CS 189 Introduction to Machine Learning
Fall 2018 HW4

This homework is due **Friday, February 16 at 10pm.**

2 Total Least Squares

In most of the models we have looked at so far, we've accounted for noise in the observed y measurement and adjusted accordingly. However, in the real world it could easily be that our feature matrix $\mathbf{X}$ of data is also corrupted or noisy. Total least squares is a way to account for this. Whereas previously we were minimizing the y distance from the data point to our predicted line because we had assumed the features were definitively accurate, now we are minimizing the entire distance from the data point to our predicted line. In this problem we will explore the mathematical intuition for the TLS formula.

Let $\mathbf{X}$ and $\mathbf{y}$ be the true measurements. Recall that in the least squares problem, we want to solve for $\mathbf{w}$ in $\min_{\mathbf{w}} ||\mathbf{X}\mathbf{w} - \mathbf{y}||$. We measure the error as the difference between $\mathbf{X}\mathbf{w}$ and $\mathbf{y}$, which can be viewed as adding an error term $\boldsymbol{\epsilon}_{\mathbf{y}}$ such that the equation $\mathbf{X}\mathbf{w} = \mathbf{y} + \boldsymbol{\epsilon}_{\mathbf{y}}$ has a solution:

$$\min_{\boldsymbol{\epsilon}_{\mathbf{y}}, \mathbf{w}} ||\boldsymbol{\epsilon}_{\mathbf{y}}||_2, \text{ subject to } \mathbf{X}\mathbf{w} = \mathbf{y} + \boldsymbol{\epsilon}_{\mathbf{y}} \tag{1}$$

Although this optimization formulation allows for errors in the measurements of $\mathbf{y}$, it does not allow for errors in the feature matrix $\mathbf{X}$ that is measured from the data. In this problem, we will explore a method called *total least squares* that allows for both error in the matrix $\mathbf{X}$ and the vector $\mathbf{y}$, represented by $\boldsymbol{\epsilon}_{\mathbf{X}}$ and $\boldsymbol{\epsilon}_{\mathbf{y}}$, respectively. For convenience, we absorb the negative sign into $\boldsymbol{\epsilon}_{\mathbf{y}}$ and $\boldsymbol{\epsilon}_{\mathbf{X}}$ and define true measurements $\mathbf{y}$ and $\mathbf{X}$ like so:

$$\mathbf{y}^{true} = \mathbf{y} + \boldsymbol{\epsilon}_{\mathbf{y}} \tag{2}$$
$$\mathbf{X}^{true} = \mathbf{X} + \boldsymbol{\epsilon}_{\mathbf{X}} \tag{3}$$

Specifically, the total least squares problem is to find the solution for $\mathbf{w}$ in the following minimization problem:

$$\min_{\boldsymbol{\epsilon}_{\mathbf{y}}, \boldsymbol{\epsilon}_{\mathbf{X}}, \mathbf{w}} ||[\boldsymbol{\epsilon}_{\mathbf{X}}, \boldsymbol{\epsilon}_{\mathbf{y}}]||_F^2, \text{ subject to } (\mathbf{X} + \boldsymbol{\epsilon}_{\mathbf{X}})\mathbf{w} = \mathbf{y} + \boldsymbol{\epsilon}_{\mathbf{y}} \tag{4}$$

where the matrix $[\boldsymbol{\epsilon}_{\mathbf{X}}, \boldsymbol{\epsilon}_{\mathbf{y}}]$ is the concatenation of the columns of $\boldsymbol{\epsilon}_{\mathbf{X}}$ with the column vector $\mathbf{y}$. Notice that the minimization is over $\mathbf{w}$ because it's a free parameter, and it does not necessarily

have to be in the objective function. Intuitively, this equation is finding the smallest perturbation to the matrix of data points $\mathbf{X}$ and the outputs $\mathbf{y}$ such that the linear model can be solved exactly. The constraint in the minimization problem can be rewritten as

$$[\mathbf{X} + \boldsymbol{\epsilon}_\mathbf{X}, \mathbf{y} + \boldsymbol{\epsilon}_\mathbf{y}] \begin{bmatrix} \mathbf{w} \\ -1 \end{bmatrix} = \mathbf{0} \tag{5}$$

(a) Let the matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ and $\mathbf{y} \in \mathbb{R}^n$ and note that $\boldsymbol{\epsilon}_\mathbf{X} \in \mathbb{R}^{n \times d}$ and $\boldsymbol{\epsilon}_\mathbf{y} \in \mathbb{R}^n$. **Assuming that $n > d$ and rank$(\mathbf{X} + \boldsymbol{\epsilon}_\mathbf{X}) = d$, explain why rank$([\mathbf{X} + \boldsymbol{\epsilon}_\mathbf{X}, \mathbf{y} + \boldsymbol{\epsilon}_\mathbf{y}]) = d$ for $\boldsymbol{\epsilon}_\mathbf{X}, \boldsymbol{\epsilon}_\mathbf{y}$ satisfying** (5)**.**

Solution:

Given that the constraint in the minimization problem from Equation (4) is satisfied, the last column of the matrix $[\mathbf{X}+\boldsymbol{\epsilon}_\mathbf{X}, \mathbf{y}+\boldsymbol{\epsilon}_\mathbf{y}]$. which is $\mathbf{y}+\boldsymbol{\epsilon}_\mathbf{y}$ can be expressed as a linear combination of the remaining columns in the matrix. Therefore, the rank of the matrix is the rank of the remaining columns, which we previously assumed was d.

(b) Before continuing, we establish some linear algebra background. Let $\mathbf{A} \in \mathbb{R}^{n \times m}$ be an arbitrary matrix. Recall that the *Frobenius norm* of $\mathbf{A}$ is defined as $\|\mathbf{A}\|_F := \sqrt{\sum_{i=1}^n \sum_{j=1}^d \mathbf{A}_{ij}^2}$, which can be thought of as the ℓ_2-norm of $\mathbf{A}$ viewed as one long vector in $\mathbb{R}^{nd}$. **Prove that $\|\mathbf{A}\|_F = \sqrt{\text{tr}\left(\mathbf{A}^\top \mathbf{A}\right)}$, and that if that $\mathbf{O} \in \mathbb{R}^{n \times n}$ and $\mathbf{P} \in \mathbb{R}^{m \times m}$ are orthogonal matrices, then**

$$\|\mathbf{OAP}\|_F = \|\mathbf{A}\|_F,$$

that is, the Frobenius norm is rotation invariant.

(c) Suppose that $n \geq m$, and $\mathbf{A} \in \mathbb{R}^{n \times m}$ has compact SVD

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top,$$

where $\mathbf{U} \in \mathbb{R}^{n \times m}$ and $\mathbf{V} \in \mathbb{R}^{m \times m}$ have orthonormal columns, and $\boldsymbol{\Sigma} \in \mathbb{R}^{m \times m}$ where $\boldsymbol{\Sigma}_{ii} = \sigma_i$ for $i \in \{1, \ldots, m\}$, and where $\sigma_1 \geq \sigma_2 \cdots \geq \sigma_m$ are the singular values of $\mathbf{A}$ arranged in descending order. **Show using previous parts of this problem that $\|\mathbf{A}\|_F^2 = \sum_{i=1}^d \sigma_i^2$.**

Moreover, recall that the rank-k SVD of $\mathbf{A}$ is defined as

$$\mathbf{A}_k = \mathbf{U} \begin{bmatrix} \boldsymbol{\Sigma}_{1:k} & \mathbf{0}_{k \times (m-k)} \\ \mathbf{0}_{(m-k) \times k} & \mathbf{0}_{(m-k) \times (m-k)} \end{bmatrix} \mathbf{V}^\top,$$

where $\boldsymbol{\Sigma}_{1:k} \in \mathbb{R}^{k \times k}$ has a $\sigma_1 \geq \cdots \geq \sigma_k$ of $\mathbf{A}$ arranged on the diagonal (we shall assume that $\sigma_k > \sigma_{k+1}$.) **Show, using first principles (i.e. no fancy theorems!) that that $\|\mathbf{A} - \mathbf{A}_k\|_F^2 = \sum_{i=k+1}^m \sigma_i^2$.**

