Project Report: A New Unsupervised Feature Selection Algorithm

Lane Lewis, Kevin Lin, Alexa Aucoin

December 2021

1 Abstract

A common technique used in unsupervised learning is Principal Component Analysis (PCA). However the components found by performing PCA may not be easily interpreted by researchers using them in analysis. In this paper we investigate an unsupervised feature selection algorithm which selects features with properties similar to principal Components using leverage sampling. First, we show a potential weakness of this algorithm and present an alternative algorithm. Then, we perform some analysis on the new algorithm and compare the two algorithms' empirical performances across multiple datasets.

2 Background

This paper builds of off existing literature in unsupervised feature selection and four mathematical subtopics: The Grassmanian space and metrics, Principal Component Analysis, Leverage in the context of regression and subspace projections, and statistical leverage sampling algorithms as used in CUR matrix decompositions. To give a brief overview on how the different components fit together for this paper, we are looking at algorithms that act similar to the linear subspace of best fit to a data matrix, which is a formulation for the principal component vectors. To describe the best fit, we utilize distances between linear subspaces, which are the space of Grassmanians. Leverage provides a way of finding the columns of a matrix that lie near the principal component subspace, and leverage sampling utilizes this to build an unsupervised feature selection algorithm. Finally, the paper builds a new algorithm for doing unsupervised feature selection that selects columns that lie near the principal components using the above mathematical background.

2.1 Notation

For the following paper, we use the upper index notation $X^{(i)}$ to denote the ith column vector of a matrix and the lower index X_i to denote the ith row of a matrix. The character $\phi(k)$ is used to denote some subset of size k along an index. The notation $(:i)$ is used to denote all indexes less than or equal to i.

2.2 Grassmanian Background

2.2.1 Distances Between Subspaces in $G(k,n)$

The Grassmanian space $G(k,n)$is composed of all vector subspaces of dimension k imbedded in a vector space of dimension n. One of the most common ways of measuring the distance between two different subspaces $A, B \in G(k,n)$ is via their shared principal angles. which are defined as:

$$\theta_i = \cos^{-1}(\max_{a_i,b_i} \frac{a_i \cdot b_i}{||a_i||||b_i||}) \text{ for } i \in \mathbb{N} \leqslant k, a_i \in A, b_i \in B \text{ s.t. } a_i \perp a_j, b_i \perp b_j \text{ for } j < i$$

Equivalently, the principal angles can be found by via the SVD of two orthonormal matrices: Let $U\alpha \in A$, $S\beta \in B$ st. $S^TS = I$ and $U^TU = I$ and $\alpha, \beta \in \mathbb{R}^{k\times 1}$ then the principal angles are given by $\cos^{-1}(\Sigma(U^TS))$. A quick proof of this follows:

$$\frac{a_i \cdot b_i}{||a_i||||b_i||} = \frac{(U\alpha_i)^T(S\beta_i)}{||\alpha_i||||\beta_i||}$$

which is maximized when $\alpha_i = A^TB\beta_i$, so the max for any β_i is equivalent to:

$$\frac{\beta_i^T S^T UU^T S\beta_i}{||U^TS\beta_i||} = \frac{\beta_i^T S^T UU^T S\beta_i}{\sqrt{\beta_i^T S^T UU^T S\beta_i}}$$

Which simplifies to a symmetric matrix eigenvalue maximization problem, with orthogonal solutions of eigenvectors $\beta_1..\beta_k$ and eigenvalues of $\lambda_1...\lambda_k$ if the inner products $<\alpha_i, \alpha_j>= 0$:

$$<\alpha_i, \alpha_j>=<U^TS\beta_i, U^TS\beta_j>= \beta_j^T S^T UU^T S\beta_i = \lambda_i < \beta_i, \beta_j >= 0$$

So, $\theta_i = cos^{-1}(\Sigma(U^T S)_i^{(i)})$. In this paper we will be using a metric derived from the principal angles called the chordal distance [9]. The chordal distance is defined as:

$$D_{\text{Chordal}}(A,B) = \sqrt{\sum_{i=1}^{k} \text{Sin}^2(\theta_i)} = \sqrt{k - \sum_{i=i}^{k} \text{Cos}^2(\theta_i)} = \sqrt{k - \text{Trace}(\Sigma(U^T S)^2)} = \sqrt{k - ||U^T S||_F^2}$$

2.2.2 Distances Between Subspaces in $G(k,n), G(r,n)$

The above metric fails however in representing a distance between two subspaces $A \in G(k,n), B \in G(r,n)$ where $r \neq k$. A generalization of the principal angles, and corresponding distances, to subspaces of different dimension is to define the angles by [9]:

$$\theta_1 .. \theta_{\min(\text{k,r})}$$

and the chordal distance as:

$$D^m_{\text{Chordal}}(A,B) = \sqrt{\sum_{i=1}^{\min(k,r)} \text{Sin}^2(\theta_i)}$$

Let $S\alpha \in A, U\beta \in B$, where $\alpha, \in \mathbb{R}^{k\times 1}, \beta, \in \mathbb{R}^{r\times 1}$ then using $\text{Rank}(S^T U) \leqslant \min(\text{Rank}(S), \text{Rank}(U))$, $\Sigma(S^T U)_i^{(i)} = 0 \ \forall i > \min(r,k)$

$$D^m_{\text{Chordal}}(A,B) = \sqrt{\min(k,r) - ||U^T S||_F^2}$$

2.2.3 Inequality

Theorem 2.1 *Now, without loss of generality assume that $r < k$. Then taking B and embedding it in the space of $G(k,n)$ by adding $k-r$ orthonormal columns to S such that $S' = S, W$ and $B' = Span\{S, W\}$ where $S'^T S' = I$*

$$D^m_{Chordal}(A,B) \leqslant D_{Chordal}(A,B')$$

Proof:

$$D_{\text{Chordal}}(A,B') = \sqrt{k - ||U^T S'||_F^2} = \sqrt{r - ||U^T S||_F^2 + (k-r) - ||U^T W||_F^2}$$
$$= \sqrt{D^m_{\text{Chordal}}(A,B)^2 + D^m_{\text{Chordal}}(A,W)^2} \geqslant D^m_{\text{Chordal}}(A,B)$$

