references.bib

Applied Machine Learning
Homework 11

Anand Kamble
Department of Computer Science
Florida State University

Principal Component Analysis (PCA) on Horse Images

In this report, we perform Principal Component Analysis (PCA) on a set of horse images and a single bird image. The PCA is performed using Singular Value Decomposition (SVD) of the centered horse images. We explore various components of PCA such as discarding certain principal components, projecting data onto principal components, reconstructing images using different numbers of principal components, and analyzing distances in PCA-transformed spaces.

Approach and Implementation

Data Loading and Normalization

We load a total of 327 horse images and a single bird image, converting them to grayscale and flattening them into vectors. Pixel values are normalized to the range [0,1]:

[fontsize=,frame=single]python import numpy as np import matplotlib.pyplot as plt from PIL import Image import glob

$horse_images_path = sorted(glob.glob('horses/*.png')) horses = [] for img_path in horse_images_path : img = Image.open(img_path).convert('L') img_array = np.asarray(img, dtype = float)/255.0 Normalize pixel value to [0, 1] horses.append($

$np.array(horses) shape : (327, 128 * 128)$

$bird_image_path =' bird.png' bird_img = Image.open(bird_image_path).convert('L') bird_array = np.asarray(bird_img, dtype = float)/255.0 bird_vector = bird_array.flatten() Flatten bird image$

Computing Mean and Centering Horse Data

We compute the mean of the horse images and center them by subtracting this mean from each image vector. This step is crucial for PCA:

[fontsize=,frame=single]python $mean_horse = np.mean(horses, axis = 0)$

$horses_centered = horses - mean_horse$

Performing PCA using SVD

Principal Component Analysis is performed by computing the Singular Value Decomposition (SVD) of the centered horse data. The principal components (PCs) are derived from the singular vectors:

[fontsize=,frame=single]python $U, S, VT = np.linalg.svd(horses_centered, full_matrices = False) V = VT.T Principal components$

a) Discarding the Two Largest Singular Values

We visualize the singular values after discarding the two largest ones, to focus on the smaller and subsequent components:

[fontsize=,frame=single]python $remaining_singular_values = S[2 :] discard the first two largest singular values plt.figure(figsize =$
$(6, 4)) plt.plot(remaining_singular_values, marker =' o') plt.title('Remaining Singular Values (Sorted in Decreasing Order)') plt.xlab$

b) Projecting the Horses onto the First Two Principal Components

We project the centered horse images onto the first two principal components (PC1 and PC2) and plot them to visualize the data distribution in the new 2D feature space:

[fontsize=,frame=single]python $pcs = V[:, :2]$ first two principal components $horse_projections = horses_centered.dot(pcs)$

plt.figure(figsize=(6,4)) plt.scatter($horse_projections[:, 0], horse_projections[:, 1], c =' black', label =' Horses') plt.title('Projectio$
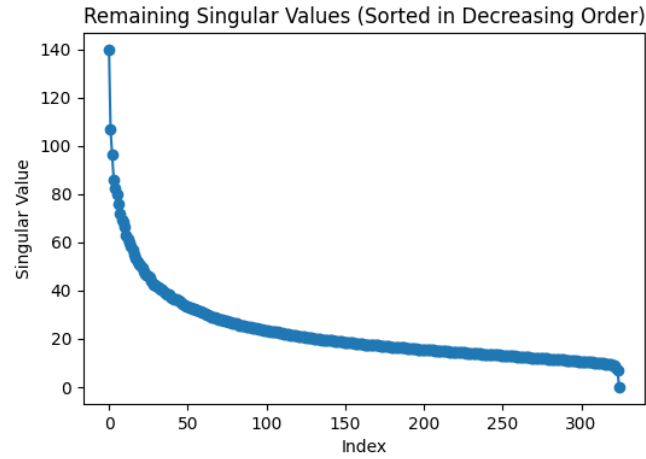
The output is shown in 2

