CS189

1. (a) By myself. consulting piazza.

(b) I certify that all solutions are entirely in my words and that I have not looked at another student's solutions. I have credited all external sources in this write up.

Yao Cao

HW5 Yao Cai

2. (a) $y + \epsilon_y = (X + \epsilon_x)W$

means that $y + \epsilon_y$ is in the column space of $X + \epsilon_X$, so adding $y + \epsilon_y$ as another column to $X + \epsilon_x$ won't increase rank

$\text{rank}([X + \epsilon_x, y + \epsilon_y]) = \text{rank}([X + \epsilon_x]) = d$

(b) $[\epsilon_x, \epsilon_y] = [X^{true}, y^{true}] - [X, y]$

$$= \begin{bmatrix} U_{xx} & U_{xy} \\ U_{yx}^T & U_{yy} \end{bmatrix} \left(\begin{bmatrix} \Sigma_d & \\ & 0 \end{bmatrix} - \begin{bmatrix} \Sigma_d & \\ & \sigma_{d+1} \end{bmatrix} \right) \begin{bmatrix} V_{xx} & V_{xy} \\ V_{yx}^T & V_{yy} \end{bmatrix}^T$$

$$= \begin{bmatrix} U_{xx} & U_{xy} \\ U_{yx}^T & U_{yy} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & -\sigma_{d+1} \end{bmatrix} \begin{bmatrix} V_{xx} & V_{xy} \\ V_{yx}^T & V_{yy} \end{bmatrix}^T$$

$$= \begin{bmatrix} 0 & -U_{xy}\sigma_{d+1} \\ 0 & -U_{yy}\sigma_{d+1} \end{bmatrix} \begin{bmatrix} V_{xx}^T & V_{yx} \\ V_{xy}^T & V_{yy}^T \end{bmatrix}$$

$$= -\begin{bmatrix} U_{xy}\sigma_{d+1}V_{xy}^T & U_{xy}\sigma_{d+1}V_{yy}^T \\ U_{yy}\sigma_{d+1}V_{xy}^T & U_{yy}\sigma_{d+1}V_{yy}^T \end{bmatrix}$$

$$= -\begin{bmatrix} U_{xy} \\ U_{yy} \end{bmatrix} \sigma_{d+1} \begin{bmatrix} V_{xy} \\ V_{yy} \end{bmatrix}^T$$

(c) We know that $\begin{bmatrix} w \\ -1 \end{bmatrix}$ is in the null space of $[X + \epsilon_X,\ y + \epsilon_y]$

$$\text{null}([X + \epsilon_x,\ y + \epsilon_y]) = \text{null}\left(U \begin{bmatrix} \Sigma_d & \\ & 0 \end{bmatrix} V^T\right)$$

$$= \text{span}\left\{ \begin{bmatrix} V_{xy} \\ V_{yy} \end{bmatrix} \right\}$$

since $\begin{bmatrix} V_{xy} \\ V_{yy} \end{bmatrix} \perp \begin{bmatrix} V_{xx} \\ V_{yx}^T \end{bmatrix}$

so we only need to find $\begin{bmatrix} V_{xy} \\ V_{yy} \end{bmatrix}$ and then

$$w = \frac{V_{xy}}{-V_{yy}}$$

(d) $\begin{bmatrix} w \\ -1 \end{bmatrix}$ is a right singular vector of $[X\ y]$ and

thus is an eigenvector of $[X\ y]^T [X\ y]$ with eigenvalue σ_{d+1}^2

$$[X\ \ y]^T [X\ \ y] = \begin{bmatrix} X^TX & X^Ty \\ y^TX & y^Ty \end{bmatrix}$$

$$\begin{bmatrix} X^TX & X^Ty \\ y^TX & y^Ty \end{bmatrix} \begin{bmatrix} w \\ -1 \end{bmatrix} = \sigma_{d+1}^2 \begin{bmatrix} w \\ -1 \end{bmatrix}$$

$$\Rightarrow \quad X^TXw - X^Ty = \sigma_{d+1}^2 w$$

$$\Rightarrow \quad (X^TX - \sigma_{d+1}^2 I)w = X^Ty$$

2e OLS

```
Coefficients:
[[ 202.31845431  162.41956802  149.07075034]
 [ -27.66555164  -17.88905339  -12.92356688]
 [  -5.15203925   -4.51375871   -4.24262639]
 [  -1.08629293    0.42947012    1.15475569]
 [  -3.14053107   -3.70269907   -3.74382934]
 [  23.67671768   23.15698002   21.94638397]
 [  -3.82167171    0.57606634    1.81637483]
 [   4.7346737     1.4677692    -1.12253649]
 [  -9.72739616   -5.75691108   -4.8395598 ]]
```