For the remainder of the paper we use an abuse of notation to write, for two matrices $M, N \in \mathbb{R}^{k\times n}$,

$$D_{\text{chordal}}(M,N) = D_{\text{chordal}}(\text{Span}\{M\}, \text{Span}\{\text{N}\})$$

and similarly for matrices $M \in \mathbb{R}^{k\times n}$, $N \in \mathbb{R}^{r\times n}$

$$D^m_{\text{chordal}}(M,N) = D_{\text{chordal}}(\text{Span}\{M\}, \text{Span}\{\text{N}\})$$

2.3 PCA background

In unsupervised machine learning, the goal of the machine is to understand the structure of a dataset if no target outputs are presented to the algorithm. One common method used for performing unsupervised learning is Principal Component Analysis (PCA). One description of PCA is that it finds a set of k orthoganal vectors that provide the best projection of the dataset onto a linear subspace minimizing the Frobenius norm. Written into mathematical language, for a centered design matrix $X \in \mathbb{R}^{m\times n}$, and $U^{(:k)} \in \mathbb{R}^{m\times k}$ s.t. $U^{(:k)T} U^{(:k)} = I$:

$$U^{(:k)} = \underset{U^{(:k)}}{\arg\min} \, ||X - U^{(:k)} U^{(:k)T} X||_F$$

Under a simple rearrangement, we can see that this becomes an eigenvalue problem:

$$\begin{aligned}
&\arg\min_{U^{(:k)}} ||X - U^{(:k)}U^{(:k)T}X||_F \\
&= \arg\min_{U^{(:k)}} ||X - U^{(:k)}U^{(:k)T}X||_F^2 \\
&= \arg\max_{U^{(:k)}} ||U^{(:k)T}X||_F^2 \\
&= \arg\max_{U^{(:k)}} \sum_{i=0}^{k} ||U^{(i)T}X||^2 \\
&= \arg\max_{U^{(:k)}} \sum_{i=0}^{k} U^{(i)T}XX^TU^{(i)}
\end{aligned}$$

Since $U^{(:k)T}U^{(:k)} = I$, this implies that $U^{(:k)}$ must correspond to the eigenvectors of XX^T aligning with the largest eigenvalues. One fact that will be used is that the left and right Principal Components can be found via the SVD decomposition of matrix X such that:

$$\text{SVD}(X) = U\Sigma V^T$$

Truncating the SVD to the first k principal components then gives the projection onto the first k principal components in the space of X as:

$$X \approx U^{(:k)}\Sigma^{(:k)}V^T_{(:k)}$$

and the PCs as

$$X \approx U^{(:k)}\Sigma^{(:k)}$$

2.4 PCA Inspired Unsupervised Feature Selection

Despite the statistical interpretability of principal Components, in practice they may not give easily interpreted features for researchers [1] to work with. Instead of giving linear combinations of features as in PCA, a more scientifically interpretable result could be to select a subset of features that give results similar to what doing PCA would. To place this concretely, we could modify the definition of the above minimization criteria of PCA to allow for projections onto columns of X itself:

$$X^{\phi(k)} = \arg\min_{X^{\phi(k)}} ||X - P_{X^{\phi(k)}}X||_F$$

However, this problem is combinatorial with complexity

$$O(X^{\phi(k)}) = \binom{n}{k}$$

Several approximation methods have been presented in the literature for finding features near $X^{\phi(k)}$ [1][2]. One of the most recent in the literature is done by doing a probabilistic sampling from the leverage scores of the design matrix and gives a provable error bound on the selected vectors' distance from the PCs.

2.5 Leverage Background

Leverage has a deep history in linear regression. One of its most common uses is in outlier detection distances in OLS such as cook's distance [3]. A leverage score in the context of linear regression is given by:

$$l_i = X_{(i)}(X^TX)^{-1}X^T_{(i)}$$

where we recall that OLS finds

$$\beta_{\text{ols}} = \arg\min_{\beta} ||y - X\beta||^2 = (X^TX)^{-1}X^Ty$$

$$\hat{y} = Hy$$

$$H = X(X^TX)^{-1}X^T$$

One useful property of the leverage score is that:

$$\frac{\partial \hat{y}_i}{\partial y_i} = l_i$$

Hence, the leverage gives an idea on how much influence a point's value has on the model's prediction of that single point. Using the SVD we can decompose X to write H as:

$$H = UU^T$$

Similar to regression, the solution to the linear subspace of best fit can be written as a kth subspace approximation to H.

$$U^{(:k)} = \arg\min_{U^{(:k)}} ||X - U^{(:k)}U^{(:k)T}X||_F$$
$$\hat{X} = U^{(:k)}U^{(:k)T}X$$

While there is not a clean partial derivative "influence" relationship between the rows of X and the subspace projected model's prediction, $\hat{X}$, like in regression, there is a qualitative similarity. For high leverage scores, a row of $\hat{X}$ will be built almost entirely using the information of the row X_i while limiting the contributions of other rows, such that for $l_i = 1 - \delta$

Theorem 2.2 *For $\hat{X} = (U^{(:k)}U^{(:k)T}X)$ and leverage $l_i = 1 - \delta$*

$$||\hat{X}_i - l_i X_i|| \leqslant \delta(1+\delta)\sqrt{||X||_F^2 - ||X_i||^2}$$

So, similar to leverage giving the influence of a point in OLS, leverage in the context of subspace projections gives a measure of how much a row influences the model's prediction of that row. In addition, a high leverage implies that the row of X it corresponds to lies close to the the singular vector projection space.

Theorem 2.3 *Let σ_i be the ith singular value X*

$$\frac{||X_i V^{(:k)}||}{||X_i||} \leqslant \sqrt{1 - \frac{\delta \sum_{i=k+1}^{n} \sigma_i^2}{||X_i||^2}}$$

So, selecting high leverage rows of a matrix, selects out influential rows that lie near the singular vector space. We can similarly give the leverage of the columns instead by using the transpose of the matrix. In that case, the leverage is defined by:

$$l_i = (V^{(:k)}V^{(:k)T})_i^{(i)}$$

This is the leverage score that will be used and referenced for the rest of the paper.