Figure 1: Remaining Singular Values After Discarding the Two Largest
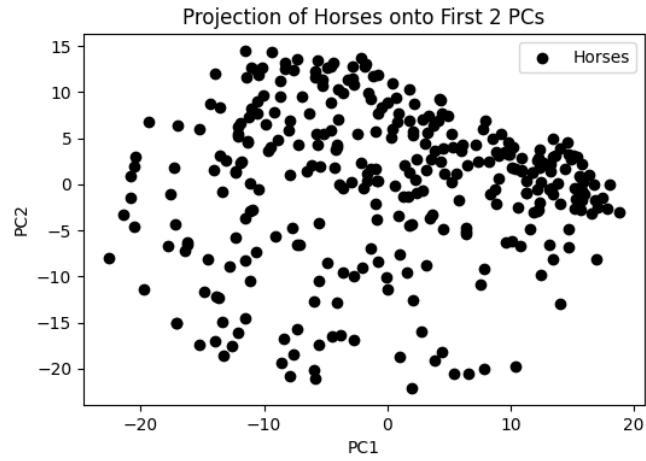

Figure 2: Horse Images Projected onto the First Two Principal Components

c) Plotting Bird Projection Alongside Horse Projections

We project the bird image onto the same two principal components (PC1 and PC2) derived from the horse images and plot it alongside the horse projections:

[fontsize=,frame=single]pythonprojection in red 'x' $bird_centered = bird_vector - mean_horse bird_projection = bird_centered.dot(pcs)$

plt.figure(figsize=(6,4)) plt.scatter(horse$_p$rojections$[:, 0]$, horse$_p$rojections$[:, 1]$, $c =' black', label =' Horses')plt.scatter(bird_pro$ ... $red', marker =' x', s = 100, label =' Bird')plt.title('HorseProjectionsandBirdProjectiononFirst2PCs')plt.xlabel('PC1')plt.ylab$ ...
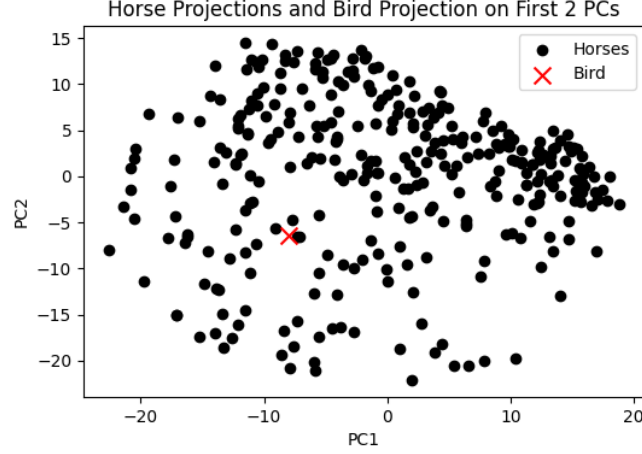
Figure 3: Horse and Bird Image Projections on the First Two Principal Components

d) Reconstructing a Horse Image Using 32 Principal Components

We reconstruct one of the horse images, `horse070.png`, using the first 32 principal components. The reconstruction is then thresholded at 0.5 to create a binary image:

[fontsize=,frame=single]python $horse070_img = Image.open('horses/horse070.png').convert('L')horse070_{array} = np.asarray(horse070_img, dtype = float)/255.0horse070_vector = horse070_{array}.flatten()$

Center the horse070 image $horse070_centered = horse070_vector - mean_horse$

Reconstruct using first 32 PCs $num_pcs = 32 pcs_3 2 = V[:, : num_pcs]scores_3 2 = horse070_centered.dot(pcs_3 2)horse070_reconstruct$ $mean_horse + scores_3 2.dot(pcs_3 2.T)horse070_reconstructed_binary = (horse070_reconstructed > 0.5).astype(float)$

Display the original and reconstructed binary images fig, axs = plt.subplots(1, 2, figsize=(8,4)) axs[0].imshow($horse070_{array}.resh$ $gray')axs[0].set_title('OriginalHorse070')axs[0].axis('off')axs[1].imshow(horse070_reconstructed_binary.reshape(128, 128), cmap$ $gray')axs[1].set_title('BinaryReconstruction(32PCs)')axs[1].axis('off')plt.show()$
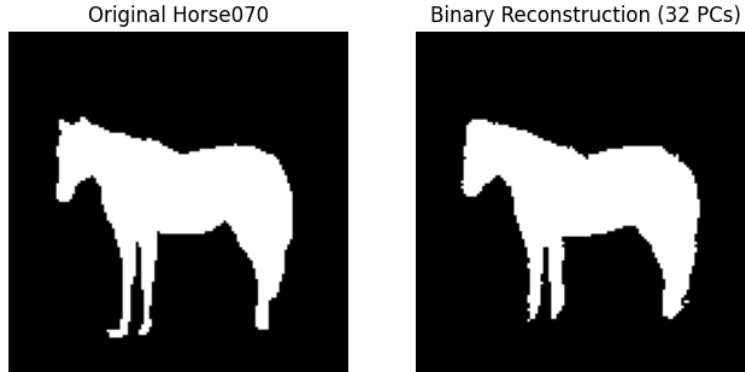

