CIS 419/519: Applied Machine Learning Fall 2024

Homework 3

Handed Out: October 23 *Due: 7:59 pm November 6*

Name: Yueyang Li

PennKey: ezel22

PennID: 47700221

1 Declaration

- **Person(s) discussed with:** *Your answer*
- **Affiliation to the course: student, TA, prof etc.** *Your answer*
- **Which question(s) in coding / written HW did you discuss?** ***Your answer***
- **Briefly explain what was discussed.** *Your answer*

2 Multiple Choice & Written Questions

1. (a) Here's the images:

   image.png

   image01.png

   (b) Calculate the total Variance:

   $$\text{Total Variance} = 2.1 + 1.8 + 1.3 + 0.9 + 0.4 + 0.2 + 0.15 + 0.02 + 0.001 = 6.871$$

   Therefore, 75% of the total Variance is just $0.75 * 6.871 = 5.15325$
   First Eigienvalue = 2.1
   First + Second Eigienvalue = 3.9
   First + Second + Third Eigienvalue = 5.2
   Therefore, $K = 3$

   (c) A

2. (a) For this problem:

1st Iter

$d(A,1) = 7.21$, $d(A,2) = 6.08$
$d(B,1) = 3.60$, $d(B,2) = 4.47$
$d(C,1) = 8.06$, $d(C,2) = 4.24$
$d(D,1) = 5.0$, $d(D,2) = 8.25$
Cluster 1 members: $\{B, D\}$
Cluster 1 Updated Centroid: $\{7, 9\}$
Cluster 2 members: $\{A, C\}$
Cluster 2 Updated Centroid: $\{3.5, 2\}$

iter01.png

2nd Iter

$d(A,1) = 7.81$, $d(A,2) = 1.80$
$d(B,1) = 4.24$, $d(B,2) = 4.03$
$d(C,1) = 8.24$, $d(C,2) = 1.80$
$d(D,1) = 4.24$, $d(D,2) = 11.92$
Cluster 1 members: $\{D\}$
Cluster 1 Updated Centroid: $\{\frac{11}{3}, \frac{10}{3}\}$
Cluster 2 members: $\{A, B, C\}$
Cluster 2 Updated Centroid: $\{10, 12\}$

iter02.png

3. (a) B This is a kernel in charge of Blurring the original images, where each pixel's value is added with the surrounding pixels in the 3x3 grid and averaged, resulting in a blurred version of the original image.

   (b) A This kernel is essentially apply 2 times intensity to each pixel in the image, resulting in a much brighter image that matches A

   (c) C This kernel is a sobel vertical edge detector that is used to detect any vertical edges and strengthen the the comparison. It basically takes a smoothed derivatives across the edges, and extract any significant changes and treated that change as an edge.

4. (a) This problem can be computed as:
   1. Layer (a) - Conv2d:

- Input: $232 \times 232 \times 3$
- Kernel: 5×5, Stride: 1, Padding: 0
- Output channels: 5

$$\text{Output size } = \frac{232-5+2 \times 0}{1}+1=228$$

- Output: $228 \times 228 \times 5$

2. Layer (b) - ReLU:
- This layer does not change the dimensions.
- Output: $228 \times 228 \times 5$

3. Layer (c) - MaxPool2d:
- Kernel: 2×2, Stride: 2, Padding: 0

$$\text{Output size } = \frac{228-2+2 \times 0}{2}+1=114$$

- Output: $114 \times 114 \times 5$

4. Layer (d) - Conv2d:
- Input channels: 5, Output channels: 10
- Kernel: 3×3, Stride: 1 , Padding: 0

Output size $=\frac{114-3+2 \times 0}{1}+1=112$
- Output: $112 \times 112 \times 10$