2.5.1 Proof of Theorem 3.2

$$P = U^{(:k)}U^{(:k)T}$$
$$\implies P = P^T$$
$$P^2 = P^T P = (U^{(:k)}U^{(:k)T})^T U^{(:k)}U^{(:k)T} = P$$
$$\implies P_{ii} = P_{ii}^2 + \sum_{j \neq i} P_{ij}^2$$

Let $l_i = P_{ii} = 1 - \delta$ then:

$$\sum_{j \neq i} P_{ij}^2 = (1-\delta) - (1-\delta)^2 = \delta(1+\delta)$$

Using $\hat{X}_i = \sum_j P_{ij}X_j$

$$\hat{X}_i = (1-\delta)X_i + \sum_{j \neq i} P_{ij}X_j$$
$$||\hat{X}_i - (1-\delta)X_i|| = ||\sum_{j \neq i} P_{ij}X_j|| \leqslant \delta(1+\delta)\sqrt{||X||_F^2 - ||X_i||^2}$$

2.5.2 Proof of Theorem 3.3

$$X_l = \sum_{i=1}^{n} \sigma_i V_i^T U_i^{T(l)}$$

$$||XV^{(k+1:)}|| = \sqrt{\sum_{i=1}^{k} (\sigma_i U_i^{T(l)})^2} \leqslant \sqrt{\left(\sum_{i=k+1}^{n} \sigma_i^2\right)\left(\sum_{i=k+1}^{n} (U_i^{T(l)})^2\right)} \leqslant \sqrt{(1-l)\sum_{i=k+1}^{n} \sigma_i^2}$$

$$\frac{||X_i V||^2}{||X_i||^2} - \frac{||X_i V^{(:k)}||^2}{||X_i||^2} = \frac{||X_i V^{(k+1:)}||^2}{||X_i||^2}$$

$$\implies 1 - \frac{||X_i V^{(:k)}||^2}{||X_i||^2} \leqslant \frac{(1-l_i)\sum_{i=k+1}^{n} \sigma_i^2}{||X_i||^2}$$

$$\implies \frac{||X_i V^{(:k)}||}{||X_i||} \geqslant \sqrt{1 - \frac{(1-l_i)\sum_{i=k+1}^{n} \sigma_i^2}{||X_i||^2}}$$

2.6 Leverage Sampling Algorithm

The algorithm presented by Drineas et. all [1] goes as follows: Select a column with replacement by the probability distribution over the columns where

$$P(\text{column i}) = l_i/k$$

Where the selected columns form a scaled matrix C of a subset of the features of X. Then for $c = O(k\log k\log 1/\delta/\epsilon^2)$ with probability at least $1-\delta$

$$||X - CC^+X|| \leqslant (1+\epsilon)||X - U^{(:k)}U^{(:k)T}X||$$

So, we sample from the "outliers" of the columns that nearly lie in the left singular vector space. This algorithm is highly efficient, and operates in O(SVD(X)) time.
In this paper, to give a fair comparison between this algorithm and a new one, we will introduce the hyperparamter q. We will select q sets of columns using the leverage sampling technique and find the columns set that minimizes $D_{\text{Chordal}}(U^{(:k)}, X^{c(k)})$ to give a final output.

3 A New Algorithm

3.1 Motivation

The motivation for looking at an alternative algorithm for selecting features stems from a perceived shortfall of the above algorithm. The main motivation is that leverage is sensitive to the variance of a feature, which may cause it to overweight some features that may be further away from the singular subspace than others. A proof of the leverage sensitivity to the column norm is as follows:
Let X be a centered design matrix and X' be X with its first column multiplied by a scaling factor α. l_1 denotes the leverage of the first column of X and l_1' denotes the leverage of the first column of X'

$$X' = (\alpha X^{(1)}, X^{(2:)}), l_1 = X^{(1)T}(XX^T)^{-1}X^{(1)}, l_1' = X'^{(1)T}(X'X'^T)^{-1}X'^{(1)}$$

$$X'X'^T = (\alpha - 1)X^{(1)}X^{(1)T} + XX^T$$

Using the Sherman-Morrison inversion formula,

$$(X'X'^T)^{-1} = (XX^T)^{-1} - \frac{(XX^T)^{-1}X^{(1)}X^{(1)T}(XX^T)^{-1}(\alpha-1)^2}{1 + X^{(1)T}(XX^T)^{-1}X^{(1)}(\alpha-1)^2}$$

$$\implies l_1' = \alpha^2 l - \frac{l_1^2(\alpha-1)^2\alpha^2}{1 + l_1(\alpha-1)^2}$$

$$= \frac{\alpha^2 l_1}{1 + l_1(\alpha-1)^2}$$

Which taking the limit:

$$\text{Lim}_{\alpha\to\infty}(l_1') = 1$$

So, as the norm increases the total leverage of a column approaches 1. It is important to note that this proof applies only to the leverage as a whole, and does not explicitly say anything about a k approximation's leverage. However, as the leverage increases, it will be applied amidst the leverage from each left singular vector, so it will be observed in some k approximation. Now, consider what would happen if there were features with large norms amidst small norm features. If the large norms increase the leverage in the k approximation, then the small norm features would rarely be selected by the leverage selection algorithm. As an example to demonstrate this, consider:

$$M = \begin{bmatrix} -3 & -6.3 & -.106 \\ 0 & 4.67 & -.65 \\ 3 & 1.66 & .75 \end{bmatrix}$$

A numerical calculation of the top left singular vector via the SVD yields that:

$$U^{(1)} \approx \begin{bmatrix} -.107 \\ -.647 \\ -.754 \end{bmatrix}$$

Using the distance:

$$d_1(X^{(j)T}, U^{(1)}) = 1 - |cos(\theta)|^2 = 1 - (\frac{||X^{(j)T}U^{(1)}||}{||X^{(j)T}||})^2$$

we get:

$$S \approx \begin{bmatrix} 0.63 & 0.98 & 1 \times 10^{-5} \end{bmatrix}$$

So, if we cared about getting the column vector of this matrix that most aligns with $U^{(1)}$, we should choose column 3. A calculation of the leverage score for each however gives:

$$L \approx \begin{bmatrix} .15 & .85 & 2 \times 10^{-7} \end{bmatrix}$$

So, the third column would almost never be selected by the leverage sampling algorithm even though it appears to be the best. As a counterpoint, this example may seem contrived since getting a feature that doesn't impact the location of the singular vectors, yet lies very close to them sounds improbable. However, in the case of high features but low observations where the column space could be efficiently explored by the number of features, this may not be far fetched.