One can show (you don't need to do this) that in fact, if $\mathbf{B}$ is any rank k matrix, then $\|\mathbf{A} - \mathbf{B}\|_F^2 \geq \sum_{i=k+1}^m \sigma_i^2$, which equality if and only if $\mathbf{B} = \mathbf{A}_k$. Thus, $\mathbf{A}_k$ is in fact the *best rank-k approximation to the matrix A*; this is known as the *Eckart-Minsky-Young theorem.*

(d) We will now leverage the low rank approximation to develop total least squares. For the solution $\mathbf{w}$ to be unique, the matrix $[\mathbf{X} + \boldsymbol{\epsilon}_\mathbf{X}, \mathbf{y} + \boldsymbol{\epsilon}_\mathbf{y}]$ must have exactly d linearly independent columns. Since this matrix has $d+1$ columns in total, it must be rank-deficient by 1. Recall that the Eckart-Young-Mirsky Theorem tells us that the closest lower-rank matrix in the Frobenius norm is obtained by discarding the smallest singular values. Therefore, the rank d matrix $[\mathbf{X} + \boldsymbol{\epsilon}_\mathbf{X}, \mathbf{y} + \boldsymbol{\epsilon}_\mathbf{y}]$ that minimizes

$$|||[\boldsymbol{\epsilon}_\mathbf{X}, \boldsymbol{\epsilon}_\mathbf{y}]|||_F^2 = |||[\mathbf{X}^{true}, \mathbf{y}^{true}] - [\mathbf{X}, \mathbf{y}]|||_F^2$$

is given by

$$[\mathbf{X} + \boldsymbol{\epsilon}_\mathbf{X}, \mathbf{y} + \boldsymbol{\epsilon}_\mathbf{y}] = \mathbf{U} \begin{bmatrix} \boldsymbol{\Sigma}_d & \\ & 0 \end{bmatrix} \mathbf{V}^\top$$

where $[\mathbf{X}, \mathbf{y}] = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$.

Suppose we express the SVD of $[\mathbf{X}, \mathbf{y}]$ in terms of submatrices and vectors:

$$[\mathbf{X}, \mathbf{y}] = \begin{bmatrix} \mathbf{U}_{xx} & \mathbf{u}_{xy} \\ \mathbf{u}_{yx}^\top & u_{yy} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma}_d & \\ & \sigma_{d+1} \end{bmatrix} \begin{bmatrix} \mathbf{V}_{xx} & \mathbf{v}_{xy} \\ \mathbf{v}_{yx}^\top & v_{yy} \end{bmatrix}^\top$$

$\mathbf{u}_{xy} \in \mathbb{R}^{n-1}$ is the first $(n-1)$ elements of the $(d+1)$-th column of $\mathbf{U}$, $\mathbf{u}_{yx}^\top \in \mathbb{R}^d$ is the first d elements of the n-th row of $\mathbf{U}$, u_{yy} is the n-th element of the $(d+1)$-th column of $\mathbf{U}$, $\mathbf{U}_{xx} \in \mathbb{R}^{(n-1)\times d}$ is the $(n-1) \times d$ top left submatrix of $\mathbf{U}$.

Similarly, $\mathbf{v}_{xy} \in \mathbb{R}^d$ is the first d elements of the $(d+1)$-th column of $\mathbf{V}$, $\mathbf{v}_{yx}^\top \in \mathbb{R}^d$ is the first d elements of the $(d+1)$-th row of $\mathbf{V}$, v_{yy} is the $(d+1)$-th element of the $(d+1)$-th column of $\mathbf{V}$, $\mathbf{V}_{xx} \in \mathbb{R}^{d\times d}$ is the $d \times d$ top left submatrix of $\mathbf{V}$. σ_{d+1} is the $(d+1)$-th eigenvalue of $[\mathbf{X}, \mathbf{y}]$. $\boldsymbol{\Sigma}_d$ is the diagonal matrix of the d largest singular values of $[\mathbf{X}, \mathbf{y}]$

Using this information show that

$$[\boldsymbol{\epsilon}_\mathbf{X}, \boldsymbol{\epsilon}_\mathbf{y}] = -\begin{bmatrix} \mathbf{u}_{xy} \\ u_{yy} \end{bmatrix} \sigma_{d+1} \begin{bmatrix} \mathbf{v}_{xy} \\ v_{yy} \end{bmatrix}^\top$$

Solution:

We have

$$[\mathbf{X}, \mathbf{y}] + [\boldsymbol{\epsilon}_\mathbf{X}, \boldsymbol{\epsilon}_\mathbf{y}] = \begin{bmatrix} \mathbf{U}_{xx} & \mathbf{u}_{xy} \\ \mathbf{u}_{yx}^\top & u_{yy} \end{bmatrix} \begin{bmatrix} \Sigma_d & \\ & 0 \end{bmatrix} \begin{bmatrix} \mathbf{V}_{xx} & \mathbf{v}_{xy} \\ \mathbf{v}_{yx}^\top & v_{yy} \end{bmatrix}^\top$$

We can subtract the second equation from the first to solve for $[\boldsymbol{\epsilon}_\mathbf{X}, \boldsymbol{\epsilon}_\mathbf{y}]$:

$$
\begin{aligned}
[\mathbf{X}, \mathbf{y}] - ([\mathbf{X}, \mathbf{y}] + [\boldsymbol{\epsilon}_\mathbf{X}, \boldsymbol{\epsilon}_\mathbf{y}]) &= \begin{bmatrix} \mathbf{U}_{xx} & \mathbf{u}_{xy} \\ \mathbf{u}_{yx}^\top & u_{yy} \end{bmatrix} \begin{bmatrix} \Sigma_d & \\ & \sigma_{d+1} \end{bmatrix} \begin{bmatrix} \mathbf{V}_{xx} & \mathbf{v}_{xy} \\ \mathbf{v}_{yx}^\top & v_{yy} \end{bmatrix}^\top \\
&\quad - \begin{bmatrix} \mathbf{U}_{xx} & \mathbf{u}_{xy} \\ \mathbf{u}_{yx}^\top & u_{yy} \end{bmatrix} \begin{bmatrix} \Sigma_d & \\ & 0 \end{bmatrix} \begin{bmatrix} \mathbf{V}_{xx} & \mathbf{v}_{xy} \\ \mathbf{v}_{yx}^\top & v_{yy} \end{bmatrix}^\top \\
-[\boldsymbol{\epsilon}_\mathbf{X}, \boldsymbol{\epsilon}_\mathbf{y}] &= \begin{bmatrix} \mathbf{U}_{xx} & \mathbf{u}_{xy} \\ \mathbf{u}_{yx}^\top & u_{yy} \end{bmatrix} \left(\begin{bmatrix} \Sigma_d & \\ & \sigma_{d+1} \end{bmatrix} - \begin{bmatrix} \Sigma_d & \\ & 0 \end{bmatrix} \right) \begin{bmatrix} \mathbf{V}_{xx} & \mathbf{v}_{xy} \\ \mathbf{v}_{yx}^\top & v_{yy} \end{bmatrix}^\top \\
[\boldsymbol{\epsilon}_\mathbf{X}, \boldsymbol{\epsilon}_\mathbf{y}] &= - \begin{bmatrix} \mathbf{U}_{xx} & \mathbf{u}_{xy} \\ \mathbf{u}_{yx}^\top & u_{yy} \end{bmatrix} \begin{bmatrix} 0 & \\ & \sigma_{d+1} \end{bmatrix} \begin{bmatrix} \mathbf{V}_{xx} & \mathbf{v}_{xy} \\ \mathbf{v}_{yx}^\top & v_{yy} \end{bmatrix}^\top \\
&= - \begin{bmatrix} \mathbf{u}_{xy} \\ u_{yy} \end{bmatrix} \sigma_{d+1} \begin{bmatrix} \mathbf{v}_{xy} \\ v_{yy} \end{bmatrix}^\top
\end{aligned}
$$

(e) **Using the result from the previous part and the fact that v_{yy} is not 0 (see notes on Total Least Squares), find a nonzero solution to $[\mathbf{X} + \boldsymbol{\epsilon}_\mathbf{X}, \mathbf{y} + \boldsymbol{\epsilon}_\mathbf{y}] \begin{bmatrix} \mathbf{w} \\ -1 \end{bmatrix} = 0$ and thus solve for $\mathbf{w}$ in Equation (5).**

HINT: Looking at the last column of the product $[\mathbf{X}, \mathbf{y}]\mathbf{V}$ may or may not be useful for this problem, depending on how you solve it.