5. Layer (e) - ReLU:
- No change in dimensions.
- Output: $112 \times 112 \times 10$

6. Layer (f) - MaxPool2d:
- Kernel: 2×2, Stride: 2 , Padding: 0

$$\text{Output size } = \frac{112-2+2 \times 0}{2}+1=56$$

- Output: $56 \times 56 \times 10$

7. Layer (g) - Conv2d:
- Input channels: 10, Output channels: 20
- Kernel: 3×3, Stride: 1 , Padding: 0

$$\text{Output size } = \frac{56-3+2 \times 0}{1}+1=54$$

- Output: $54 \times 54 \times 20$

8. Layer (h) - ReLU:
- No change in dimensions.
- Output: $54 \times 54 \times 20$

9. Layer (i) - MaxPool2d:
- Kernel: 2×2 , Stride: 2, Padding: 0

$$\text{Output size } = \frac{54 - 2 + 2 \times 0}{2} + 1 = 27$$

- Output: $27 \times 27 \times 20$

Therefore, dim 1 = 27

(b) **Therefore, dim 2 = 27**

(c) **Therefore, dim 3 = 20**

(d) This is how this problem will be solved:
1. Layer (a) - Conv2d (5 output channels, 5×5 kernel, 3 input channels):
- Number of parameters per output channel: $5 \times 5 \times 3 = 75$
- Total parameters: $75 \times 5 = 375$
- Plus biases (1 per output channel): 5
- Total for this layer: $375 + 5 = 380$
2. Layer (d) - Conv2d (10 output channels, 3×3 kernel, 5 input channels):
- Number of parameters per output channel: $3 \times 3 \times 5 = 45$
- Total parameters: $45 \times 10 = 450$
- Plus biases (1 per output channel): 10
- Total for this layer: $450 + 10 = 460$
3. Layer (g) - Conv2d (20 output channels, 3×3 kernel, 10 input channels):
- Number of parameters per output channel: $3 \times 3 \times 10 = 90$
- Total parameters: $90 \times 20 = 1800$
- Plus biases (1 per output channel): 20
- Total for this layer: $1800 + 20 = 1820$

Summing Up the Parameters
Total learnable parameters in the network:

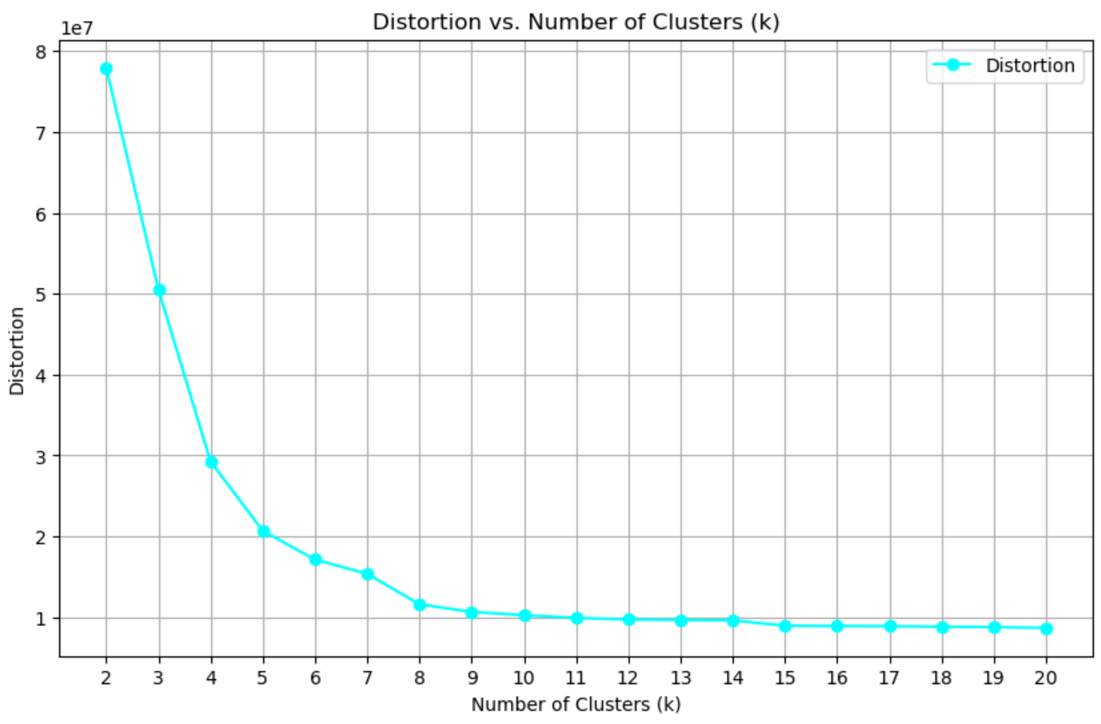
$$380 + 460 + 1820 = 2660$$

So, Number of learnable parameters = 2660.