3.2 Orthogonal Subspace Algorithm

The new algorithm we will be demonstrating takes this potential L2 norm discrepancy into account. It goes as follows:

Algorithm 1 Subspace Feature Selection $O(q(k^2m^2 + knm) + SVD(X))$

Require: $X \in \mathbb{R}^{m \times n}$
Require: k
Require: q
  $U^{(:k)}$, = LeftSingularVectorsFromTruncatedSVD(X)
  for i = 1..c **do**
    M_i, Δ_i = OrthogonalSubspaceAlg($X, k, U^{(:k)}$)
  end for
  BestIndex = $\arg\min_i (D_{\text{Proj}}(U^{(:k)}, X^{M_i(k)}))$
  return $(D_{\text{Proj}}(U^{(:k)}, X^{M_{\text{BestIndex}}(k)}))$,$M_{\text{BestIndex}}$,$\Delta_{\text{BestIndex}}$

The algorithm **Orthogonal Subspace** works by finding the closest feature to the subspace given by the left singular vectors as defined by d_1 above. Then from this starting point, it finds the next feature which points the greatest amount into the subspace given by the left singular vectors subtracting out the direction that the first vector pointed in the subspace. This continues, up to k, where the directions pointed by each previous vector into the subspace are subtracted out and then maximum feature pointing into the remaining subspace is selected. The motivation for building this algorithm is that it ensures that the features have a high degree of orthogonality as well as lie close to the singular vector subspace. In turn, this ensures that the subspace built by the selected features is near to that of the singular vector subspace. The vector of m_n returned is the set of indexes and δ_n is the euclidian distance from the remaining subspace when it was selected.
The **Subspace** algorithm works by running **Orthogonal Subspace** repeatedly, choosing the ith closest vector to the space as a starting point and then running the algorithm as detailed above. After collecting all the selected subsets, it then chooses the best set that minimizes the projection distance and returns it.

Algorithm 2 Orthogonal Subspace Alg $O(k^2m^2 + knm)$

Require: $X \in \mathbb{R}^{m \times n}$
Require: k
Require: q
Require: $U^{(:k)}$ from TrucatedSVD(X)
  $X^{(n)} = X^{(n)}/||X^{(n)}||$
  $\epsilon_n = ||U^{(:k)T} X^{(n)}||^2$
  for $i = 1..k$ **do**
    if $i > 1$ **then**
      $m_i \leftarrow \arg\min_i \epsilon_n$
    else
      $m_i \leftarrow q^{th}$ least ϵ_n
    end if
    $p_i = U^{(:k)} U^{(:k)T} X^{(m)}$
    if $i > 1$ **then**
      for $j = 1..(i-1)$ **do**
        $p_i \leftarrow p_i - (p_i \dot{b}_j) b_j$
      end for
    end if
    $b_i \leftarrow p_i/||p_i||$
    $\epsilon_n = \epsilon_n - (b_i \dot{X}^{(n)})^2$
    $\delta_i \leftarrow \sqrt{\epsilon_i}$
    Delete(ϵ_i)
  end for
  return m_n, δ_n

4 Orthogonal Subspace Error

In this section we construct a formal proof on both the Frobenius projection distance and chordal distance between the subspace selected by Algorithm 2 and that of the left k singular vectors. Let Δ be the diagonal matrix with elements given by the error terms returned by the algorithm, δ_n.

Theorem 4.1 *Let k be the number of top left singular vectors chosen, $X^{\phi(k)}$ be the column subset chosen by **Orthogonal Subspace Alg**, and Δ be the diagonal matrix with elements given by the error terms returned by the algorithm, δ_n. Assume that $X^{\phi(k)}$ has linearly independent columns. Then:*

$$D_{chordal}(U^{(:k)}, X^{\phi(k)}) \leqslant \sqrt{k - \frac{k - ||\Delta||_F^2}{(1 + 2k||\Delta||_F)^2}}$$

Corollary 4.1.1 *Under the same assumptions as **Theorem 4.1** except with $X^{\phi(k)}$ under no linear independence assumption. Let $r = Rank(X^{\phi(k)})$*

$$D^m_{chordal}(U^{(:k)}, X^{\phi(k)}) \leqslant \sqrt{k - \frac{k - ||\Delta||_F^2}{(1 + 2k||\Delta||_F)^2}}$$

Corollary 4.1.2 *Let $X^{\phi(k)+}$ denote the Moore-Penrose pseudoinverse of $X^{\phi(k)}$*

$$D_{proj}(X, X^{\phi(k)}) = ||X - X^{\phi(k)} X^{\phi(k)+} X^{\phi(k)}||_F \leqslant ||X - U^{(:k)} U^{(:k)T} X||_F + \left(\sqrt{2}\sqrt{k - \frac{k - ||\Delta||_F^2}{(1 + 2k||\Delta||_F)^2}} + \sqrt{k - r}\right) ||X||_F$$

Both chordal distance bounds have the property that as the error $||\Delta||_F$ approaches 0, the distance from the singular vector subspace also approaches 0. This implies that the algorithm is useful in minimizing the chordal distance assuming it can select out small error terms successfully. In the case of the distance by projection onto the pseudoinverse, the error term shares this same quality if the Rank($X^{\phi(k)}$) = k. Since the algorithm cannot select the same column twice, the only source of rank deficiency would be from shared linear dependance between previously selected columns and future ones. Since the algorithm attempts to choose columns that lie orthogonal to eachother, this deficiency should be small, however proving this seems difficult without making strong assumptions on the distribution of the data columns.

