth is not known. Thus, we must estimate this p some other way. In this section, we will briefly review the key concepts of principal components analysis and then discuss the distribution of the eigenvalues of noise.

Warm-up question 2: True or False: If the isotropic noise (without specific direction) is added to the clean data set and created the diffuse data, the directions of principal component changes.

Activity question 1: Suppose there is a matrix $A = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}$ and its singular value decomposition is $A = USV^T$. Find the singular values of A by hand.
Hint 1: Try to calculate the eigenvalues of AA^T or A^TA.
Hint 2: Use the fact that $\det(AA^T)$ is the product of the eigenvalues.
Hint 3: The sum of eigenvalues is equal to the trace.

4.1 PCA review and eigenvalues of noise

Given some data X in an $M \times N$ matrix, finding the principal components involve using the singular value decomposition $X = USV^T$ to find $XX^T = US^2U^T$. Thus, the square of the singular values are the eigenvalues of XX^T and the left singular vectors (columns of U) correspond to the principal components. The variance associated with the kth principal component is then $\frac{\sigma_k^2}{N}$. If we suppose that our data is low rank of rank p, then we can approximate X with $\tilde{U}\tilde{S}\tilde{V}$, where we use the first p columns of U and V, and the first p singular values of S.

Now suppose we have an $M \times N$ random matrix E, which represents measurement error or noise, such that in practice when we try to measure X, we get $\tilde{X} = X + E$. We can reduce the effect of noise by using the largest p principal components, but how do we actually find such a p in practice? One approach, which is the one explored here, is to try to understand the distribution of the singular values of E, or equivalently, the eigenvalues of EE^T. This is where a beautiful result from random matrix theory comes into play.

4.2 Distribution of noise eigenvalues

Given a random matrix E, what happens when we take the eigenvalues of EE^T? First, we will explore this question numerically for different sized matrices. Please paste the following code into the MATLAB console to generate histograms of noise matrices of increasing dimension.

```
% 10*10
pd = makedist('Normal');
r = random(pd,[10,10]);
A = 1/(10*10)*r*r'; %1/n*E*E'
e = eig(A);
histogram(e,5);
title('10 X 10 random matrix');

% 50*50
pd = makedist('Normal');
r = random(pd,[50,50]);
A = 1/(50*50)*r*r'; %1/n*E*E'
e = eig(A);
histogram(e,15);
title('50 X 50 random matrix');

% 250*250
pd = makedist('Normal');
r = random(pd,[250,250]);
A = 1/(250*250)*r*r'; %1/n*E*E'
e = eig(A);
histogram(e,30);
title('250 X 250 random matrix');

% 1250*1250
pd = makedist('Normal');
r = random(pd,[1250,1250]);
A = 1/(1250*1250)*r*r'; %1/n*E*E'
e = eig(A);
histogram(e,30);
title('1250 X 1250 random matrix');
```

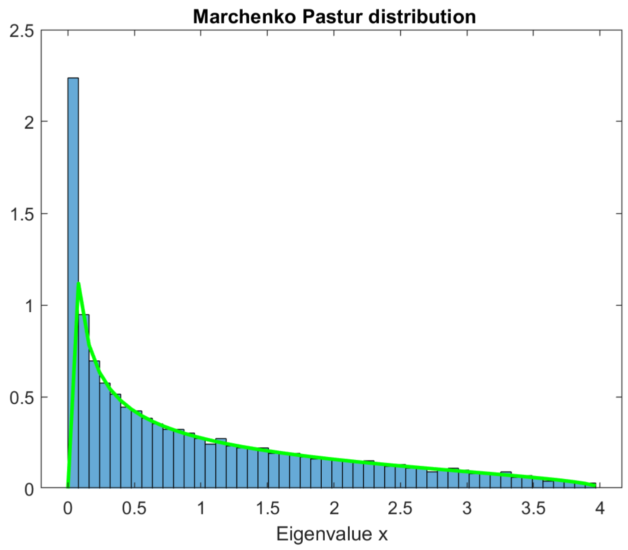
We can rerun the code for the large matrices several times and notice that the distribution is fairly stable. This is because the Marchenko-Pastur Law tells us that as $m, n \to \infty$ and $m/n \to \gamma$, the probability distribution function of the eigenvalues of $\frac{1}{n}EE^T$ converges to the function

$$g(x) = \begin{cases} \frac{1}{2\pi\sigma^2}\frac{\sqrt{(\lambda_+ - x)(x-\lambda_-)}}{\gamma x} I_{\{x\in[\lambda_-,\lambda_+]\}}, & \text{for } \gamma \in [0,1] \\ \frac{1}{2\pi\sigma^2}\frac{\sqrt{(\lambda_+ - x)(x-\lambda_-)}}{\gamma x} I_{\{x\in[\lambda_-,\lambda_+]\}} + (1-\frac{1}{\gamma})\delta(x), & \text{for } \gamma > 1 \end{cases}, \tag{1}$$

where $\sigma^2 = 1$, and $\lambda_- = \sigma^2(1-\sqrt{\gamma})^2$ and $\lambda_+ = \sigma^2(1+\sqrt{\gamma})^2$.[1]