2f TLS

```
Coefficients:
[[  2.15827324e+02   1.73455256e+02   1.59223403e+02]
 [ -3.35499677e+01  -2.12337236e+01  -1.49769554e+01]
 [ -5.97766487e+00  -5.00991002e+00  -4.65713383e+00]
 [ -1.52907691e+00   1.24334039e-01   9.26150410e-01]
 [ -5.72765071e+01  -1.11647137e+02  -1.35024636e+02]
 [  4.53381943e+02   7.69484676e+02   8.97691622e+02]
 [ -4.24471050e+00   4.19737129e-01   1.71938827e+00]
 [  1.15793863e+02   1.29978076e+02   7.71272589e+01]
 [ -1.27507558e+01  -7.63093253e+00  -6.44815495e+00]]
```

2g TLS

We can scale f(n) by 255*1.5/max(f(n))

```
Coefficients:
[[ 1.10551685  0.88982383  0.81724741]
 [-0.16590439 -0.1081011  -0.07821557]
 [-0.03076651 -0.02717456 -0.02562091]
 [-0.00644545  0.0026217   0.00701448]
 [-0.04642399 -0.06119454 -0.06377764]
 [ 0.35231871  0.38605603  0.3774685 ]
 [-0.02110705  0.00317497  0.01005183]
 [ 0.07254515  0.02736135 -0.01626912]
 [-0.06066721 -0.03632074 -0.03062851]]
```

```
def computeBasis(ns):
    # Returns the first 9 spherical harmonic basis functions

    B = np.ones((len(ns),9)) # This line is here just to fill space
    x = ns[:, 0]
    y = ns[:, 1]
    z = ns[:, 2]

    B[:, 0] = 1
    B[:, 1] = y
    B[:, 2] = x
    B[:, 3] = z
    B[:, 4] = x*y
    B[:, 5] = y*z
    B[:, 6] = 3*z*z-1
    B[:, 7] = x*z
    B[:, 8] = x**2 - y**2
    return B/np.max(B)*255*1.5

if __name__ == '__main__':

    data,tennis,target = loadImages()
    ns, vs = extractNormals(data)
    B = computeBasis(ns)

    # reduce the number of samples because computing the SVD on
    # the entire data set takes too long
    Bp = B[::50]
    vsp = vs[::50]

    ##################################################
    # TODO: Solve for the coefficients using least squares
    # or total least squares here
    ##################################################
    coeff = np.zeros((9,3))
```

```
##OLS
#coeff = inv(Bp.T @ Bp) @ Bp.T @ vsp

##TLS
for i in range(3):
    n = vsp.shape[0]
    matrix = np.hstack((Bp, vsp[:, i].reshape((n, 1))))
    u, s, vh = svd(matrix, full_matrices = False)
    coeff[:, i] = -vh.T[:-1, -1]/vh.T[-1, -1]


img = relightSphere(tennis,coeff)

output = compositeImages(img,target)

print('Coefficients:\n'+str(coeff))

plt.figure(1)
plt.imshow(output)
plt.show()

imsave('TLS_2_output.png',output)
```

3. (a)

$$X = U\Sigma V^T$$

$$XX^T = U\Sigma V^T V \Sigma U^T$$

$$= U\Sigma^2 U^T$$

$$= [u_1 \cdots u_d]\,\Sigma^2 \begin{bmatrix} - u_1^T - \\ \vdots \\ - u_d^T - \end{bmatrix}$$

$$= \sum_{m=1}^{d} \sigma_m^2 u_m u_m^T$$

$$XX^T = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} [x_1 \cdots x_n] \qquad (XX^T)_{ij} = x_i^T x_j$$

$$= \begin{bmatrix} x_1^T x_1 & x_1^T x_2 & \cdots & x_1^T x_n \\ \vdots & & & \\ x_n^T x_1 & \cdots & & x_n^T x_n \end{bmatrix}$$

$$X^T X = V\Sigma U^T U \Sigma V^T$$

$$= V\Sigma^2 V^T$$

$$= [v_1 \cdots v_d]\,\Sigma^2 \begin{bmatrix} - v_1^T - \\ \vdots \\ - v_d^T - \end{bmatrix}$$