Figure 4: Original Horse070 (left) and Binary Reconstruction Using 32 PCs (right)

e) Reconstructing the Bird Image Using 32 Principal Components

We also reconstruct the bird image using the horse PCA model and the first 32 principal components. The reconstructed image is thresholded at 0.5 to create a binary image:

[fontsize=,frame=single]python $bird_reconstructed = mean_horse + (bird_centered.dot(pcs_3 2)).dot(pcs_3 2.T)bird_reconstructed_{bin}$ $(bird_reconstructed > 0.5).astype(float)$

Display the original and reconstructed binary bird images fig, axs = plt.subplots(1, 2, figsize=(8,4)) axs[0].imshow($bird_{array}.resh$ $gray')axs[0].set_title('OriginalBird')axs[0].axis('off')axs[1].imshow(bird_reconstructed_binary.reshape(128, 128), cmap =' gray')axs[1].set_title('BinaryReconstruction(32PCs)')axs[1].axis('off')plt.show()$

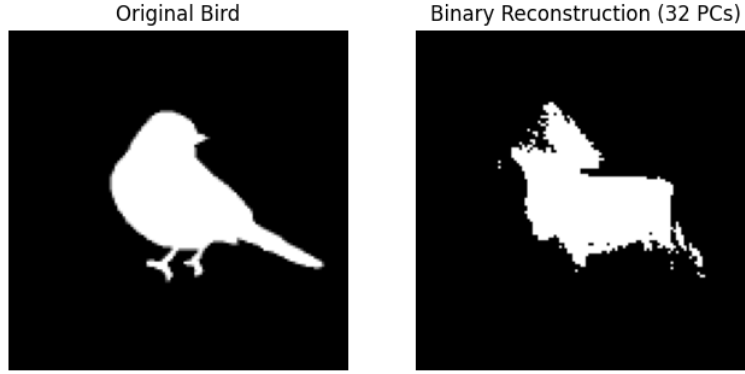Original Bird   Binary Reconstruction (32 PCs)

Figure 5: Original Bird (left) and Binary Reconstruction Using 32 PCs (right)

f) Distances to the 32-PC Plane

We compute the distances of both the horse images and the bird image to the plane generated by the 32 largest principal components. We then plot these distances on the y-axis against the second principal component projection on the x-axis:

[fontsize=, frame=single,breaklines]python reconstructing each image using 32 PCs. $horse_r econstructions_3 2 = mean_h orse + (horses_c entered.dot(pcs_3 2)).dot(pcs_3 2.T) distances_h orses = np.linalg.norm(horses_c entered - (horse_r econstructions_3 2 - mean_h orse), axis = 1) distance_b ird = np.linalg.norm(bird_c entered - (bird_r econstructed - mean_h orse))$

On the same graph, plot the computed distances vs the coordinates of the projections on the second PC plt.figure(figsize=(6,4)) plt.scatter(horse$_p rojections[:, 1], distances_h orses, c =' black', label =' Horses')$plt.scatter(bird$_p rojection[1$ $red', marker =' x', s = 100, label =' Bird')$plt.title($'Distances vs. Projection on 2nd PC'$)plt.xlabel($'Projection on PC2'$)plt.ylabel($'1$ $PC Plane'$)plt.legend()plt.show()
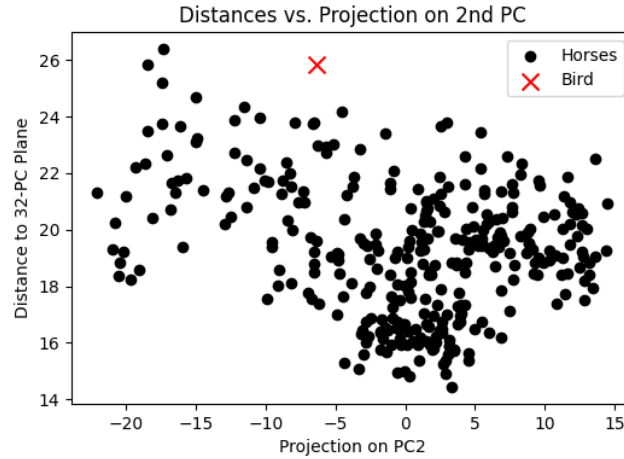