Solution:

Following the hint, we look at the last column of the product $[\mathbf{X}, \mathbf{y}]\mathbf{V}$, which is given by

$$
[\mathbf{X}, \mathbf{y}] \begin{bmatrix} \mathbf{v}_{xy} \\ v_{yy} \end{bmatrix} = \begin{bmatrix} \mathbf{u}_{xy} \\ u_{yy} \end{bmatrix} \sigma_{d+1}
$$

and we can substitute this into the equation above to get

$$
[\boldsymbol{\epsilon}_\mathbf{X}, \boldsymbol{\epsilon}_\mathbf{y}] = -[\mathbf{X}, \mathbf{y}] \begin{bmatrix} \mathbf{v}_{xy} \\ v_{yy} \end{bmatrix} \begin{bmatrix} \mathbf{v}_{xy} \\ v_{yy} \end{bmatrix}^\top
$$

and thus

$$
[\mathbf{X}, \mathbf{y}] + [\boldsymbol{\epsilon}_\mathbf{X}, \boldsymbol{\epsilon}_\mathbf{y}] = [\mathbf{X}, \mathbf{y}] - [\mathbf{X}, \mathbf{y}] \begin{bmatrix} \mathbf{v}_{xy} \\ v_{yy} \end{bmatrix} \begin{bmatrix} \mathbf{v}_{xy} \\ v_{yy} \end{bmatrix}^\top
$$

$$[\mathbf{X}+\boldsymbol{\epsilon}_\mathbf{X}, \mathbf{y}+\boldsymbol{\epsilon}_\mathbf{y}]\begin{bmatrix}\mathbf{v}_{xy}\\ v_{yy}\end{bmatrix} = [\mathbf{X},\mathbf{y}]\begin{bmatrix}\mathbf{v}_{xy}\\ v_{yy}\end{bmatrix} - [\mathbf{X},\mathbf{y}]\begin{bmatrix}\mathbf{v}_{xy}\\ v_{yy}\end{bmatrix}$$
$$= \mathbf{0}$$

This makes the nonzero $\begin{bmatrix}\mathbf{v}_{xy}\\ v_{yy}\end{bmatrix}$ be in the nullspace, and hence almost gives us what we want $\begin{bmatrix}\mathbf{w}\\ -1\end{bmatrix}$, except for scaling and having v_{yy} instead of a -1. We know from the notes on Total Least Squares that v_{yy} is not 0 almost surely for continuous noise, so we can just rescale to find $\mathbf{w} = -\mathbf{v}_{xy}v_{yy}^{-1}$ from Equation (5).

(f) From the previous part, you can see that $\begin{bmatrix}\mathbf{w}\\ -1\end{bmatrix}$ is a right-singular vector of $[\mathbf{X},\mathbf{y}]$. **Show that**

$$(\mathbf{X}^\top\mathbf{X} - \sigma_{d+1}^2 I)\mathbf{w} = \mathbf{X}^\top\mathbf{y} \tag{14}$$

Solution:

Since $\begin{bmatrix}\mathbf{w}\\ -1\end{bmatrix}$ is a right-singular vector of $\begin{bmatrix}\mathbf{X} & \mathbf{y}\end{bmatrix}$, it is a right-eigenvector of the matrix

$$\begin{bmatrix}\mathbf{X} & \mathbf{y}\end{bmatrix}^\top\begin{bmatrix}\mathbf{X} & \mathbf{y}\end{bmatrix} = \begin{bmatrix}\mathbf{X}^\top\mathbf{X} & \mathbf{X}^\top\mathbf{y}\\ \mathbf{y}^\top\mathbf{X} & \mathbf{y}^\top\mathbf{y}\end{bmatrix} \tag{11}$$

So to find it we solve

$$\begin{bmatrix}\mathbf{X}^\top\mathbf{X} & \mathbf{X}^\top\mathbf{y}\\ \mathbf{y}^\top\mathbf{X} & \mathbf{y}^\top\mathbf{y}\end{bmatrix}\begin{bmatrix}\mathbf{w}\\ -1\end{bmatrix} = \sigma_{d+1}^2\begin{bmatrix}\mathbf{w}\\ -1\end{bmatrix} \tag{12}$$

Ignore the bottom equation and consider the solution of the top equation:

$$\mathbf{X}^\top\mathbf{X}\mathbf{w} - \mathbf{X}^\top\mathbf{y} = \sigma_{d+1}^2\mathbf{w} \tag{13}$$

which can be rewritten as

$$(\mathbf{X}^\top\mathbf{X} - \sigma_{d+1}^2 I)\mathbf{w} = \mathbf{X}^\top\mathbf{y} \tag{14}$$

This result is like ridge regression, but with a *negative* regularization constant!

3 Random Feature Embeddings

In this question, we revisit the task of dimensionality reduction. Dimensionality reduction is useful for several purposes, including visualization, storage, faster computation, etc. We can formalize dimensionality reduction as an embedding function, or *embedding*, $\psi : \mathbb{R}^d \to \mathbb{R}^k$, which maps data points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ with d-dimensional features to reduced data points $\psi(\mathbf{x}_1), \ldots, \psi(\mathbf{x}_n)$ with k-dimensional features.

For the reduced data to remain useful, it may be necessary for the reductions to preserve some properties of the original data. Often, geometric properties like distance and inner products are important for machine learning tasks. And as a result, we may want to perform dimensionality reduction while ensuring that we approximately maintain the pairwise distances and inner products.

While you have already seen many properties of PCA so far, in this question we investigate whether random feature embeddings are a good alternative for dimensionality reduction. A few advantages of random feature embeddings over PCA can be: (1) PCA is expensive when the underlying dimension is high and the number of principal components is also large (however note that there are several very fast algorithms dedicated to doing PCA), (2) PCA requires you to have access to the feature matrix for performing computations. The second requirement of PCA is a bottleneck when you want to take only a low dimensional measurement of a very high dimensional data, e.g., in FMRI and in compressed sensing. In such cases, one needs to design an embedding scheme before seeing the data. We now turn to a concrete setting to study a few properties of PCA and random feature embeddings.

Suppose you are given n points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in $\mathbb{R}^d$.

Notation: The symbol $[n]$ stands for the set $\{1, \ldots, n\}$.

(a) Now consider an arbitrary embedding $\psi : \mathbb{R}^d \mapsto \mathbb{R}^k$ which preserves all pairwise distances and norms up-to a multiplicative factor for all points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in the data set, that is,

$$(1-\epsilon)\|\mathbf{x}_i\|^2 \leq \|\psi(\mathbf{x}_i)\|^2 \leq (1+\epsilon)\|\mathbf{x}_i\|^2 \qquad \text{for all } i \in [n], \quad \text{and} \quad (6)$$

$$(1-\epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|^2 \leq \|\psi(\mathbf{x}_i) - \psi(\mathbf{x}_j)\|^2 \leq (1+\epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|^2 \qquad \text{for all } i, j \in [n], \quad (7)$$

where $0 < \epsilon \ll 1$ is a small scalar. Further assume that $\|\mathbf{x}_i\| \leq 1$ for all $i \in [n]$. **Show that the embedding ψ satisfying equations (7) and (6) preserves *each pairwise inner product*:**

$$|\psi(\mathbf{x}_i)^\top \psi(\mathbf{x}_j) - (\mathbf{x}_i^\top \mathbf{x}_j)| \leq C\epsilon, \qquad \text{for all } i, j \in [n], \quad (8)$$

for some constant C. Thus, we find that if an embedding approximately preserves distances and norms up to a small multiplicative factor, and the points have bounded norms, then inner products are also approximately preserved upto an additive factor.