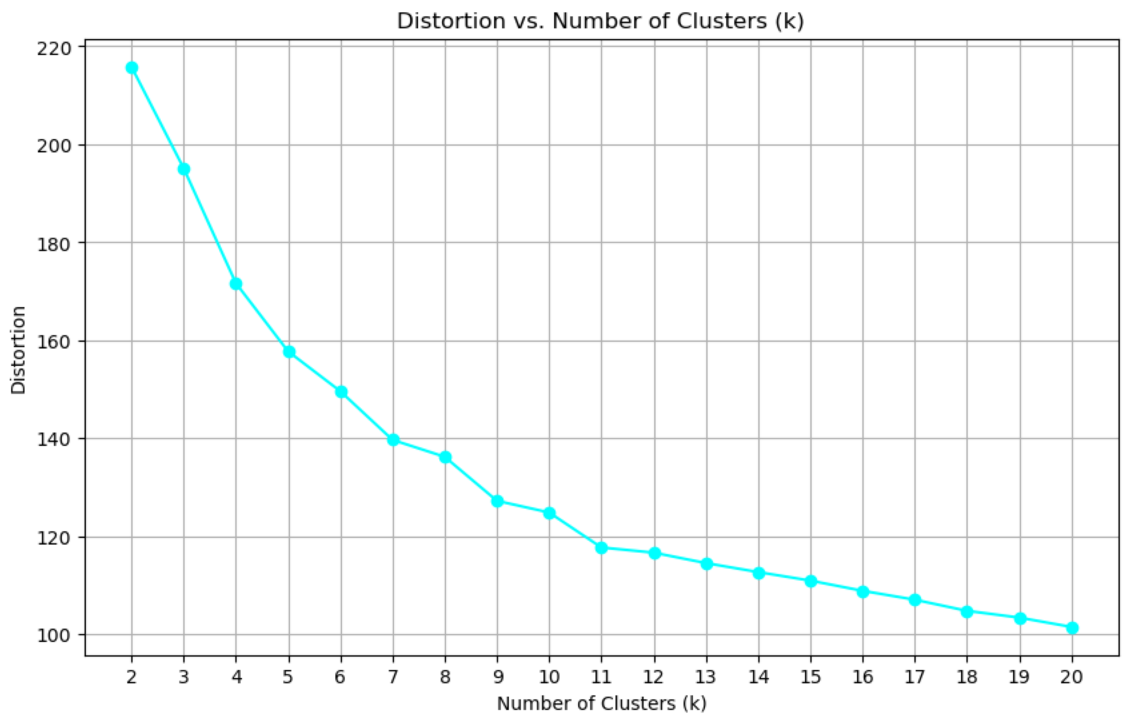
3 Python Programming Questions

3.1 Programming 1.4

Plots:



3.2 Programming 1.5



3.3 Programming 1.6

Why do you get different results with and without feature scaling?
Should you scale the features before fitting k-means? Why or why not?

Answer

1. Without scaling, features with larger ranges dominate the clustering because k-means relies on distance calculations. This can cause the clustering to be biased toward these features. When features are scaled, they contribute equally to the distance calculations, leading to more balanced and accurate clusters.
2. Yes, you should scale the features before k-means. Scaling ensures all features contribute equally to the clustering, preventing those with larger ranges from having too much influence. This results in more meaningful clusters that reflect patterns in the data accurately.

3.4 Programming 2.1

Accuracy vs no. of PC plot vs Baseline Plot

`PCA_Accuracy_vs_NoPC_Plot.png`

Explain the reason behind the trend:
1. **Increased Accuracy before No. of PC=4:**
Initial PCs Captures primary variance in data, they removed the noise and redunant information, but since when No. of PC is too small, they are not enough to capture all relevant features, hence causing Accuracy less than baseline. With the number of PC growing, more features are included to raise the accuracy of prediction.
2. **Peak Performance at No. of PC=4** Optimal balance achieved whe4e the trade off from information vs noises reached the optimal balance, i.e., we maintains the best signal-to-noise ratio at No. of PC=4 2. **Decline and Plateatu** As we add more PCs beyond the optimal point:
- components that capture less significant variance
- These components often represent noise rather than signal
- The model starts learning from this noise (overfitting)
- Additional components don't add meaningful information

Conclusion you could draw:
This trend demonstrates the fundamental trade-off in dimensionality reduction: we need enough components to capture essential patterns but not so many that we introduce noise. The optimal number of principal components occurs at the peak, where the model achieves the best balance between information retention and noise reduction. This point represents the most efficient dimensionality for the specific dataset and modeling task.

3.5 Programming 2.22

top three features that are explained by **PCs**

PC1:
worst area: 0.852063391798146
mean area: 0.5168264687224582
area error: 0.05572716691107043
PC2:
mean area: 0.8518237204834195
worst area: -0.5197423583172234
mean perimeter: 0.0627480827489297
PC3:
area error: 0.99024587828333069
worst perimeter: -0.09231337908602152
mean perimeter: -0.07166948144445899
PC4:
worst perimeter: 0.6668164509717286
worst texture: 0.5420574121750545
mean texture: 0.3624151106592851

3.6 Programming 2.3