CS 189 Introduction to Machine Learning
Fall 2018 HW4

1 Getting Started

Read through this page carefully. You may typeset your homework in latex or submit neatly handwritten/scanned solutions. Please start each question on a new page. Deliverables:

1. Submit a PDF of your writeup to assignment on Gradescope, "HW4 Write-Up". If there are graphs, include those graphs in the correct sections. Do not simply reference your appendix.

(a) Who else did you work with on this homework? In case of course events, just describe the group. How did you work on this homework? Any comments about the homework?

Just myself

(b) Please copy the following statement and sign next to it. We just want to make it *extra* clear so that no one inadverdently cheats.

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.

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.

This homework is due **Monday, October 1st 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)**.**

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

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

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

$\mathbf{u}_{xy} \in \mathbb{R}^{n-1}$ 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 $[\mathbf{X}, \mathbf{y}]$. $\mathbf{\Sigma}_d$ is the diagonal matrix of the d largest singular values of $[\mathbf{X}, \mathbf{y}]$

Using this information show that

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

(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.

(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**

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

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,

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

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

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*:**

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

for some constant C. Thus, we find that if an embedding approximately preserves distances and norms upto 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 $\psi(\mathbf{x}_i)^\top \psi(\mathbf{x}_j) - (\mathbf{x}_i^\top \mathbf{x}_j) \geq C\epsilon\epsilon$, 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.

(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.

(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

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

Note that this bound suggests that $\sum_{i=1}^{k} Z_i^2 \approx k = \sum_{i=1}^{k} E[Z_i^2]$ 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}.

(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.

(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.

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 $\mathbf{V}$.**

(b) **Show that**

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

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.

(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:**

$$\frac{1}{\sum_{i=1}^n \sum_{j=1}^n (\mathbf{x}_i^\top \mathbf{x}_j)^2} \sum_{i=1}^n \sum_{j=1}^n \left| (\mathbf{x}_i^\top \mathbf{x}_j) - (\psi_{\text{PCA}}(\mathbf{x}_i)^\top \psi_{\text{PCA}}(\mathbf{x}_j)) \right|^2 \leq \epsilon. \tag{9}$$

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.

(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.

(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 $\mathbf{X}$ 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?**

(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.

(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?**