Hint: Break up the problem into showing that $\psi(\mathbf{x}_i)^\top \psi(\mathbf{x}_j) - (\mathbf{x}_i^\top \mathbf{x}_j) \geq C\epsilon$, and $\psi(\mathbf{x}_i)^\top \psi(\mathbf{x}_j) - (\mathbf{x}_i^\top \mathbf{x}_j) \leq C\epsilon$. The constant $C = 3$ should work, though you can use a larger constant if you need. You may also want to use the Cauchy-Schwarz inequality.

Solution: We will show that

$$-3\epsilon \leq (\mathbf{x}_i^\top \mathbf{x}_j - \psi(\mathbf{x}_i)^\top \psi(\mathbf{x}_j)) \leq 3\epsilon.$$

Taking absolute value, we obtain the claimed result for $C = 3$.

Upper bound: Expanding the second inequality from equation (7), we obtain

$$\left\|\psi(\mathbf{x}_i)\right\|^2 + \left\|\psi(\mathbf{x}_j)\right\|^2 - 2\psi(\mathbf{x}_i)^\top\psi(\mathbf{x}_j) \leq (1+\epsilon)\|\mathbf{x}_i\|^2 + (1+\epsilon)\left\|\mathbf{x}_j\right\|^2 - 2(1+\epsilon)\mathbf{x}_i^\top\mathbf{x}_j.$$

We use the lower bounds from equation (6):

$$\begin{aligned}\left\|\psi(\mathbf{x}_i)\right\|^2 &\geq (1-\epsilon)\|\mathbf{x}_i\|^2\\ \left\|\psi(\mathbf{x}_j)\right\|^2 &\geq (1-\epsilon)\left\|\mathbf{x}_j\right\|^2.\end{aligned}$$

And hence we have

$$(1-\epsilon)\|\mathbf{x}_i\|^2 + (1-\epsilon)\left\|\mathbf{x}_j\right\|^2 - 2\psi(\mathbf{x}_i)^\top\psi(\mathbf{x}_j) \leq (1+\epsilon)\|\mathbf{x}_i\|^2 + (1+\epsilon)\left\|\mathbf{x}_j\right\|^2 - 2(1+\epsilon)\mathbf{x}_i^\top\mathbf{x}_j.$$

Moving terms around we obtain

$$\begin{aligned}2(\mathbf{x}_i^\top\mathbf{x}_j - \psi(\mathbf{x}_i)^\top\psi(\mathbf{x}_j)) &\leq 2\epsilon(\|\mathbf{x}_i\|^2 + \left\|\mathbf{x}_j\right\|^2 - \mathbf{x}_i^\top\mathbf{x}_j)\\ &\leq 2\epsilon(\|\mathbf{x}_i\|^2 + \left\|\mathbf{x}_j\right\|^2 + \|\mathbf{x}_i\|\left\|\mathbf{x}_j\right\|)\\ &\leq 6\epsilon,\end{aligned}$$

since $\|\mathbf{x}_i\| \leq 1$. Thus we have

$$(\mathbf{x}_i^\top\mathbf{x}_j - \psi(\mathbf{x}_i)^\top\psi(\mathbf{x}_j)) \leq 3\epsilon.$$

Lower bound: Expanding the first inequality from equation (7), we obtain

$$\left\|\psi(\mathbf{x}_i)\right\|^2 + \left\|\psi(\mathbf{x}_j)\right\|^2 - 2\psi(\mathbf{x}_i)^\top\psi(\mathbf{x}_j) \geq (1-\epsilon)\|\mathbf{x}_i\|^2 + (1-\epsilon)\left\|\mathbf{x}_j\right\|^2 - 2(1-\epsilon)\mathbf{x}_i^\top\mathbf{x}_j.$$

We use the upper bounds from equation (6):

$$\begin{aligned}\left\|\psi(\mathbf{x}_i)\right\|^2 &\leq (1+\epsilon)\|\mathbf{x}_i\|^2\\ \left\|\psi(\mathbf{x}_j)\right\|^2 &\leq (1+\epsilon)\left\|\mathbf{x}_j\right\|^2.\end{aligned}$$

And hence we have

$$(1+\epsilon)\|\mathbf{x}_i\|^2 + (1+\epsilon)\left\|\mathbf{x}_j\right\|^2 - 2\psi(\mathbf{x}_i)^\top\psi(\mathbf{x}_j) \geq (1-\epsilon)\|\mathbf{x}_i\|^2 + (1-\epsilon)\left\|\mathbf{x}_j\right\|^2 - 2(1-\epsilon)\mathbf{x}_i^\top\mathbf{x}_j.$$

Moving terms around we obtain

$$\begin{aligned}2(\mathbf{x}_i^\top\mathbf{x}_j - \psi(\mathbf{x}_i)^\top\psi(\mathbf{x}_j)) &\geq -2\epsilon(\|\mathbf{x}_i\|^2 + \left\|\mathbf{x}_j\right\|^2 - \mathbf{x}_i^\top\mathbf{x}_j)\\ &\geq -2\epsilon(\|\mathbf{x}_i\|^2 + \left\|\mathbf{x}_j\right\|^2 + \|\mathbf{x}_i\|\left\|\mathbf{x}_j\right\|)\\ &\geq -6\epsilon,\end{aligned}$$

since $\|\mathbf{x}_i\| \leq 1$. And thus we are done.

(b) Now we consider the *random feature embedding* using a Gaussian matrix. In next few parts, we work towards proving that if the dimension of embedding is moderately big, then with high probability, the random embedding preserves norms and pairwise distances approximately as described in equations (7) and (6).

Consider the random matrix $\mathbf{J} \in \mathbb{R}^{k \times d}$ with each of its entries being i.i.d. $\mathcal{N}(0,1)$ and consider the map $\psi_{\mathbf{J}} : \mathbb{R}^d \mapsto \mathbb{R}^k$ such that $\psi_{\mathbf{J}}(\mathbf{x}) = \frac{1}{\sqrt{k}}\mathbf{J}\mathbf{x}$. **Show that for any *fixed non-zero vector* $\mathbf{u}$, the random variable $\dfrac{\|\psi_{\mathbf{J}}(\mathbf{u})\|^2}{\|\mathbf{u}\|^2}$ can be written as**

$$\frac{1}{k}\sum_{i=1}^{k} Z_i^2$$

where Z_i's are i.i.d. $\mathcal{N}(0,1)$ random variables.

Solution: Let $\mathbf{J} = \begin{bmatrix} J_1^\top \\ \vdots \\ J_k^\top \end{bmatrix}$. Then, we have

$$\frac{\|\psi_{\mathbf{J}}(\mathbf{u})\|^2}{\|\mathbf{u}\|^2} = \frac{1}{k}\sum_{i=1}^{k} \frac{(J_i^\top \mathbf{u})^2}{\|\mathbf{u}\|^2}.$$

Note that

$$J_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \mathbf{I}_d),$$

and hence

$$(J_i^\top \mathbf{u}) \sim \mathcal{N}(0, \|\mathbf{u}\|^2),$$

and consequently, we have

$$Z_i = \frac{J_i^\top \mathbf{u}}{\|\mathbf{u}\|} \overset{i.i.d.}{\sim} \mathcal{N}(0,1).$$

As a result, we conclude that

$$\frac{\|\psi_{\mathbf{J}}(\mathbf{u})\|^2}{\|\mathbf{u}\|^2} = \frac{1}{k}\sum_{i=1}^{k} \frac{(J_i^\top \mathbf{u})^2}{\|\mathbf{u}\|^2}. = \frac{1}{k}\sum_{i=1}^{k} Z_i^2$$

where Z_i's are i.i.d. $\mathcal{N}(0,1)$ random variables.

(c) For any fixed pair of indices $i \neq j$, define the events

$$A_{ij} := \left\{ \frac{\left\|\psi_{\mathbf{J}}(\mathbf{x}_i) - \psi_{\mathbf{J}}(\mathbf{x}_j)\right\|^2}{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2} \in (1-\epsilon, 1+\epsilon) \right\}.$$

which corresponds to the event that the embedding $\psi_{\mathbf{J}}$ approximately preserves the angles between $\mathbf{x}_i$ and $\mathbf{x}_j$. In this part, we show that A_{ij} occurs with high probability.