Figure 6: Distances to the 32-PC Plane vs. Projection on PC2 for Horses (black) and Bird (red)

g) Histogram of Distances for Horses

We plot a histogram of the distances of the horse images to the plane generated by the 32 largest PCs to understand the distribution of these distances:

[fontsize=,frame=single,breaklines]python Plot the histogram of the distances obtained at f) for the horses plt.figure(figsize=(6,4)) plt.hist($distances_horses, bins = 20, color =' black'$)$plt.title('Histogram of Distances to 32-PC Plane for Horses')plt.xlabel('Distance')plt.ylabel('Frequency')plt.show()$
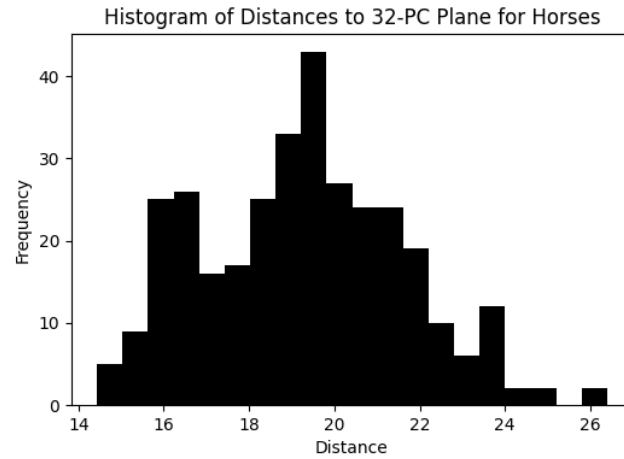

Figure 7: Histogram of Distances to the 32-PC Plane for Horses

Appendix: Code

[fontsize=, breaklines, frame=single]python import numpy as np import matplotlib.pyplot as plt from PIL import Image import glob import os

1. Load the horse images and bird image Assuming 'horses' directory contains horse images named 'horse001.png' to 'horse327.png' $horse_i mages_p ath : list[str] = sorted(glob.glob('horses/ * .png'))bird_i mage_p ath =' bird.png'$

horses: list[np.ndarray] — np.ndarray = [] for $img_p athinhorse_i mages_p ath : img : Image.Image = Image.open(img_p ath).conver$ $np.asarray(img, dtype = float)/255.0horses.append(img_a rray.flatten())horses = np.array(horses)shape :$ $(327, 128 * 128)$

$bird_i mg : Image.Image = Image.open(bird_i mage_p ath).convert('L')bird_a rray = np.asarray(bird_i mg, dtype = float)/255.0bird_v ector : np.ndarray = bird_a rray.flatten()Flattenbirdimage$

2. Compute the mean of the horse dataset $mean_h orse : np.ndarray = np.mean(horses, axis = 0)$

3. Center the horse data by subtracting the mean $horses_c entered = horses - mean_h orse$

4. Perform SVD on the horse images (PCA) U, S, VT = np.linalg.svd($horses_c entered, full_m atrices = False)V = VT.TPrincipalcomponents(eigenvectors)$

a) Discard the two largest singular values and plot the remaining singular values discard the first two largest singular values $remaining_s ingular_v alues : np.ndarray = S[2 :]plt.figure(figsize = (6, 4))plt.plot(remaining_s ingular_v alues, marke$ $o')plt.title('RemainingSingularV alues(SortedinDecreasingOrder)')plt.xlabel('Index')plt.ylabel('SingularV alue')plt.savefig('r$

b) Project the horses onto the first two principal components and plot pcs:np.ndarray = V[:, :2] first two principal components $horse_p rojections : np.ndarray = horses_c entered.dot(pcs)shape : (327, 2)$

plt.figure(figsize=(6, 4)) plt.scatter($horse_p rojections[:, 0], horse_p rojections[:, 1], c =' black', label =' Horses')plt.title('Projecti$