4.0.1 Proof of Corollary 4.1.1

This follows directly from **Theorem 2.1** and **Theorem 4.1**. Adding a span of $r-k$ orthonormal columns to the space spanned by $U'^{(:r)}$ gives the space $U'^{(:k)}$ so:

$$D^m_{\text{chordal}}(U^{(:k)}, X^{\phi(k)}) = D^m_{\text{chordal}}(U^{(:k)}, U'^{(:r)}) \leqslant D_{\text{chordal}}(U^{(:k)}, U'^{(:k)}) \leqslant \sqrt{\frac{k - ||\Delta||_F^2}{(1+2k||\Delta||_F)^2}}$$

4.0.2 Proof of Corollary 4.1.2

Let $U', \Sigma', V' = \text{SVD}(X)$ and $U', \Sigma', V' = \text{SVD}(X^{\phi(k)})$ for k $\leqslant$ Rank(X) and $r = \text{Rank}(X^{\phi(k)})$:

$$\begin{aligned}
||X - X^{\phi(k)}X^{\phi(k)+}X^{\phi(k)}||_F &= ||X - U'\Sigma V'^T V'\Sigma'^+ U'^T X||_F = ||X - U'^{(:r)}U'^{(:r)T}X||_F \\
&= ||X - U^{(:k)}U^{(:k)T}X + U'^{(:r)}U'^{(:r)T}X + U^{(:k)}U^{(:k)T}X||_F \\
&\leqslant ||X - U^{(:k)}U^{(:k)T}X||_F + ||U'^{(:r)}U'^{(:r)T} - U^{(:k)}U^{(:k)T}||_F||X||_F
\end{aligned}$$

By definition,

$$\begin{aligned}
&||U'^{(:r)}U'^{(:r)T} - U^{(:k)}U^{(:k)T}||_F = \sqrt{\text{Trace}((U'^{(:r)}U'^{(:r)T} - U^{(:k)}U^{(:k)T})^T(U'^{(:r)}U'^{(:r)T} - U^{(:k)}U^{(:k)T}))} \\
&= \sqrt{\text{Trace}(U'^{(:r)}U'^{(:r)T}) - \text{Trace}(U^{(:k)}U^{(:k)T}) - \text{Trace}((U^{(:k)T}U^{(:r)})^T(U^{(:k)T}U^{(:r)}))} = \sqrt{k + r - 2||U^{(:k)T}U^{(:r)}||_F^2} \\
&\leqslant \sqrt{2}\sqrt{k - ||U^{(:k)T}U^{(:r)}||_F^2} + \sqrt{k-r} = \sqrt{2}D^m_{\text{chordal}}(X^{\phi(k)}, U^{(:k)}) + \sqrt{k-r}
\end{aligned}$$

Hence, by the above result and **Corollary 4.1.1**, we get

$$\begin{aligned}
||X - X^{\phi(k)}X^{\phi(k)+}X^{\phi(k)}||_F &\leqslant ||X - U^{(:k)}U^{(:k)T}X||_F + ||U'^{(:r)}U'^{(:r)T} - U^{(:k)}U^{(:k)T}||_F||X||_F \\
&\leqslant ||X - U^{(:k)}U^{(:k)T}X||_F + (\sqrt{2}D^m_{\text{chordal}}(X^{\phi(k)}, U^{(:k)}) + \sqrt{k-r})||X||_F \\
&\leqslant ||X - U^{(:k)}U^{(:k)T}X||_F + \left(\sqrt{2}\sqrt{k - \frac{k - ||\Delta||_F^2}{(1+2k||\Delta||_F)^2}} + \sqrt{k-r}\right)||X||_F
\end{aligned}$$

4.1 Proof of Theorem 4.1

We begin by showing a small connected set of theorems (proved below) for $S^{(:k)}B = X^{\phi(k)}$ s.t. $S^{(:k)T}S^{(:k)} = \text{I}$, and $\Sigma(B), \Sigma(X^{\phi(k)})$ denoting the singular values of $X^{\phi(k)}$ and B respectively.

Theorem 4.2

$$||U^{(:k)T}X^{\phi(k)}||_F^2 = ||U^{(:k)T}S^{(:k)}B||_F^2 \leqslant ||U^{(:k)}TS^{(:k)}||_F^2(1 + ||\Sigma(B) - I||_F)^2$$

Theorem 4.3

$$k - ||\Delta||_F^2 \leqslant ||U^{(:k)T}S^{(:k)}B||_F^2$$

Theorem 4.4

$$||I - \Sigma(X^{\phi(k)})||_F \leqslant \sqrt{2k}||\Delta||_F$$

Theorem 4.5

$$\Sigma(B) = \Sigma(X^{\phi(k)})$$

Combining these together yields:

$$\begin{aligned}
k - ||\Delta||_F^2 &\leqslant ||U^{(:k)T}S^{(:k)}B||_F^2 \\
&\leqslant ||U^{(:k)T}S^{(:k)}||_F^2(1 + ||\Sigma(B) - I||_F)^2 \\
&\leqslant ||U^{(:k)T}S^{(:k)}||_F^2(1 + \sqrt{2k}||\Delta||_F)^2 \\
\implies \frac{k - ||\Delta||_F^2}{(1+2k||\Delta||_F)^2} &\leqslant ||U^{(:k)T}S^{(:k)}||_F^2
\end{aligned}$$

From the definition of the chordal distance:

$$D_{\text{Chordal}}(X^{\phi(k)}, U^{(:k)}) = \sqrt{k - ||U^{(:k)T}S||_F^2} \leqslant \sqrt{k - \frac{k - ||\Delta||_F^2}{(1+2k||\Delta||_F)^2}}$$

4.1.1 Proof of Theorem 4.2

Consider the SVD decomposition of $X^{\phi(k)} = S^{(:k)}\Sigma^{(:k)}W^T$ with $B = \Sigma^{(:k)}W^T \in \mathbb{R}^{k\times k}$. Let R denote a rotation matrix and M denote a generic $\mathbb{R}^{k\times k}$ matrix.

$$||U^{(:k)T}X^{\phi(k)}||_F = ||U^{(:k)T}S^{(:k)}B||_F = ||U^{(:k)T}S^{(:k)}(R+M)||_F$$
$$\leqslant ||U^{(:k)T}S^{(:k)}R||_F + ||U^{(:k)T}S^{(:k)}M||_F \leqslant ||U^{(:k)T}S^{(:k)}||_F + ||U^{(:k)T}S^{(:k)}||_F||M||_F$$

Then since $B = R + M$ we get that: $||B - R||_F = ||M||_F$
Now, using the fact that $||B - R||_F$ is minimized over orthonormal R for a square B by $R = UV^T$ (from SVD)[6] with error $||\Sigma(B) - I||_F$ the least upper bound on $||M||_F$ gives:

$$||U^{(:k)T}S^{(:k)}||_F + ||U^{(:k)T}S^{(:k)}||_F||\Sigma(B) - I||_F = ||U^{(:k)T}S^{(:k)}||_F(1 + ||\Sigma(B) - I||_F)$$