$$= \sum_{m=1}^{d} \sigma_m^2 v_m v_m^T$$

$$X^T X = [x_1 \cdots x_n] \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

$$= \sum_{m=1}^{n} x_m x_m^T \qquad x_m = \begin{pmatrix} x_{m1} \\ x_{m2} \\ \vdots \\ x_{md} \end{pmatrix}$$

$$= \sum_{m=1}^{n} \begin{pmatrix} x_{m1} \\ \vdots \\ x_{md} \end{pmatrix} (x_{m1} \cdots x_{md})$$

$$= \sum_{m=1}^{n} \begin{pmatrix} x_{m1}x_{m1} & x_{m1}x_{m2} & \cdots & x_{m1}x_{md} \\ \vdots & & & \\ x_{md}x_{m1} & \cdots & & x_{md}x_{md} \end{pmatrix}$$

$$(X^T X)_{ij} = \sum_{m=1}^{n} (x_{mi} x_{mj})$$

(b) $\psi_{PCA}(x_i)^T \psi_{PCA}(x_j)$

$$= (v_1^T x_i,\ v_2^T x_i,\ \cdots\ v_k^T x_i) \begin{pmatrix} x_j^T v_1 \\ x_j^T v_2 \\ \vdots \\ x_j^T v_k \end{pmatrix}$$

$$= (x_i^T v_1,\ x_i^T v_2,\ \cdots\ x_i^T v_k) \begin{pmatrix} v_1^T x_j \\ v_2^T x_j \\ \vdots \\ v_k^T x_j \end{pmatrix}$$

$$= x_i^T (v_1 \cdots v_k) \begin{pmatrix} v_1^T \\ \vdots \\ v_k^T \end{pmatrix} x_j$$

$$= x_i^T V_k V_k^T x_j$$

$$\left.\begin{aligned} V_k V_k^T &= [v_1 \cdots v_k] \begin{bmatrix} v_1^T \\ \vdots \\ v_k^T \end{bmatrix} = \sum_{i=1}^{k} v_i v_i^T \\ V I^k V^T &= [v_1 \cdots v_k \cdots v_d] \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & 0 & \\ & & & & \ddots \\ & & & & & 0 \end{bmatrix} \begin{bmatrix} v_1^T \\ \vdots \\ v_d^T \end{bmatrix} \\ &= [v_1 \cdots v_k\ 0 \cdots 0] \begin{bmatrix} v_1^T \\ \vdots \\ v_d^T \end{bmatrix} = \sum_{i=1}^{k} v_i v_i^T \end{aligned}\right\} \Rightarrow V_k V_k^T = V I^k V^T$$

(c) $\sum_{i=1}^{n}\sum_{j=1}^{n}(x_i^T x_j)^2 = \sum_{i=1}^{n}\sum_{j=1}^{n}((XX^T)_{ij})^2 = \|XX^T\|_F^2 = tr(XX^TXX^T)$

$$= tr(U\Sigma^2U^TU\Sigma^2U^T)$$
$$= tr(\Sigma^2\Sigma^2)$$
$$= tr(\Sigma^4) = \sum_{i=1}^{d}\sigma_i^4$$

$$(x_i^T x_j) - (\psi_{PCA}(x_i)^T \psi_{PCA}(x_j))$$

$$= x_i^T x_j - x_i^T V_K V_K^T x_j$$
$$= x_i^T I^d x_j - x_i^T V I^k V^T x_j$$
$$= x_i^T V I^d V^T x_j - x_i^T V I^k V^T x_j$$
$$= x_i^T V (I^d - I^k) V^T x_j$$
$$= x_i^T V I^{d-K} V^T x_j$$