[1]This is the same parameter as the σ^2 briefly seen in video and exercises relating to the Gaussian kernel regression (Video 6.3). For students with some background in probability or statistics, this parameter is called a **variance**. The Marchenko-Pastur Law is universal in the sense that holds when the noise distribution is any probability distribution with a finite variance: $\sigma^2 < \infty$. Students without background in probability or statistics can intuitively understand σ^2 as controlling the width of the distribution.

This equation looks intimidating at first, but do not worry! We will walk through each part. First, let's look at what the theoretical distribution looks like when $\gamma = M/N = 1$ and $\sigma^2 = 1$ (green). Here we have also superimposed a histogram of the eigenvalues of a 1250×1250 random matrix for comparison (blue). Notice how the edges of this distribution are at $\lambda_- = \sigma^2(1-\sqrt{\gamma})^2 = 1 * (1-1)^2 = 0$ and $\lambda_+ = \sigma^2(1+\sqrt{\gamma})^2 = 1 * (1+1)^2 = 4$.



Let's focus on the first case, since here $\gamma = 1$. We see that from the indicator function, the smallest eigenvalue will be λ_- and the largest will be λ_+. Indeed, if x is below λ_- or above λ_+ we would take the square root of a negative number. Thus, we have no real roots/ignore these possibilities. The $\lambda_\pm$ are both defined by the parameters σ^2, and $\gamma = M/N$. Thus, we see that the shape of the matrix E should affect $\lambda_\pm$ and thus the width of the distribution. Please explore this computationally in the first activity question.

Activity question 2:

1. We have just seen what happens when E is square. That is, $m, n \to \infty$ and $m/n = \lambda = 1$. Using the code provided for finding the distribution of noise eigenvalues of $\frac{1}{n}EE^T$ (marchenkopasturlaw.m), find what the eigenvalue distribution of rectangular noise matrices looks like with $m/n = 0.3, 0.6, 1.3, 1.6$. To receive full credit, repeat this 2 times for each λ, first with small m and small n and then with large m and large n. You will have a total of 8 graphs.

2. How does the ratio between the number of rows to number of columns of E affect the spread of eigenvalues?

Now that we have a sense for what the distribution looks like, let's continue unpacking equation 1. Here it is again:

$$g(x) = \begin{cases} \frac{1}{2\pi\sigma^2}\frac{\sqrt{(\lambda_+-x)(x-\lambda_-)}}{\gamma x} I_{\{x\in[\lambda_-,\lambda_+]\}}, & \text{for } \gamma \in [0,1] \\ \frac{1}{2\pi\sigma^2}\frac{\sqrt{(\lambda_+-x)(x-\lambda_-)}}{\gamma x} I_{\{x\in[\lambda_-,\lambda_+]\}} + (1-\frac{1}{\gamma})\delta(x), & \text{for } \gamma > 1 \end{cases},$$

We have gotten a sense of the shape of $\frac{1}{2\pi\sigma^2}\frac{\sqrt{(\lambda_+ - x)(x-\lambda_-)}}{\gamma x} I_{\{x\in[\lambda_-,\lambda_+]\}}$ from the previous activity. Now, for the other case when $\gamma > 1$, the only difference is that we add $(1-\frac{1}{\gamma})\delta(x)$. Simply put, this is a correction term because $\frac{1}{n}EE^T$ will have many zero eigenvalues, since $\gamma > 1$ implies that $m > n$. $\frac{1}{n}EE^T$ is a square matrix of dimension $m \times m$, so it will have m eigenvalues, but since the rank of E is at most n, n/m proportion of the eigenvalues will be nonzero and $1 - n/m = 1 - \frac{1}{\gamma}$ eigenvalues will be zero.