Thus:

$$||U^{(:k)T}X^{\phi}(k)||_F^2 = ||U^{(:k)T}S^{(:k)}B||_F^2 \leqslant ||U^{(:k)T}S^{(:k)}||_F^2(1 + ||\Sigma(B) - I||_F)^2$$

4.1.2 Proof of Theorem 4.3

For rotation matrix $R \in \mathbb{R}^{m\times m}$

$$||U^{(:k)T}S^{(:k)}B||_F^2 = ||RU^{(:k)T}S^{(:k)}B||_F^2 = ||(U^{(:k)}R^T)^TS^{(:k)}B||_F^2$$

Let $P = R^T$ then P is also a rotation matrix

$$||(U^{(:k)}P)^TS^{(:k)}B||_F^2 = ||(U^{(:k)}P^{(:k)})^TS^{(:k)}B||_F^2 + ||(U^{(:k)}P^{(k:)})^TS^{(:k)}B||_F^2 \geqslant ||(U^{(:k)}P^{(k:)})^TS^{(:k)}B||_F^2$$

Through definition of the error terms for a specific P determined implicitly in running the algorithm

$$k - ||\Delta||_F^2 = ||(U^{(:k)}P^{(k:)})^TS^{(:k)}B||_F^2 \leqslant \sum_{i=1}^{k}||U^{(:k)T}S^{(:k)}B^{(i)}||_F^2 = ||(U^{(:k)}P^{(k:)})^TS^{(:k)}B||_F^2$$

4.1.3 Proof of Theorem 4.4

Let W be the direction of the error terms from the singular vector subspace when they are selected out, and R be the rotation matrix inherently chosen by the algorithm as nearest to each selected point. $(UR)^{(i)} \perp W^{(i)}$

$$X = UR(I - \Delta) + W\Delta = UR + (W - UR)\Delta$$

Using $\sqrt{\sum_i^k(\sigma_i - \tilde{\sigma_i})^2} \leqslant ||E||_F$ for matrix A with singular values $\sigma_i(A)$ and the perturbation $A + E$ with singular values $\tilde{\sigma}_i(A + E)$. [4]

$$\sigma_i(UR) = 1$$
$$\sqrt{\sum_i^k(1 - \sigma_i(\tilde{X}))^2} = ||I - \Sigma(X)||_F \leqslant ||(W - UR)\Delta|| \leqslant ||W - UR||_F||\Delta||_F$$

Using the fact that $(UR)^{(i)} \perp W^{(i)}$

$$||W - UR||_F||\Delta||_F = \sqrt{2k}||\Delta||_F$$

So all together:

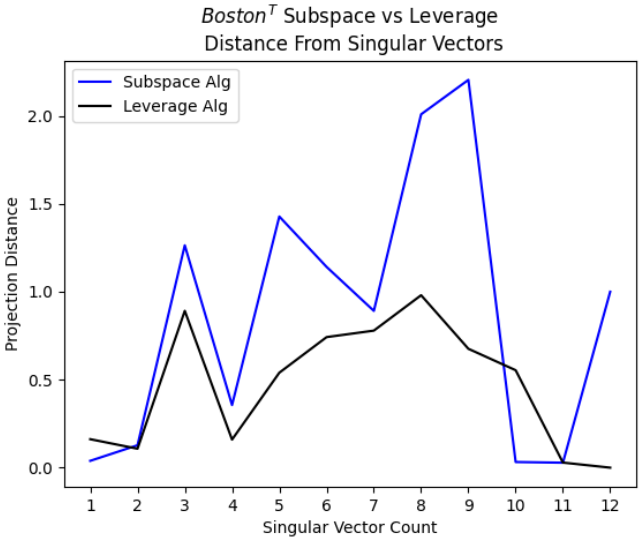
$$||I - \Sigma(X)||_F \leqslant \sqrt{2k}||\Delta||_F$$

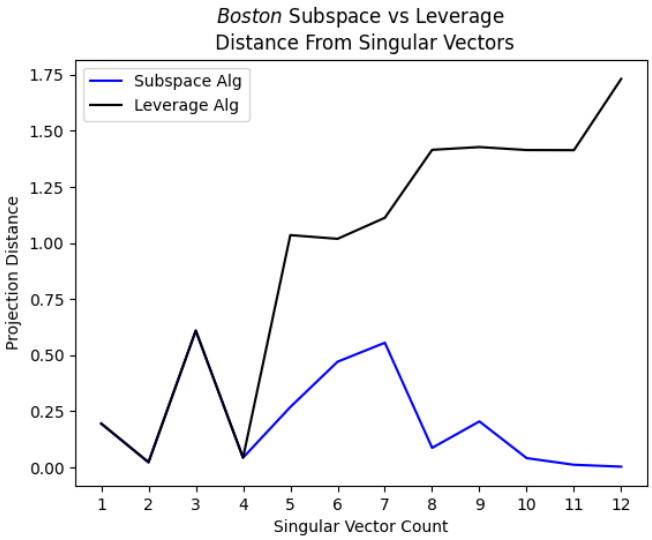
4.1.4 Proof of Theorem 4.5

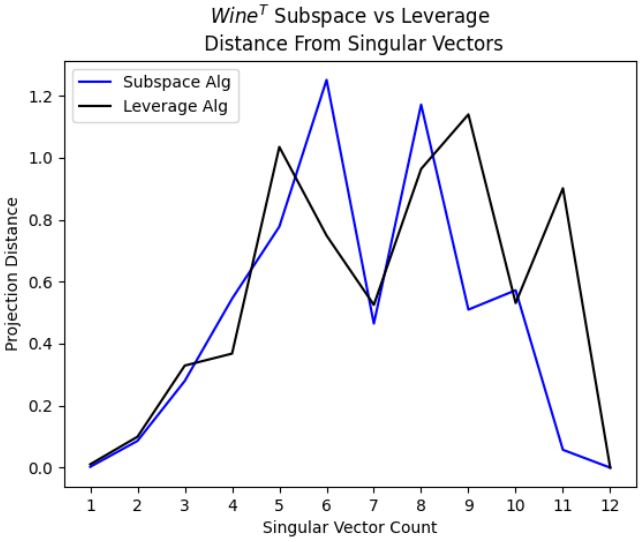
Since SVD$(X^{\phi(k)}) = S\Sigma V^T = SB$, SVD$(B)$ can be written as $I\Sigma V^T$. So, $\Sigma(X^{(\phi(k)}) = \Sigma(B)$