$$I^{d-K} = I^d - I^k$$
$$= \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & 1 \end{pmatrix} - \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & 0 \\ & & & & & \ddots \\ & & & & & & 0 \end{pmatrix}$$
$$= \begin{pmatrix} 0 & & & & \\ & \ddots & & & \\ & & 0 & & \\ & & & 1 & \\ & & & & \ddots \\ & & & & & 1 \end{pmatrix} \begin{matrix} \left.\vphantom{\begin{matrix}0\\ \ddots\\ 0\end{matrix}}\right\} k \\ \left.\vphantom{\begin{matrix}1\\ \ddots\\ 1\end{matrix}}\right\} d-k \end{matrix}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\left|(x_i^T x_j) - (\psi_{PCA}(x_i)^T \psi_{PCA}(x_j))\right|^2$$
$$= \|XVI^{d-K}V^TX^T\|_F^2$$
$$= tr(XVI^{d-K}V^TX^TXVI^{d-K}V^TX^T)$$
$$= tr(VI^{d-K}V^TX^TXVI^{d-K}V^TX^TX)$$
$$= tr(VI^{d-K}V^TV\Sigma^2V^TVI^{d-K}V^TV\Sigma^2V^T)$$
$$= tr(I^{d-K}\Sigma^2 I^{d-K}\Sigma^2)$$
$$= tr(I^{d-K}\Sigma^4)$$
$$= \sum_{i=k+1}^{d}\sigma_i^4$$

$$\frac{\sum_{i=1}^{K}\sigma_i^2}{\sum_{i=1}^{d}\sigma_i^2} \geq 1-\varepsilon \Rightarrow \frac{\sum_{i=1}^{K}\sigma_i^2}{\sum_{i=K+1}^{d}\sigma_i^2} \geq \frac{1-\varepsilon}{\varepsilon}$$

$$\frac{\sum_{i=k+1}^{d}\sigma_i^4}{\sum_{i=1}^{d}\sigma_i^4} = \frac{\sum_{i=k+1}^{d}\sigma_i^4}{\sum_{i=1}^{K}\sigma_i^4 + \sum_{i=k+1}^{d}\sigma_i^4} = \frac{1}{\frac{\sum_{i=1}^{K}\sigma_i^4}{\sum_{i=k+1}^{d}\sigma_i^4} + 1} \leq \frac{1}{\frac{\sum_{i=1}^{K}\sigma_i^2 \cdot \sigma_K^2}{\sum_{i=K+1}^{d}\sigma_i^2 \cdot \sigma_{K+1}^2} + 1}$$

$$\leq \frac{1}{\frac{1-\varepsilon}{\varepsilon} + 1} = \varepsilon$$

(d) $x^T y = \dfrac{\|x\|^2 + \|y\|^2 - \|x-y\|^2}{2}$

$$\left| \psi(x_i)^T \psi(x_j) - x_i^T x_j \right|$$

$$= \left| \frac{\|\psi(x_i)\|^2 + \|\psi(x_j)\|^2 - \|\psi(x_i) - \psi(x_j)\|^2 - \|x_i\|^2 - \|x_j\|^2 + \|x_i - x_j\|^2}{2} \right|$$

$$\leq \frac{\left| \|\psi(x_i)\|^2 - \|x_i\|^2 \right|}{2} + \frac{\left| \|\psi(x_j)\|^2 - \|x_j\|^2 \right|}{2} + \frac{\left| \|\psi(x_i) - \psi(x_j)\|^2 - \|x_i - x_j\|^2 \right|}{2}$$

$$\leq \frac{\varepsilon \|x_i\|^2}{2} + \frac{\varepsilon \|x_j\|^2}{2} + \frac{\varepsilon}{2} \|x_i - x_j\|^2$$

$$= \varepsilon \left(\frac{\|x_i\|^2}{2} + \frac{\|x_j\|^2}{2} + \frac{\|x_i - x_j\|^2}{2} \right) \quad ①$$

we know $\|x_i\| \leq 1$

so $\|x_i\|^2, \|x_j\|^2 \leq 1$

$\|x_i - x_j\|^2 \leq 4$

① $\leq 3\varepsilon$

(e) $\|\psi_J(u)\|^2 = \|\frac{1}{\sqrt{k}} J u\|^2 = \frac{1}{k} u^T J^T J u$

$$= \frac{1}{k} u^T \begin{pmatrix} z_{11} & z_{21} & z_{31} & \cdots & z_{k1} \\ \vdots & & & & \\ z_{1d} & \cdots & & & z_{kd} \end{pmatrix} \begin{pmatrix} z_{11} & z_{12} & \cdots & z_{1d} \\ \vdots & & & \\ z_{k1} & \cdots & & z_{kd} \end{pmatrix} u$$

$$= \frac{1}{k} u^T \begin{pmatrix} \sum_{i=1}^k z_{i1}^2 & & & 0 \\ & \sum_{i=1}^k z_{i2}^2 & & \\ & & \ddots & \\ 0 & & & \sum_{i=1}^k z_{id}^2 \end{pmatrix} u = \frac{1}{k} \sum_{i=1}^k z_i^2 u^T I u = \frac{1}{k} \sum_{i=1}^k z_i^2 \|u\|^2$$