To do this, you will use the fact that for independent random variables $Z_i \sim \mathcal{N}(0,1)$, we have the following probability bound

$$\mathbb{P}\left[\left|\frac{1}{k}\sum_{i=1}^{k} Z_i^2\right| \notin (1-t, 1+t)\right] \leq 2e^{-kt^2/8}, \qquad \text{for all } t \in (0,1).$$

Note that this bound suggests that $\sum_{i=1}^{k} Z_i^2 \approx k = \sum_{i=1}^{k} \mathbb{E}[Z_i^2]$ with high probability. In other words, sum of squares of Gaussian random variables concentrates around its mean with high probability. **Using this bound and the previous subproblem, show that**

$$\mathbb{P}\left[A_{ij}^c\right] \leq 2e^{-k\epsilon^2/8},$$

where A_{ij}^c denotes the complement of the event A_{ij}. **Solution:**

$$\begin{aligned}\mathbb{P}\left[A_{ij}^c\right] &= \mathbb{P}\left[\frac{\left\|\psi_{\mathbf{J}}(\mathbf{x}_i) - \psi_{\mathbf{J}}(\mathbf{x}_j)\right\|^2}{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2} \notin (1-\epsilon, 1+\epsilon)\right] \\ &\overset{\text{part (e)}}{=} \mathbb{P}\left[\left|\frac{1}{k}\sum_{i=1}^{k} Z_i^2\right| \notin (1-\epsilon, 1+\epsilon)\right] \\ &\overset{\text{prob. bnd}}{\leq} 2e^{-k\epsilon^2/8}.\end{aligned}$$

(d) Using the previous problem, now **show that if** $k \geq \frac{16}{\epsilon^2}\log\left(\frac{N}{\delta}\right)$**, then**

$$\mathbb{P}\left[\text{for all } i,j \in [n], i \neq j, \frac{\left\|\psi_{\mathbf{J}}(\mathbf{x}_i) - \psi_{\mathbf{J}}(\mathbf{x}_j)\right\|^2}{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2} \in (1-\epsilon, 1+\epsilon)\right] \geq 1-\delta.$$

That is show that for k large enough, with high probability the random feature embedding $\psi_{\mathbf{J}}$ approximately preserves the pairwise distances. Using this result, we can conclude that random feature embedding serves as a good tool for dimensionality reduction if we project to enough number of dimensions. This result is popularly known as the *Johnson-Lindenstrauss Lemma*.

Hint 1: Let

$$\mathcal{A} := \left\{\text{for all } i,j \in [n], i \neq j, \frac{\left\|\psi_{\mathbf{J}}(\mathbf{x}_i) - \psi_{\mathbf{J}}(\mathbf{x}_j)\right\|^2}{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2} \in (1-\epsilon, 1+\epsilon)\right\}$$

denote the event whose probability we would like to lower bound. Express the complement $\mathcal{A}^c$ in terms of the events A_{ij}^c, and try to apply a union bound to these events.

Solution: Recall the event from the hint:

$$\mathcal{A} := \left\{ \text{for all } i,j \in [n], i \neq j, \frac{\left\|\psi_{\mathbf{J}}(\mathbf{x}_i) - \psi_{\mathbf{J}}(\mathbf{x}_j)\right\|^2}{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2} \in (1-\epsilon, 1+\epsilon) \right\}$$

is equivalent to the intersection of all events A_{ij}, since all events A_{ij} must occur for $\mathcal{A}$ to occur. That is

$$\mathcal{A} = \bigcap_{i\in[n], j\in n, i\neq j} A_{ij}.$$

Note that there are $\binom{n}{2}$ events. Now as given in the hint, we have

$$\begin{aligned}
\mathbb{P}\left[\mathcal{A}\right] &= \mathbb{P}\left[\bigcap_{i,j\in[n],i\neq j} A_{ij}\right] \\
&= 1 - \mathbb{P}\left[\left(\bigcap_{i,j\in[n],i\neq j} A_{ij}\right)^c\right] \\
&= 1 - \mathbb{P}\left[\bigcup_{i,j\in[n],i\neq j} A_{ij}^c\right] \\
&\geq 1 - \sum_{i\in[n],j\in[n],i\neq j} \mathbb{P}\left[A_{ij}^c\right].
\end{aligned}$$

By the previous part of the problem,

$$\mathbb{P}\left[A_{ij}^c\right] \leq 2e^{-k\epsilon^2/8}.$$

Thus, we have that

$$\begin{aligned}
\mathbb{P}\left[\mathcal{A}\right] &\geq 1 - \sum_{i\in[n],j\in[n],i\neq j} \mathbb{P}\left[A_{ij}^c\right] \\
&\geq 1 - \sum_{i\in[n],j\in[n],i\neq j} 2e^{-k\epsilon^2/8} \\
&\geq 1 - \binom{n}{2} 2e^{-k\epsilon^2/8} \\
&\geq 1 - n(n-1)e^{-k\epsilon^2/8} \\
&\geq 1 - n^2 e^{-k\epsilon^2/8}.
\end{aligned}$$

Now to make this probability greater than δ, we set k such that

$$n^2 e^{-k\epsilon^2/8} \leq \delta \Rightarrow 2\log n - k\frac{\epsilon^2}{8} \leq \log\delta \Rightarrow k \geq \frac{8}{\epsilon^2}\left(2\log n - \log\delta\right).$$

Note that since $\delta < 1$, we have that $k \geq \frac{16}{\epsilon^2}\log\left(\frac{N}{\delta}\right)$ implies that $k \geq \frac{8}{\epsilon^2}\left(2\log n - \log\delta\right)$, and hence we are done.

(e) Suppose there are two clusters of points $S_1 = \{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$ and $S_2 = \{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$ which are far apart, i.e., we have

$$d^2(S_1, S_2) = \min_{u \in S_1, v \in S_2} \|u - v\|^2 \geq \gamma.$$

Then using the previous part, **show that the random feature embedding $\psi_{\mathbf{J}}$ also approximately maintains the distance between the two clusters if k is large enough, that is, with high probability**

$$d^2(\psi_{\mathbf{J}}(S_1), \psi_{\mathbf{J}}(S_2)) = \min_{u \in S_1, v \in S_2} \left\|\psi_{\mathbf{J}}(\mathbf{u}) - \psi_{\mathbf{J}}(\mathbf{v})\right\|^2 \geq (1-\epsilon)\gamma \quad \text{if } k \geq \frac{C}{\epsilon^2}\log(m+n)$$

for some constant C. Note that such a property can help in several machine learning tasks. For example, if the clusters of features with different labels were far in the original dimension, then this problem shows that they will remain far in the smaller dimension.Therefore, a machine learning model can perform well even with the randomly projected data.

Solution: Using previous part, we obtain that pairwise squared distances between $m+n$ points are approximately preserved up to a factor of $(1-\epsilon, 1+\epsilon)$ if $k \geq C\log(m+n)/\epsilon^2$. Now, we consider any pair of points which minimize the objective $\min_{u \in S_1, v \in S_2} \left\|\psi_{\mathbf{J}}(\mathbf{u}) - \psi_{\mathbf{J}}(\mathbf{v})\right\|^2$. Let's call these points $\mathbf{u}_{i^*}$ and $\mathbf{v}_{j^*}$. Then we have

$$\begin{aligned}
\min_{u \in S_1, v \in S_2} \left\|\psi_{\mathbf{J}}(\mathbf{u}) - \psi_{\mathbf{J}}(\mathbf{v})\right\|^2 &= \left\|\psi_{\mathbf{J}}(\mathbf{u}_{i^*}) - \psi_{\mathbf{J}}(\mathbf{v}_{j^*})\right\|^2 \\
&\geq (1-\epsilon)\left\|\mathbf{u}_{i^*} - \mathbf{v}_{j^*}\right\|^2 \\
&\geq (1-\epsilon) \min_{u \in S_1, v \in S_2} \|u - v\|^2 \\
&= (1-\epsilon)\gamma,
\end{aligned}$$

and we are done.