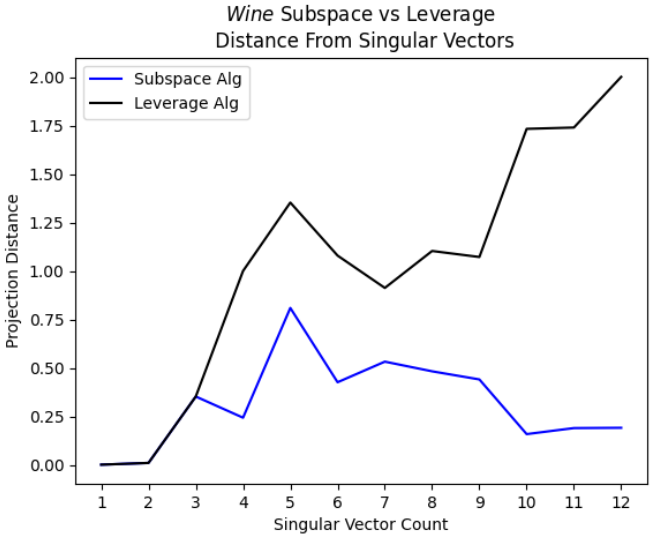
5 Empirical Results

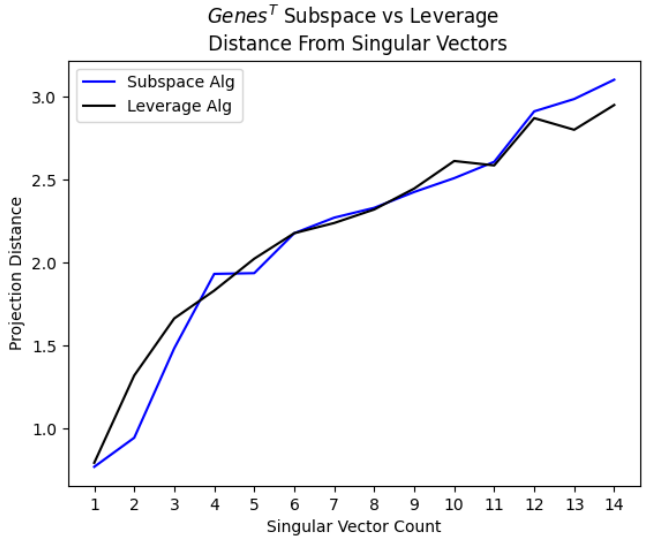
Here we examine the performance of both the leverage sampling algorithm and Algorithm 1. Both algorithms were evaluated on three different datasets. The first is a commonly used dataset from Scikit-Learn on the housing market in Boston (13×506) [8], the second (also from Scikit-Learn) is a dataset on wine classification (13×178) [8], the third (from Kaggle) is a subset of a genetics dataset for cancer classification (7129×34) [7]. For all datasets, please reference the attached commented python code for which manipulations were applied to the data. In all cases, we will be investigating both the transpose and original data to see if there is any difference between algorithm performances on wide vs tall matricies. In all cases, the datasets are centered along the feature means before being analyzed in both algorithms.

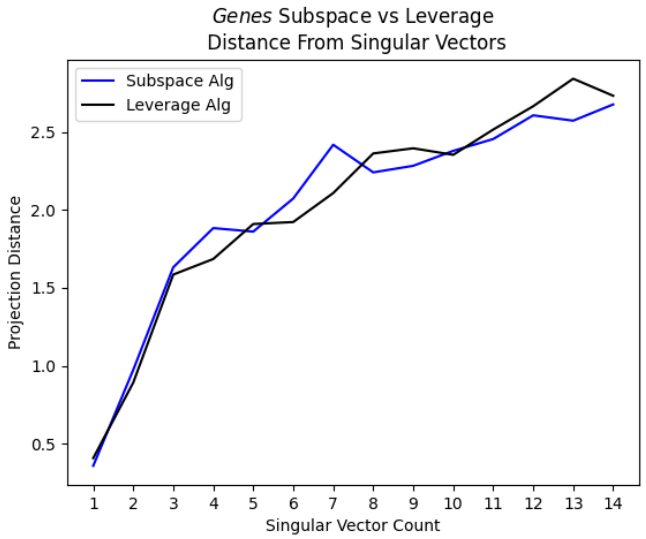












The Hyperparameters and results for each of the datasets are the following:

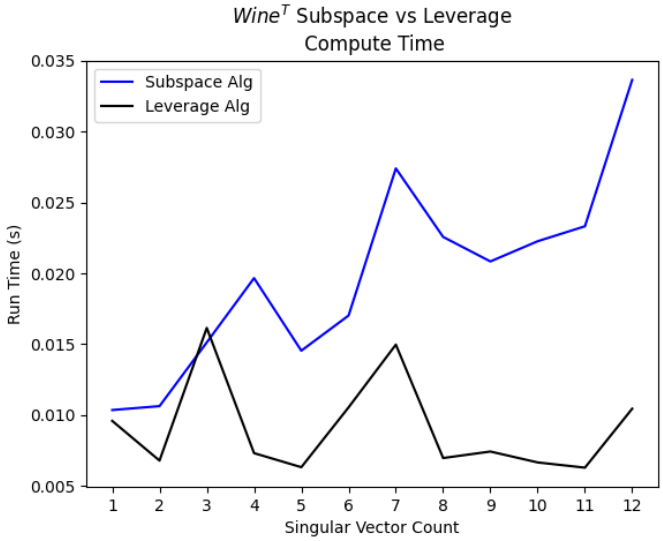
Dataset \ Results	q	k	ΔD_{Proj}	O_{L2}
Boston^T	10	13	4.89	230.6
Boston	10	13	-8.92	3185.9
Wine^T	10	13	-0.93	891.5
Wine	10	13	-8.53	3841.3
Genes^T	10	15	0.03	20889.0
Genes	10	15	-0.25	102925.5