c) On the same graph, plot the horse projections in black and bird projection in red 'x' $bird_c entered = bird_v ector - mean_h orsebird_p rojection : np.ndarray = bird_c entered.dot(pcs)$

plt.figure(figsize=(6, 4)) plt.scatter($horse_p rojections[:, 0], horse_p rojections[:, 1], c =' black', label =' Horses')plt.scatter(bird_p r$ $red', marker =' x', s = 100, label =' Bird')plt.title('HorseProjectionsandBirdProjectiononFirst2PCs')plt.xlabel('PC1')plt.ylab$

d) Using the model, display the image 'horse070.png' and its binary reconstruction using 32 PCs (threshold 0.5) Find the horse070 image vector $horse070_i mg : Image.Image = Image.open('horses/horse070.png').convert('L')horse070_a rra$ $np.asarray(horse070_i mg, dtype = float)/255.0horse070_v ector : np.ndarray = horse070_a rray.flatten()$

Center the horse070 image $horse070_c entered = horse070_v ector - mean_h orse$

Reconstruct using first 32 PCs $num_p cs = 32pcs_3 2 : np.ndarray = V[:, : num_p cs]scores_3 2 : np.ndarray =$ $horse070_c entered.dot(pcs_3 2)horse070_r econstructed : np.ndarray = mean_h orse+scores_3 2.dot(pcs_3 2.T)horse070_r econstructed_b ina$ $np.ndarray = (horse070_r econstructed > 0.5).astype(float)$

Display the original and reconstructed binary images fig, axs = plt.subplots(1, 2, figsize=(8, 4)) axs[0].imshow($horse070_a rray.res$ $gray')axs[0].set_t itle('OriginalHorse070')axs[0].axis('off')axs[1].imshow(horse070_r econstructed_b inary.reshape(128, 128), cmap ='$ $gray')axs[1].set_t itle('BinaryReconstruction(32PCs)')axs[1].axis('off')plt.savefig('horse070_r econstruction.png')Savefigurepl$

e) Using the same model, display the bird image and its binary reconstruction using 32 PCs $bird_r econstructed :$ $np.ndarray = mean_h orse + (bird_c entered.dot(pcs_3 2)).dot(pcs_3 2.T)bird_r econstructed_b inary : np.ndarray =$ $(bird_r econstructed > 0.5).astype(float)$

Display the original and reconstructed binary bird images fig, axs = plt.subplots(1, 2, figsize=(8, 4)) axs[0].imshow($bird_a rray.res$ $gray')axs[0].set_t itle('OriginalBird')axs[0].axis('off')axs[1].imshow(bird_r econstructed_b inary.reshape(128, 128), cmap ='$ $gray')axs[1].set_t itle('BinaryReconstruction(32PCs)')axs[1].axis('off')plt.savefig('bird_r econstruction.png')Savefigureplt.shou$

f) Compute the distances of the horses and the bird to the plane generated by the 32 largest PCs. The plane generated by the 32 largest PCs can be approximated by reconstructing each image using 32 PCs. $horse_r econstructions_3 2 = mean_h orse+ (horses_c entered.dot(pcs_3 2)).dot(pcs_3 2.T)distances_h orses = np.linalg.norm(horses_c enter$ $(horse_r econstructions_3 2-mean_h orse), axis = 1)distance_b ird = np.linalg.norm(bird_c entered-(bird_r econstructed- $ $mean_h orse))$

On the same graph, plot the computed distances vs the coordinates of the projections on the second PC plt.figure(figsize=(6, 4)) plt.scatter($horse_p rojections[:, 1], distances_h orses, c =' black', label =' Horses')plt.scatter(bird_p rojection[$ $red', marker =' x', s = 100, label =' Bird')plt.title('Distancesvs.Projectionon2ndPC')plt.xlabel('ProjectiononPC2')plt.ylabel('$ $PCPlane')plt.legend()plt.savefig('distances_v s_p c2.png')Saveplotplt.show()$

g) Plot the histogram of the distances obtained at f) for the horses plt.figure(figsize=(6, 4)) plt.hist($distances_h orses, bins =$ $20, color =' black')plt.title('HistogramofDistancesto32-PCPlaneforHorses')plt.xlabel('Distance')plt.ylabel('Frequency')plt.s$