4 Comparing Random Features to PCA Embeddings

In this problem, we compare the performance of the randomized embeddings from the Johnson-Lindenstrauss (JL) Lemma to embeddings from PCA. Again, suppose we are given given n points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in $\mathbb{R}^d$. Define the $n \times d$ matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix}$ where each row of the matrix represents one of the given points. In this problem, we will compare an embedding derived from PCA to the random features embedding from the previous problem.

Let $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ with $\mathbf{U} \in \mathbb{R}^{n \times d}$, $\mathbf{V} \in \mathbb{R}^{d \times d}$, and $\mathbf{\Sigma} \in \mathbb{R}^{d \times d}$ denote the compact singular value decomposition of the matrix $\mathbf{X}$. Assume that $n \geq d$ and let $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_d$ denote the singular values of $\mathbf{X}$.

Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$ denote the columns of the matrix $\mathbf{V}$. We now consider the following k-dimensional PCA embedding: $\psi_{\text{PCA}}(\mathbf{x}) = (\mathbf{v}_1^\top \mathbf{x}, \ldots, \mathbf{v}_k^\top \mathbf{x})^\top$. Note that this embedding projects a d-dimensional

vector on the linear span of the set $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$, and changes the basis so that $\mathbf{v}_i^\top \mathbf{x}$ denotes the i-th coordinate of the projected vector in the new space. We begin with a few matrix algebra relationships, which we use to investigate certain mathematical properties of PCA and random embeddings. In the later parts of this problem, we will see them in action on a synthetic dataset.

Notation: The symbol $[n]$ stands for the set $\{1, \ldots, n\}$.

(a) **What is the ij-th entry of the matrices $\mathbf{X}\mathbf{X}^\top$ and $\mathbf{X}^\top\mathbf{X}$ in terms of elements of $\mathbf{X}$? Express the matrix $\mathbf{X}\mathbf{X}^\top$ in terms of $\mathbf{U}$ and $\mathbf{\Sigma}$, and, express the matrix $\mathbf{X}^\top\mathbf{X}$ in terms of $\mathbf{\Sigma}$ and $\mathbf{V}$.**

Solution: Let x_{ik} denote the k-th entry of the vector $\mathbf{x}_i$. Then, we have

$$(\mathbf{X}\mathbf{X}^\top)_{ij} = \left[\begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 & \ldots & \mathbf{x}_n \end{bmatrix}\right]_{ij} = \mathbf{x}_i^\top \mathbf{x}_j = \sum_{k=1}^{d} x_{ik} x_{jk}$$

and

$$(\mathbf{X}^\top\mathbf{X})_{ij} = \sum_{k=1}^{n} (\mathbf{X}^\top)_{ik} (\mathbf{X})_{kj} = \sum_{k=1}^{n} x_{ki} x_{kj}.$$

Furthermore, we have

$$\mathbf{X}\mathbf{X}^\top = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^\top \quad \text{and} \quad \mathbf{X}^\top\mathbf{X} = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^\top.$$

(b) **Show that**

$$\psi_{\text{PCA}}(\mathbf{x}_i)^\top \psi_{\text{PCA}}(\mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{V}_k \mathbf{V}_k^\top \mathbf{x}_j \quad \text{where} \quad \mathbf{V}_k = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \ldots & \mathbf{v}_k \end{bmatrix}.$$

Also **show that** $\mathbf{V}_k\mathbf{V}_k^\top = \mathbf{V}\mathbf{I}^k\mathbf{V}^\top$, where the matrix $\mathbf{I}^k$ denotes a $d \times d$ diagonal matrix with first k diagonal entries as 1 and all other entries as zero.

Solution: Note that

$$\psi_{\text{PCA}}(\mathbf{x}) = \begin{bmatrix} \mathbf{v}_1^\top \mathbf{x} \\ \vdots \\ \mathbf{v}_k^\top \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1^\top \\ \vdots \\ \mathbf{v}_k^\top \end{bmatrix} \mathbf{x} = \mathbf{V}_k^\top \mathbf{x}$$

and thus we have

$$\psi_{\text{PCA}}(\mathbf{x}_i)^\top \psi_{\text{PCA}}(\mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{V}_k \mathbf{V}_k^\top \mathbf{x}_j$$

as required.

For the second part, we note that for two matrices

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \dots & \mathbf{a}_d \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{b}_1^\top \\ \vdots \\ \mathbf{b}_d^\top \end{bmatrix}$$

we have

$$\mathbf{AB} = \sum_{i=1}^{d} \mathbf{a}_i \mathbf{b}_i^\top. \tag{9}$$

Furthermore, given a diagonal matrix $\mathbf{\Gamma}$ of size d-by$d$, with $(\mathbf{\Gamma})_{ii} = \gamma_i$, we have

$$\mathbf{A\Gamma B} = \sum_{i=1}^{d} \gamma_i \mathbf{a}_i \mathbf{b}_i^\top. \tag{10}$$

Using equation (9), we have

$$\mathbf{V}_k \mathbf{V}_k^\top = \sum_{i=1}^{k} \mathbf{v}_i \mathbf{v}_i^\top$$

Furthermore, using equation (10), we obtain that

$$\mathbf{V} \mathbf{I}^k \mathbf{V}^\top = \sum_{i=1}^{d} \mathbf{v}_i (\mathbf{I}^k)_{ii} \mathbf{v}_i^\top = \sum_{i=1}^{k} \mathbf{v}_i \mathbf{v}_i^\top.$$

since $(\mathbf{I}^k)_{ii} = 1$ for $i = 1, \dots, k$ and 0 for $i > k$. The students are not required to prove so rigorously. The key take away from this part is to be comfortable with different ways to write matrices, and keep in mind the representations (9) and (10).

(c) Suppose that we know the first k singular values are the dominant singular values. In particular, we are given that

$$\frac{\sum_{i=1}^{k} \sigma_i^4}{\sum_{i=1}^{d} \sigma_i^4} \geq 1 - \epsilon,$$

for some $\epsilon \in (0, 1)$. Then **show that the PCA embedding to the first k-right singular vectors preserves the *inner products* on average:**

$$\frac{1}{\sum_{i=1}^{n} \sum_{j=1}^{n} (\mathbf{x}_i^\top \mathbf{x}_j)^2} \sum_{i=1}^{n} \sum_{j=1}^{n} |(\mathbf{x}_i^\top \mathbf{x}_j) - (\psi_{\text{PCA}}(\mathbf{x}_i)^\top \psi_{\text{PCA}}(\mathbf{x}_j))|^2 \leq \epsilon. \tag{11}$$

Thus, we find that if there are dominant singular values, PCA embedding can preserve the inner products on average.

Hint: Using previous two parts and the definition of Frobenius norm might be useful.

Solution: Using the solution from part (a), we have

$$(\mathbf{X}\mathbf{X}^\top)_{ij} = \mathbf{x}_i^\top \mathbf{x}_j. \tag{12}$$

Using the solution from part (b), we have

$$\Psi = \begin{bmatrix} \psi_{\text{PCA}}(\mathbf{x}_1)^\top \\ \vdots \\ \psi_{\text{PCA}}(\mathbf{x}_n)^\top \end{bmatrix} = \mathbf{X}\mathbf{V}_k. \tag{13}$$

We have

$$\begin{aligned} \sum_{i=1}^{n}\sum_{j=1}^{n} |(\mathbf{x}_i^\top \mathbf{x}_j) - (\psi_{\text{PCA}}(\mathbf{x}_i)^\top \psi_{\text{PCA}}(\mathbf{x}_j))|^2 &\overset{(i)}{=} \|\mathbf{X}\mathbf{X}^\top - \Psi\Psi^\top\|_{\text{F}}^2 \\ &\overset{(ii)}{=} \|\mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^\top - \mathbf{X}\mathbf{V}_k\mathbf{V}_k^\top\mathbf{X}^\top\|_{\text{F}}^2 \\ &\overset{(iii)}{=} \|\mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^\top - \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top\mathbf{V}\mathbf{I}^k\mathbf{V}^\top\mathbf{V}\mathbf{\Sigma}\mathbf{U}^\top\|_{\text{F}}^2 \\ &\overset{(iv)}{=} \|\mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^\top - \mathbf{U}\mathbf{\Sigma}\mathbf{I}_k\mathbf{\Sigma}\mathbf{U}^\top\|_{\text{F}}^2, \end{aligned}$$

where for step (i) we applied the definition of the Frobenius norm and the equations (12) and (13), for step (ii) we have used the solution from part (a) to replace $\mathbf{X}\mathbf{X}^\top = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^\top$ and for step (iii), we have used the singular value decomposition $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, and the second result from part (b). For step (iv) we make use of the fact that $\mathbf{V}^\top\mathbf{V} = \mathbf{I}$.