$$\Rightarrow \frac{\|\psi_J(u)\|^2}{\|u\|^2} = \frac{1}{k} \sum_{i=1}^k z_i^2$$

the zero terms $\Rightarrow z_{11} z_{12} + z_{21} z_{22} + z_{31} z_{32} + \cdots + z_{k1} z_{k2} = \sum_{i=1}^k z_{i1} z_{i2}$

$$\approx k E(z_{i1} z_{i2})$$

$$= k E(z_{i1}) E(z_{i2})$$

$$= 0$$

(f) $A_{ij} = \left\{ \frac{\|\psi_J(x_i) - \psi_J(x_j)\|^2}{\|x_i - x_j\|^2} \in (1-\varepsilon, 1+\varepsilon) \right\}$

$$= \left\{ \frac{\left\|\frac{1}{\sqrt{k}} J(x_i - x_j)\right\|^2}{\|x_i - x_j\|^2} \in (1-\varepsilon, 1+\varepsilon) \right\}$$

$$= \left\{ \frac{1}{k}\sum_{i=1}^{k} z_i^2 \in (1-\varepsilon, 1+\varepsilon) \right\}$$

$$P(A_{ij}^c) = P\left\{ \frac{1}{k}\sum_{i=1}^{k} z_i^2 \notin (1-\varepsilon, 1+\varepsilon) \right\} \leq 2e^{-k\varepsilon^2/8}$$

$$P[\cap_{i,j} A_{ij}] \geq 1 - \sum_{i,j} P[A_{ij}^c]$$

$$\geq 1 - \frac{N(N-1)}{2} \cdot 2e^{-k\varepsilon^2/8}$$

$$\geq 1 - N^2 e^{-k\varepsilon^2/8}$$

$$\geq 1 - N^2 e^{-\frac{16}{\varepsilon^2}\log\left(\frac{N}{\delta}\right)\varepsilon^2/8}$$

$$= 1 - N^2 e^{\log\frac{\delta}{N} \cdot 2}$$

$$= 1 - N^2\left(\frac{\delta}{N}\right)^2$$

$$= 1 - \delta^2 \geq 1 - \delta$$

(g) With high probability, for all $u \in S_1$, $v \in S_2$

$$1-\varepsilon < \frac{\|\psi_J(u) - \psi_J(v)\|^2}{\|u - v\|^2} < 1+\varepsilon \qquad \text{if } k \geq \frac{16}{\varepsilon^2}(\log(m+n) + C)$$

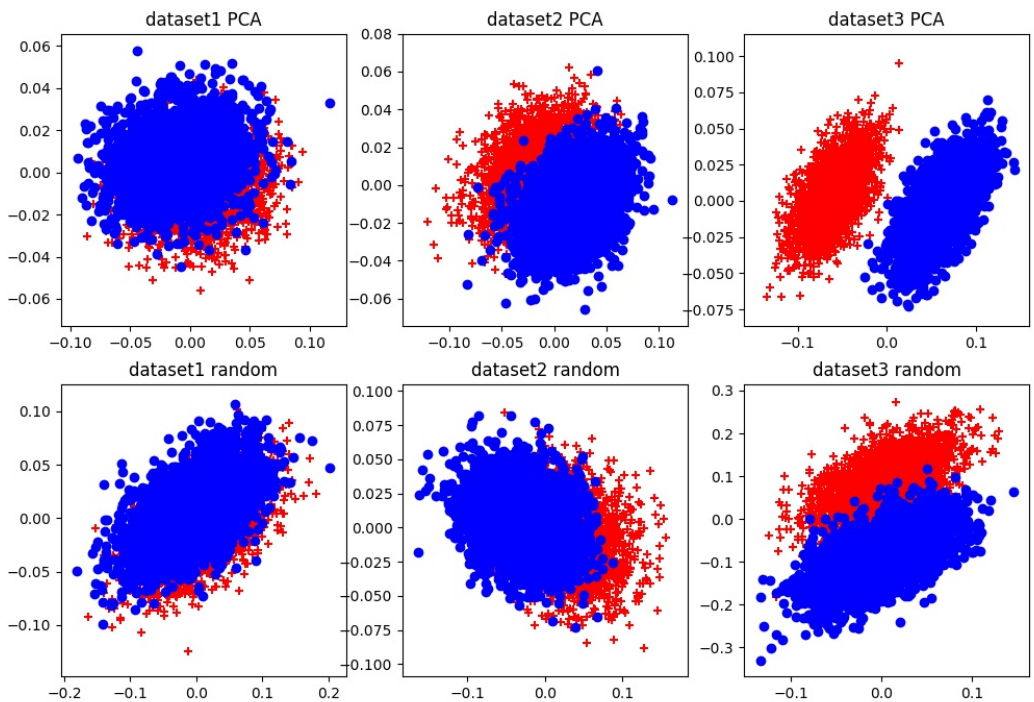
$$\Rightarrow \|\psi_J(u) - \psi_J(v)\|^2 > (1-\varepsilon)\|u-v\|^2 \geq (1-\varepsilon)\min\|u-v\|^2$$