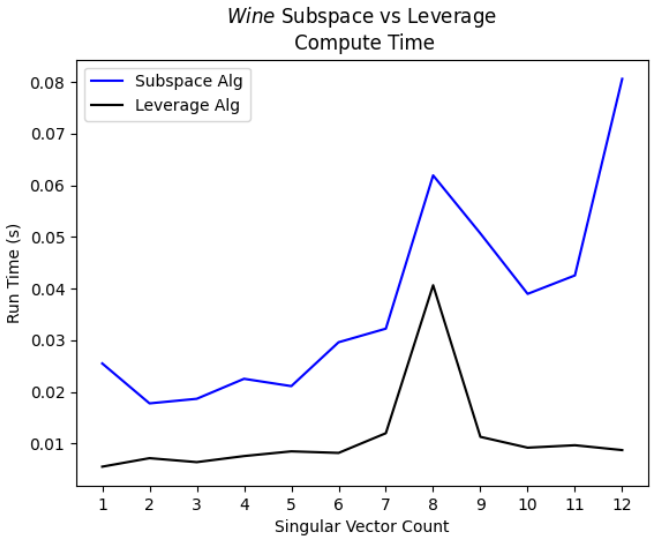
Let $X^{l(i)}$ be the leverage selected columns and $X^{s(i)}$ be the columns selected by the subspace algorithm. Then we define:

$$\Delta D_{\text{Proj}} = \sum_{i=1}^{k} D_{\text{Proj}}(U^{(:k)}, X^{l(i)}) - D_{\text{Proj}}(U^{(:k)}, X^{s(i)})$$

$$O_{L2} = \max\{||X^{(i)}|| \text{s.t.} i \in 1..n\} - \frac{\sum_{i=1}^{n} ||X^{(i)}||}{n}$$

So, a negative ΔD_{Proj} implies that the subspace algorithm is closer to $U^{(:k)}$ than the leverage algorithm. O_{L2} gives the size of the largest $L2$ norm outlier from the mean of the $L2$ norm of all the features in the dataset.
As we can see, the subspace algorithm does seem to be generally outperforming or acting on par with the leverage algorithm except in the case of Boston^T. In some cases the difference appears to be quite small, like in the case of Genes and Genes^T. However, in some cases it is quite drastic, such as the Boston and Wine datasets. Unfortunately, for large k, the subspace algorithm suffers from its quadratic complexity in k.





From our earlier theoretical observation that the leverage increases when there is high discrepancy in the L2 norm, we might

expect that both genetic datasets would benefit from the use of the Subspace algorithm. However, we see both of the algorithms performing very similarly. Something that is interesting is that in both the other datasets, our predictions did hold. Both Boston^T and Wine^T had low values of O_{L2} and in both cases, the Subspace algorithm performed marginally better or worse than the leverage algorithm. However, in Boston and Wine where O_{L2} was large, the leverage algorithm was soundly beat by the Subspace algorithm. Strangely, both of these were in the case of tall matricies, where we would expect the column space to not be explored as fully by the features.

6 Discussion

In this paper a new algorithm, Subspace, was developed and tested against an existing leverage sampling algorithm. We gave an error bound on the chordal and frobenius norm subspace distance from the singular vector subspace. However, something that was notably lacking in this analysis was a statistical guarantee on the error of the algorithm, even under extremely strong generating assumptions on the data. One reason for this is that the dependence of the singular vectors on the distribution of the features is quite complicated. In addition, unlike the leverage algorithm, the value of each feature being selected by Subspace is in sequence depends on the previous selected feature. This means that even if we could get the distribution of the singular vectors from the distribution on the columns, there would be a difficult, recursive conditional probability distribution we would need to compute. Both of these are not something I was able to figure out how to do, and remain an area for future investigation.

One remaining interesting question is how do we statistically interpret the features selected out by the Subspace algorithm? To my knowledge, this doesn't have a great answer. While a biologist working on genes for example may like the simplicity of dealing with discrete yes or no questions on whether a feature is important or not, doing so creates less statistical interpretation than PCA. PCA for example has the nice property that the principal components point along the direction of greatest variance while a feature selection algorithm does not. Leverage scores give a measure on how much a column was used to construct the singular vector space, which retains some interpretation from PCA. The motivation for the Subspace algorithm is that it chooses features that behave like principal Components, which doesn't easily lend itself to a clean statistical interpretation. However, it could probably be expected that the performance of algorithms that use the Subspace algorithm would behave similarly to using PC features.

References

[1] Drineas, Petros, Michael W. Mahoney, and Shan Muthukrishnan. "Relative-error CUR matrix decompositions." SIAM Journal on Matrix Analysis and Applications 30.2 (2008): 844-881.

[2] Petros Drineas, Ravi Kannan, and Michael W Mahoney. Fast monte carlo algorithms for matrices iii: Computing a compressed approximate matrix decomposition. SIAM Journal on Computing, 36(1):184–206, 2006.

[3] Weisberg, Sanford, Applied Linear Regression, New York, John Wiley Sons, 1985.

[4] L. MIRSKY, SYMMETRIC GAUGE FUNCTIONS AND UNITARILY INVARIANT NORMS, The Quarterly Journal of Mathematics, Volume 11, Issue 1, 1960, Pages 50–59, https://doi.org/10.1093/qmath/11.1.50

[5] Cai, T. Tony, and Anru Zhang. "Rate-optimal perturbation bounds for singular subspaces with applications to high-dimensional statistics." The Annals of Statistics 46.1 (2018): 60-89.

[6] Sarabandi, Soheil, et al. "On closed-form formulas for the 3-d nearest rotation matrix problem." IEEE Transactions on Robotics 36.4 (2020): 1333-1339.

[7] Golub TR, Slonim DK, Tamayo P, Huard C, Gaasenbeek M, Mesirov JP, Coller H, Loh ML, Downing JR, Caligiuri MA, Bloomfield CD, Lander ES. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science. 1999 Oct 15;286(5439):531-7. doi: 10.1126/science.286.5439.531. PMID: 10521349.

[8] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[9] Ye, Ke, and Lek-Heng Lim. "Schubert varieties and distances between subspaces of different dimensions." SIAM Journal on Matrix Analysis and Applications 37.3 (2016): 1176-1197.