Now we use the fact that Frobenius norm is unitary invariant (Discussion 1, problem 2 (b)). Thus, we have

$$\|\mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^\top - \mathbf{U}\mathbf{\Sigma}\mathbf{I}_k\mathbf{\Sigma}\mathbf{U}^\top\|_{\text{F}}^2 = \|\mathbf{U}\mathbf{\Sigma}^2 - \mathbf{\Sigma}\mathbf{I}_k\mathbf{\Sigma}\mathbf{U}^\top\|_{\text{F}}^2 \quad = \|\mathbf{\Sigma}^2 - \mathbf{\Sigma}\mathbf{I}_k\mathbf{\Sigma}\|_{\text{F}}^2 = \sum_{i=k+1}^{d} \sigma_i^4.$$

We now use the fact that

$$\sum_{i=1}^{n}\sum_{j=1}^{n} (\mathbf{x}_i^\top \mathbf{x}_j)^2 = \|\mathbf{X}\mathbf{X}^\top\|_{\text{F}}^2 = \|\Sigma^2\|_{\text{F}}^2 = \sum_{i=1}^{d} \sigma_i^4,$$

we obtain that

$$\begin{aligned} \frac{1}{\sum_{i=1}\sum_{j=1}(\mathbf{x}_i^\top \mathbf{x}_j)^2} \sum_{i=1}^{n}\sum_{j=1}^{n} |(\mathbf{x}_i^\top \mathbf{x}_j) - (\psi_{\text{PCA}}(\mathbf{x}_i)^\top \psi_{\text{PCA}}(\mathbf{x}_j))|^2 &= \frac{\sum_{i=k+1}^{d} \sigma_i^4}{\sum_{i=1}^{d} \sigma_i^4} \\ &= 1 - \frac{\sum_{i=1}^{k} \sigma_i^4}{\sum_{i=1}^{d} \sigma_i^4} \\ &\leq \epsilon. \end{aligned}$$

(d) In the next few parts, we visualize the effect of PCA embeddings and random embeddings on a classification task. **First, fill in the method** `random_JL_matrix(d,k)` **and the method** `pca_embedding_matrix(X,k)`**, as specified in the starter code**. Please list your final code below, in a screenshot or using the `lstinputlisting` package in LaTex.

Solution: The code for this part and the following parts is as follows:

```
import numpy as np
import matplotlib.pyplot as plt
# %matplotlib inline
import sklearn.linear_model
from sklearn.model_selection import train_test_split


######## embedding FUNCTIONS ##########
######## YOUR CODE ##########


## Input: original dimension d,
## Input: projected dimension k
## Output: d x k random
## Gaussian matrix J with entry-wise
## variances 1/k so that,
## for any row vector z^T in R^d,
## z^T J  is a random features embedding for z^T
def random_JL_matrix(d, k):
    return 1./np.sqrt(k)*np.random.normal(0, 1, (d, k))


## Input: n x d data matrix X
## Input: projected dimension k
## Output: d x k matrix V
## with orthonormal columns
## corresponding to the top k right signular vectors
## of X. Thus, for a row vector z^T in R^d
## z^T V  is the projection of z^T
## onto the the top k right-singular vectors of X,
def pca_embedding_matrix(X, k):
    n, d = X.shape
    assert(d>=k)
    _, _, Vh = np.linalg.svd(X)
    V = Vh.T
    V = V[:, :k]
    return V

# applies the linear transformation N
# to the rows of X, via X.dot(N)
def linear_feature_transform(X, N):
    return X.dot(N)

# uses pca_embedding_matrix method
# to project the rows of X onto the first
# k principle components
def pca_proj(X,k):
    V = pca_embedding_matrix(X, k)
    return linear_feature_transform(X,V)

# uses random_JL_matrix method
# to transform rows of X
# by a k-dimensional JL transform
def random_proj(X,k):
    _, d= X.shape
    J = random_JL_matrix(d, k)
    return linear_feature_transform(X,J)


```

```
60
61
62 ######### LINEAR MODEL FITTING ############
63 ######### DO NOT ALTER ##################
64 def rand_embed_accuracy_split(X, y, k):
65     '''
66     Fitting a k dimensional feature set obtained
67     from random embedding of X, versus y
68     for binary classification for y in {-1, 1}
69     '''
70
71     # test train split
72     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state
       ↪ =42)
73
74     # random embedding
75     _, d = X.shape
76     J = random_JL_matrix(d,k)
77     rand_embed_X = linear_feature_transform(X_train, J)
78
79     # fit a linear model
80     line = sklearn.linear_model.LinearRegression(fit_intercept=False)
81     line.fit(rand_embed_X, y_train)
82
83     # predict y
84     y_pred=line.predict(linear_feature_transform(X_test, J))
85
86     # return the test error
87     return 1-np.mean(np.sign(y_pred)!= y_test)
88
89 def pca_embed_accuracy_split(X, y, k):
90     '''
91     Fitting a k dimensional feature set obtained
92     from PCA embedding of X, versus y
93     for binary classification for y in {-1, 1}
94     '''
95
96     # test-train split
97     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
98
99     # pca embedding
100    V = pca_embedding_matrix(X_train, k)
101    pca_embed_X = linear_feature_transform(X_train,V)
102
103    # fit a linear model
104    line = sklearn.linear_model.LinearRegression(fit_intercept=False)
105    line.fit(pca_embed_X, y_train)
106
107     # predict y
108    y_pred=line.predict(linear_feature_transform(X_test,V))
109
110
111    # return the test error
112    return 1-np.mean(np.sign(y_pred)!= y_test)
113
114
115 ######## LOADING THE DATASETS #########
116
117 # to load the data:
118 # data = np.load('data/data1.npz')
119 # X = data['X']
120 # y = data['y']
121 # n, d = X.shape
122
123
124 # n_trials = 10  # to average for accuracies over random embeddings
125
126 ######### YOUR CODE GOES HERE ##########
```

```
127
128 # Using PCA and Random embedding for:
129 # Visualizing the datasets
130
131 # Computing the accuracies over different datasets.
132 # Don't forget to average the accuracy for multiple
133 # random embeddings to get a smooth curve.
134
135 # And computing the SVD of the feature matrix
136 ######## YOU CAN PLOT THE RESULTS HERE ########
137
138 # plt.plot, plt.scatter would be useful for plotting
139
140
141
142 np.random.seed(7283782)
143
144 for i in range(3):
145
146     ######## LOADING THE DATASETS #########
147     filename = 'jl_data/data'
148     filename = '../jl_data_2018f/data'
149
150     data = np.load(filename+str(i+1)+'.npz')
151     X = data['X']
152     y = data['y']
153
154     n, d = X.shape
155     n_trials = 10  # to average the accuracies for random embedding
156
157     ######## CLASSIFICATION USING PROJECTED FEATURES #########
158
159     # linear regression with projected features
160     random_accuracy = np.zeros((n_trials, d))
161     pca_accuracy = np.zeros(d)
162
163     for k in range(1, d+1):
164         pca_accuracy[k-1] = pca_embed_accuracy_split(X, y, k)
165         for j in range(n_trials):
166             random_accuracy[j, k-1] = rand_embed_accuracy_split(X, y, k)
167
168     # take the mean of random embedding performance across trials
169     random_accuracy = np.mean(random_accuracy, axis=0)
170
171     ######## PLOTTING RESULTS #########
172
173     plt.figure(figsize=[15, 3])
174 #     plt.suptitle('Data'+str(i), fontsize=15)
175     plt.subplot(141)
176     p = pca_proj(X, 2)
177     plt.scatter(p[:, 0], p[:, 1], c=y)
178     plt.title('First 2 PC')
179
180     plt.subplot(142)
181     r = random_proj(X, 2)
182     plt.scatter(r[:, 0], r[:, 1], c=y)
183     plt.title('Random 2D embedding')
184
185     _, S, Vh = np.linalg.svd(X)
186     plt.subplot(144)
187     plt.plot(np.arange(1, d+1), S)
188     plt.title('Singular value decay')
189
190     plt.subplot(143)
191     plt.plot(np.arange(1, d+1), random_accuracy, label="Random")
192     plt.plot(np.arange(1, d+1), pca_accuracy, label="PCA")
193     plt.legend(loc='best')
194     plt.title("Accuracy")
```

```
    plt.xlabel("#Dimensions")

    plt.tight_layout()
    plt.savefig('jl'+str(i+1)+'.pdf')
   # plt.show()
```