$$\geq (1-\varepsilon)\gamma \qquad \text{for all } u, v$$

$$\Rightarrow \min\|\psi_J(u) - \psi_J(v)\|^2 \geq (1-\varepsilon)\gamma$$

if $k \geq \frac{c}{\varepsilon^2}\log(m+n)$

3h

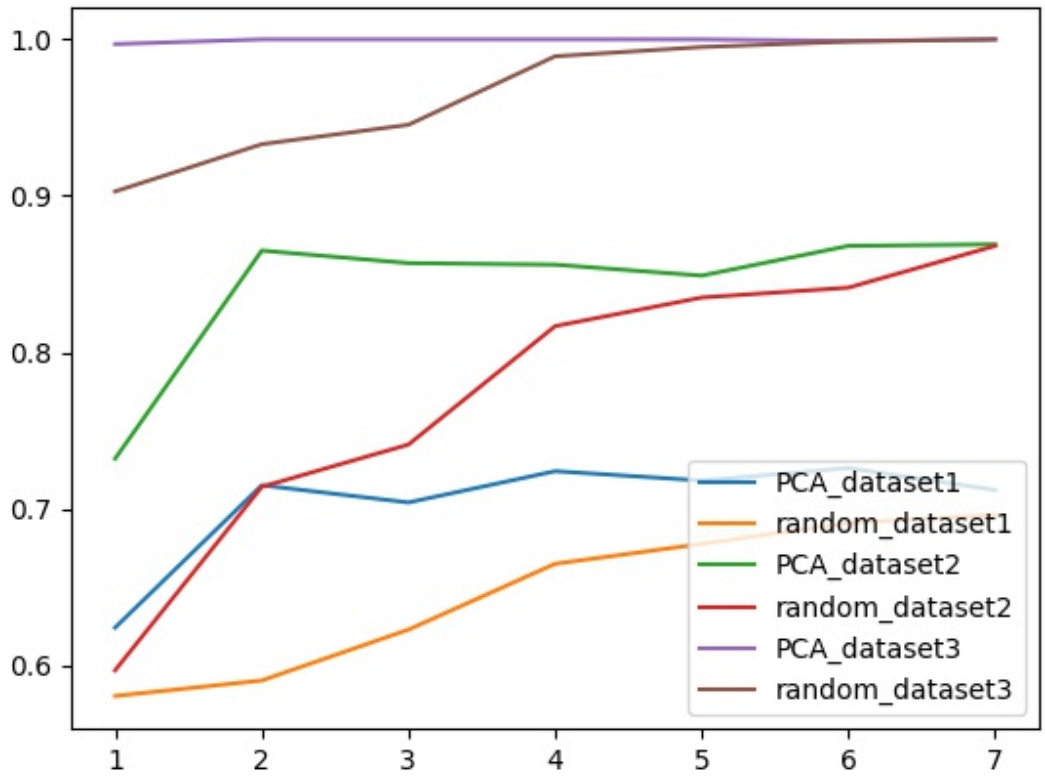


1\ For dataset 3, PCA can separate the two classes pretty well using only the first 2 principle components. Random project doesn't perform as well at low dimensions.