5 Denoising algorithm

Now that we know what the distribution of noise eigenvalues looks like, we can come up with a denoising algorithm for truncating them. The strategy follows a paper by Veraart et al. (2016). [2] The essential principle is to use truncated SVD regularization to decrease the variance of the image, using the Marchenko-Pastur distribution as a guide for selecting the appropriate cutoff value.

Warm-up question 3: Recall the non-zero singular values of a matrix X are (select all that apply):

1. Eigenvalues of XX^T
2. Squares of eigenvalues of X^TX
3. Square roots of eigenvalues of XX^T
4. Square roots of eigenvalues of X^TX

Since the singular values of X are closely related to the eigenvalues of XX^T, we can study the spectrum of XX^T in light of the Marchenko-Pastur distribution to determine an appropriate cutoff. Suppose X is an $m \times n$ noisy matrix and we compute the SVD $X = \sqrt{n}ULV^T$, where we have introduced a scalar factor for convenience (since we will divide by n shortly). We work with the covariance matrix

$$\Sigma = \frac{1}{n}XX^T = UL^2U^T,$$

whose eigenvalues will be its singular values ℓ_i^2, the squared diagonal elements of L.

It turns out that if the noise components for each entry of X are independent (the value of one entry have no effect on another entry) and identically distributed (coming from the same probability distribution), and we assume the lowest $n-p$ eigenvalues are pure noise, then those noise eigenvalues will follow the Marchenko-Pastur distribution, seen previously.

It can be shown that the average value of this distribution is the parameter σ^2. We also know that the difference of the largest and smallest eigenvalue, or the width of the spectrum, is by expanding their definitions

$$\begin{aligned}\lambda_+ - \lambda_- &= \sigma^2(1+\sqrt{\gamma})^2 - \sigma^2(1-\sqrt{\gamma})^2 \\ &= 4\sqrt{\gamma}\sigma^2\end{aligned}$$

We can take advantage of this fact. Using the estimate of $\lambda_+ = \lambda_{p+1}$ and $\lambda_- = \lambda_n$, we have

$$\sigma^2 = \frac{\lambda_{p+1} - \lambda_n}{4\sqrt{m/n}},$$

the condition $\sum_{i=p+1}^{n} \lambda_i/(n-p) = \sigma^2$ reduces to

$$\sum_{i=p+1}^{n} \lambda_i = 0.25\sqrt{n(n-p)}(\lambda_{p+1} - \lambda_n),$$

hence the cutoff below. We will then start iterating with $p = 1$ and terminate when the estimate of the left side in the equation above are greater than or equal to the right side.

```
Initialize number of significant eigenvalues, p=1
Subtract off the mean
Compute the SVD [U,S,V] = svd(X)
Use SVD to find the eigvals of XX'/n
Set the largest noise eigenvalue to be (p+1)th largest eigval
Set the smallest noise eigenvalue to be (n)th largest eigval
Compute the cutoff = 0.25*sqrt(n(n - p))*(largest eigval - smallest eigval)
While sum of n-p noise eigvals of XX'/n is smaller than cutoff
    Increment p
    Set the largest noise eigenvalue to be (p+1)th largest eigval
    Recompute cutoff

Return the truncated SVD of X up to order p, with mean added back
```

Activity question 3:

1. Run the provided denoising code in 'denoise.m' on the provided images, varying the standard deviation of noise sigma. The script 'ImageDenoising.m' will get you started. Comment on the ability of this algorithm to strike a good balance between bias and variance.

2. Look at the graphs for the eigenvalues of the image RGB components. Do you see any patterns? How does the noise level affect the MPPCA rank cutoff?

3. Try using different distributions of noise, other than the normal distribution. Are the results different? Comment on the effect of working with large samples of independent identically distributed values, such as large random noise matrices, on the exact underlying distribution used.

References

[1] Gernot Akemann, Jinho Baik, and Philippe Di Francesco. *The Oxford handbook of random matrix theory.* Oxford University Press, 2011.

[2] Jelle Veraart, Dmitry S Novikov, Daan Christiaens, Benjamin Ades-Aron, Jan Sijbers, and Els Fieremans. Denoising of diffusion mri using random matrix theory. *NeuroImage*, 142:394–406, 2016.

[3] Eugene P. Wigner. Characteristic vectors of bordered matrices with infinite dimensions. *Annals of Mathematics*, 62(3):548–564, 1955.