(e) You are given 3 datasets in the data folder, along with the starter code. Use the starter code to load the three datasets one by one. Note that there are two unique values in $\mathbf{y}$. **Visualize the features of X these datasets using (1) Top-2 PCA components, and (2) 2-dimensional random embeddings. Use the code to project the features to** 2 **dimensions and then scatter plot the** 2 **features with a different color for each value of** $\mathbf{y}$**.** Note that you will obtain 2 plots for each dataset (you should display a total of 6 plots for this part). **Do you observe a difference in PCA vs random embeddings? Do you see a trend in the three datasets?**

Solution:

Remark: For random embeddings used in the regression code, there is a slight typo – the scaling is missing. However, the learned model does not change because scaling the entire feature matrix does not change the ordinary least squares solution.

For plots, refer to figures 1, 2 and 3.

For the first and second datasets it is hard to see a difference between the 2D embeddings with PCA or random embeddings.

For the third dataset, it is clear that PCA embedding clearly separates the two clusters of points, while random embedding still has significant overlap of the two sets of points.

Using the PCA embedding, we can say that the separation between the two sets of features (corresponding to $y = 1$ and $y = -1$) is largest in data3.npz and probably smallest in data1.npz

(f) For each dataset, we will now fit a linear model on different embedding of features to perform classification. The code for fitting a linear model with projected features and predicting a label for a given feature, is given to you. Use these functions and write a code that does prediction in the following way: (1) Use top k-PCA features to obtain one set of results, and (2) Use k-dimensional random embeddings to obtain the second set of results (take average accuracy over 10 random embeddings for smooth curves). Use the embedding functions you filled in in the starter code to select these features. You should vary k from 1 to d where d is the dimension of each feature $\mathbf{x}_i$. **Plot the accuracy for PCA and Random embeddings as a function of** k**. Comment on the observations on these accuracies as a function of** k **and across different datasets.** Attach your plots below.

Solution: For plots, refer to figures 1, 2 and 3.

PCA: We see that the accuracy of PCA increase very quickly after adding 2-3 features only and then does not increase too much. In fact, for data1.npz it decays little bit after first 2 features indicating overfitting.

Random embedding: We see that the accuracy increases moderately with increase in number of dimension of the embedding matrix.

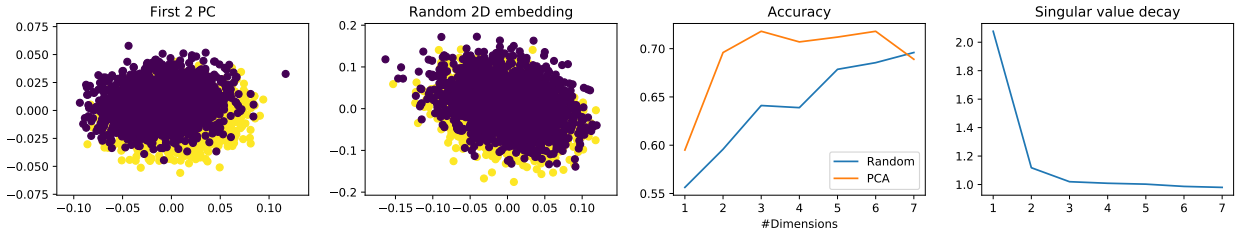

Figure 1: Plots related to data1.npz

The difference in increase in accuracy for PCA and random embedding is intuitive as PCA captures effective dimensions – and it appears from the 2D visualization that signal strength is stronger in a few directions. In particular for data3, the data is separable if we just use one principal component. However, random embedding is oblivious to the structure in the underlying data if we just use very few directions. Although as the number of dimensions increase, its performance does catch up with PCA.

Across the three datasets, we see that the accuracy for a given set of dimensions increases as we move from data1.npz to data2.npz to data3.npz, again suggesting that the signal strength = the separation between the two clusters is increasing from data1 to data3.

(g) Now plot the singular values for the feature matrices of the three datasets, and attach them below. **Do you observe a pattern across the three datasets? Does it help to explain the performance of PCA observed in the previous parts?**

Solution: For plots, refer to figures 1, 2 and 3.

The significance of the top singular value increases from data1 to data2 with maximum value occurring for data3. For data1, first singular value accounts for approximately $2/8 \approx 25\%$ of the variance, while for data2 it explains $2.5/8.5 \approx 30\%$ variance. For Data3 this percentage is close to $4.5/11 \approx 41\%$.

Intuitively, if the labels are dependent on the direction of maximum variance, we hope to see an increase in accuracy if the ratio of variance explained goes up. PCA 2d visualizations suggest that the labeling is associated with the first principal component.

As a result, we do expect PCA with just one component do very well for data3, and in general the classification accuracy to increase from data1 to data3.

However, note that as the signal to noise ratio increases, the accuracy of random embeddings also increases.

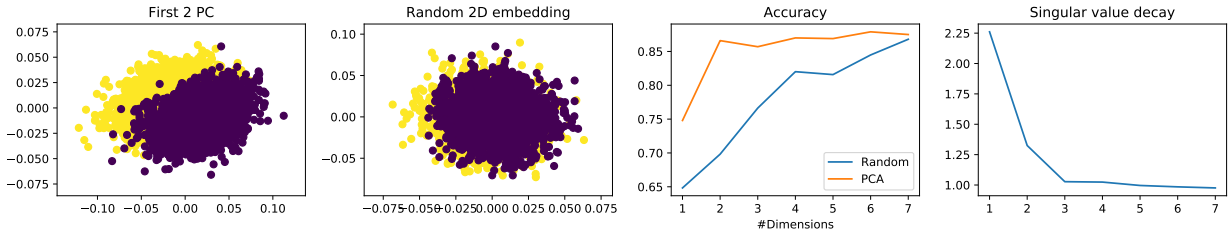

Figure 2: Plots related to data2.npz

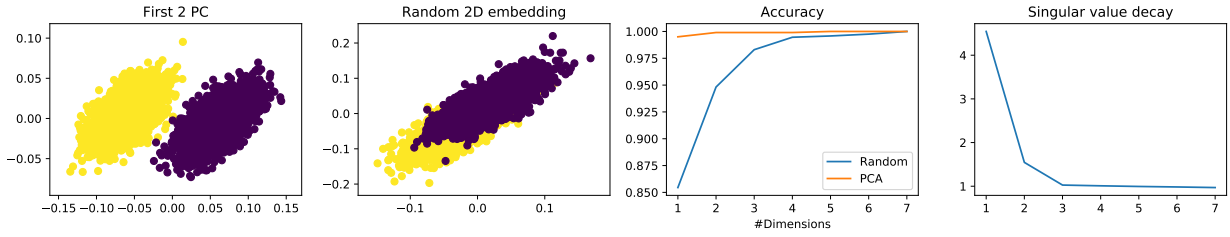

Figure 3: Plots related to data3.npz