2\ From dataset1 to dataset2 to dataset3, it becomes easier to separate the two classes.

```
# Using PCA and Random Projection for:
# Visualizing the datasets

#fig, axs = plt.subplots(2, 3, figsize = (12, 8))
#for i in range(3):
#	data = np.load('data'+str(i+1)+'.npz')
#	X = data['X']
#	y = data['y']
#	n, d = X.shape

#	pos_index = y == +1
#	neg_index = y == -1

#	X_pca = pca_proj(X, 2)
#	axs[0, i].scatter(X_pca[pos_index, 0], X_pca[pos_index, 1], color = 'r',
marker = '+')
#	axs[0, i].scatter(X_pca[neg_index, 0], X_pca[neg_index, 1], color = 'b',
marker = 'o')
#	axs[0, i].set_title('dataset' + str(i+1) + ' PCA')

#	X_random = random_proj(X, 2)
#	axs[1, i].scatter(X_random[pos_index, 0], X_random[pos_index, 1], color =
'r', marker = '+')
#	axs[1, i].scatter(X_random[neg_index, 0], X_random[neg_index, 1], color =
'b', marker = 'o')
#	axs[1, i].set_title('dataset' + str(i+1) + ' random')

#plt.savefig('3h.jpg')
#plt.show()
```

3i



1\ random projection performs better and better with increasing k

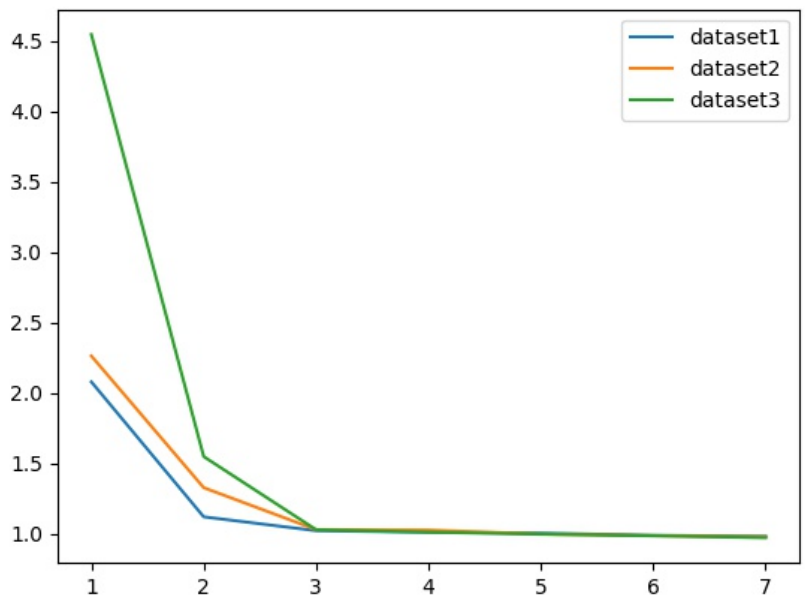
2\ In dataset1 and dataset2, PCA performance saturates at k = 2; in dataset3, PCA performance saturates at k=1.

3\ PCA always performs better than random projection, since PCA has access to the data, but random projection doesn't

4\ From dataset1 to dataset2 to dataset3, the classification accuracy increases.

```
# Computing the accuracies over different datasets.

#n_trials = 10

#for i in range(3):
#	data = np.load('data'+str(i+1)+'.npz')
#	X = data['X']
#	y = data['y']
#	n, d = X.shape

#	accuracies = np.zeros(d)
#	for k in range(1, d+1):
#		accuracies[k-1] = pca_proj_accuracy(X, y, k)
#	plt.plot(range(1, d+1), accuracies, label = 'PCA_' + 'dataset' + str(i+1))

#	accuracies = np.zeros((n_trials, d))
#	for j in range(n_trials):
#		for k in range(1, d+1):
#			accuracies[j, k-1] = rand_proj_accuracy_split(X, y, k)
#	plt.plot(range(1, d+1), np.average(accuracies, axis=0), label = 'random_' +
'dataset' + str(i+1))

#plt.legend()
#plt.savefig('3i.jpg')
#plt.show()
```

3j



For dataset3, the most informative part is in the first principle component, so we only need the first principle component to classify, and more components won't help much. For dataset 1 and 2, the most informative part is in the first two principle components, so we only need the first two principle components to classify, and more components won't help much.

```
# And computing the SVD of the feature matrix

for i in range(3):
        data = np.load('data'+str(i+1)+'.npz')
        X = data['X']
        y = data['y']
        n, d = X.shape

        s =  svd(X, compute_uv = False)
        plt.plot(range(1, d+1), s, label = 'dataset'+str(i+1))

plt.legend()
plt.savefig('3j.jpg')
plt.show()
```