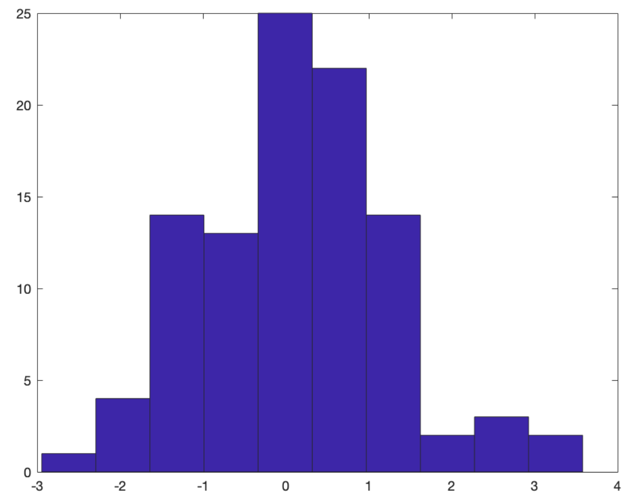
[4] Eugene P. Wigner. On the distribution of the roots of certain symmetric matrices. *Annals of Mathematics*, 67(2):325–327, 1958.

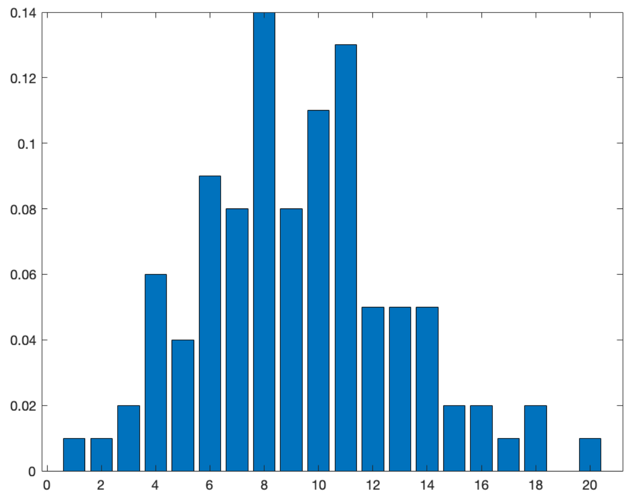
[5] John Wishart. The generalised product moment distribution in samples from a normal multivariate population. *Biometrika*, pages 32–52, 1928.

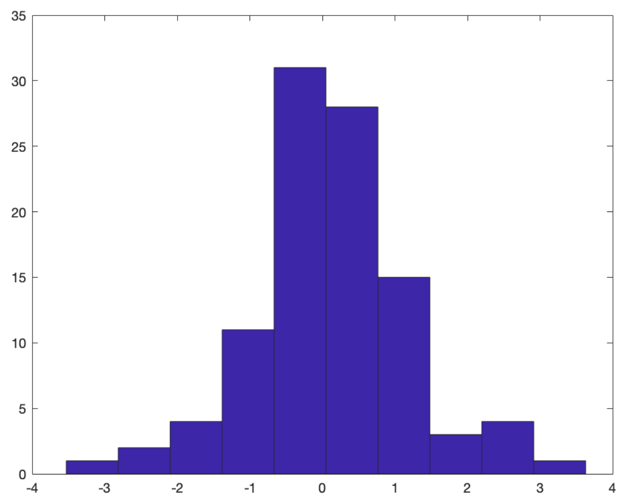
6 Appendix: Solutions

6.1 Warm-up question 1

Here are the histograms of the normal, Chi, and T distributions.







6.2 Warm-up question 2

False

6.3 Warm-up question 3

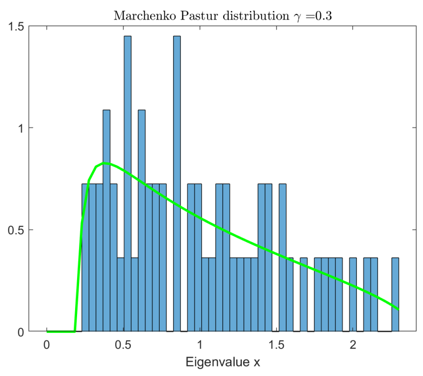
Square roots of eigenvalues of XX^T; Square roots of eigenvalues of X^TX

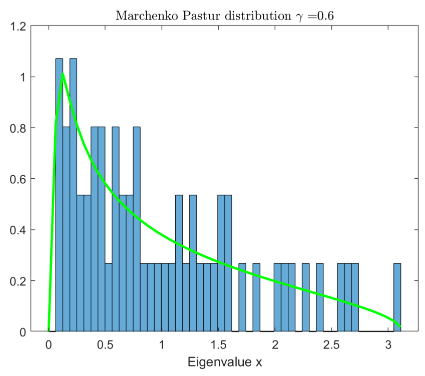
6.4 Activity question 1

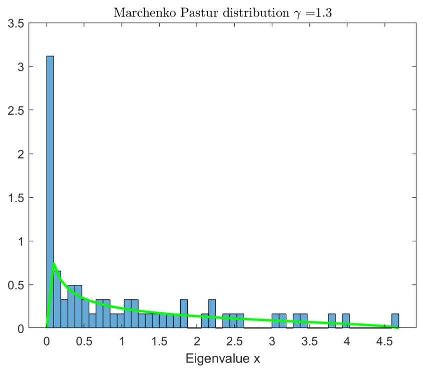
First, find the compute AA^T and find its eigenvalues. $AA^T = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$. Then we know that the sum of eigenvalues of a matrix is equal to the trace of a matrix, so the sum of eigenvalues of AA^T is $\lambda_1 + \lambda_2 = 2 + 2 = 4$. We also know that the product of eigenvalues is equal to the determinant of AA^T, so $\lambda_1\lambda_2 = 4 - 1 = 3$. Then, we see that the eigenvalues are 1 and 3.

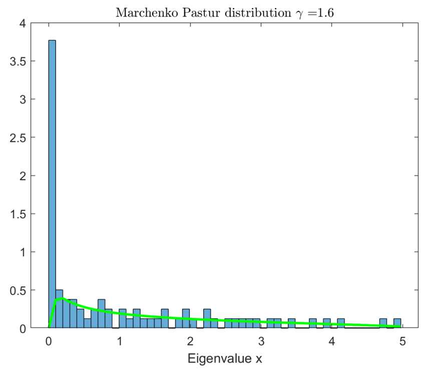
6.5 Activity question 2

1. For this question, we only need to change the settings for m and n in the script, then run it to obtain the figures. Here are the figures with small dimensions

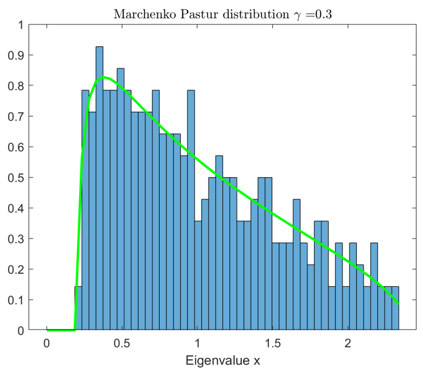


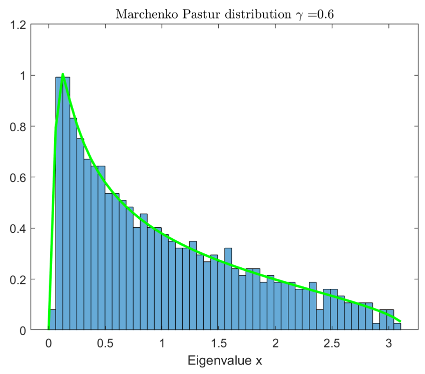


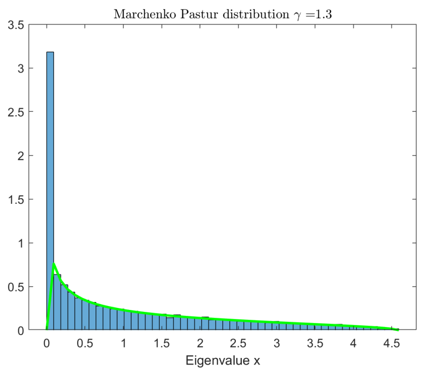


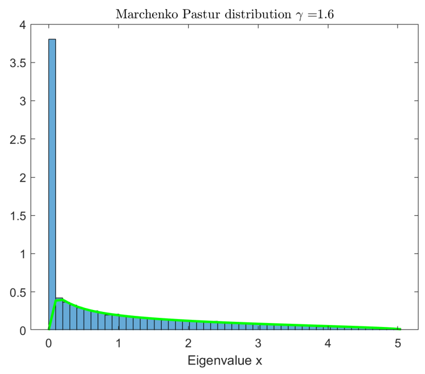


Here are the figures with large dimensions









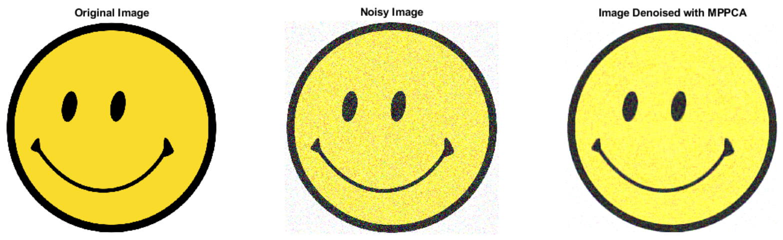
2. As the matrix gets fatter, the spread of eigenvalues gets wider.

6.6 Warm-up question 3

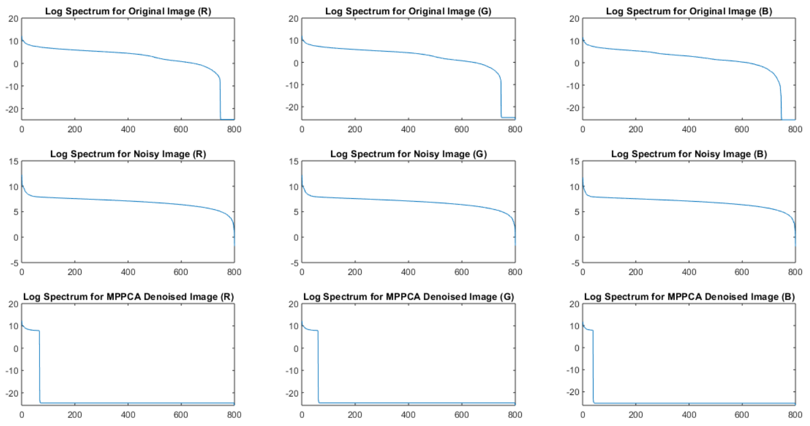
Square roots of eigenvalues of XX^T, Square roots of eigenvalues of X^TX

6.7 Activity question 3

1. Students should notice that the algorithm strikes a good balance between bias and variance, denoising the images without causing excessive blurring for reasonable noise levels.

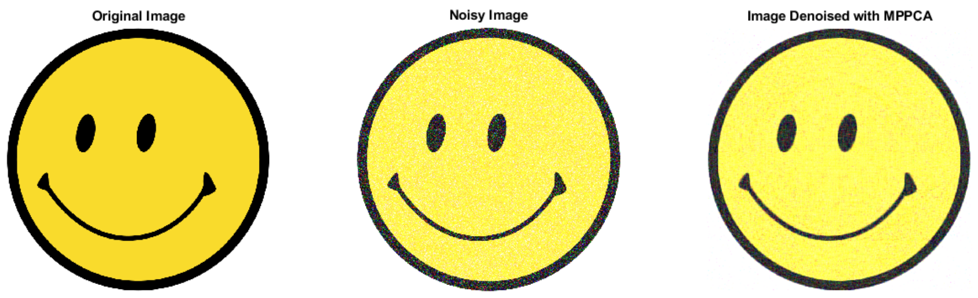


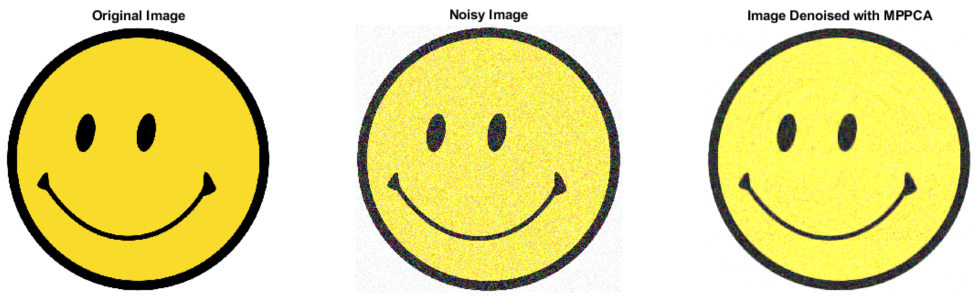
2. The noise introduces some variation in the eigenvalues, smaller ones especially, and the algorithm simply cuts off the eigenvalues at some point. At higher noise levels, the rank cutoff will be smaller, resulting in a blurry denoised image, whereas at lower noise levels, the cutoff value will be higher, resulting in clean and accurate images in the resulting denoised version.



3. Using different distributions should have little effect on the outcome, so long as the mean is zero. This reflects the central limit theorem: regardless of the underlying distributions used, the distribution of the mn samples will be approximately normal

for large m and n. The below images are using a gamma and uniform distribution, respectively, with the same mean and variance as before.


Original Image
Noisy Image
Image Denoised with MPPCA


Original Image
Noisy Image
Image Denoised with MPPCA