EECS 16B Designing Information Systems and Devices II UC Berkeley Spring 2024

Note 16: Low-Rank Approximation and Principal Component Analysis

1 Overview and Motivation

In this note, we will examine two important, applications of the singular value decomposition (SVD).

The first application is using *low rank approximations* for dimensionality reduction of data.

Key Idea 1 (Low-Rank Approximation)
Low-rank approximation of a matrix $A \in \mathbb{R}^{m \times n}$ with rank r is the process of finding another matrix $A_\ell \in \mathbb{R}^{m \times n}$ with rank $\ell \ll r$ such that $A - A_\ell$ is "small" in some sense.

Our second application is a tool for data analysis. When we collect data, there are a lots of factors that influence the values we measure. Our goal is to figure out the most important ones. This allows us to compress our data removing the dimensions("factors") that are not important. We do this via *principal component analysis* (PCA).

Key Idea 2 (Principal Component Analysis)
Principal component analysis is a way of capturing the most important dimensions of data.

2 Low-Rank Approximation

Given a matrix $A \in \mathbb{R}^{m \times n}$ of rank $r \leq \min\{m, n\}$, we saw in Note 15 that we can write A using the outer-product form of the SVD:

$$A = \sum_{i=1}^{r} \sigma_i \vec{u}_i \vec{v}_i^\top. \tag{1}$$

If our matrix is high-rank, i.e., $r \approx \min\{m, n\}$, then almost all the σ_i will be nonzero and non-negligible. However, if the data has some linear, low-rank structure, as is usually the case with real data such as images, most of our singular values will be very small (but usually nonzero due to noise or disturbances). If, say, the data has intrinsic linear rank ℓ, then the first ℓ singular values are large, and the remaining $r - \ell$ are small:

$$A = \sum_{i=1}^{r} \sigma_i \vec{u}_i \vec{v}_i^\top = \sum_{i=1}^{\ell} \sigma_i \vec{u}_i \vec{v}_i^\top + \sum_{i=\ell+1}^{r} \underbrace{\sigma_i}_{\approx 0} \vec{u}_i \vec{v}_i^\top \approx \sum_{i=1}^{\ell} \sigma_i \vec{u}_i \vec{v}_i^\top. \tag{2}$$

This motivates approximating the data as

$$A_\ell := \sum_{i=1}^{\ell} \sigma_i \vec{u}_i \vec{v}_i^\top \tag{3}$$

and using this compressed data for further analysis.

For notation's sake, if we define

$$U_\ell := \begin{bmatrix} \vec{u}_1 & \cdots & \vec{u}_\ell \end{bmatrix} \qquad V_\ell := \begin{bmatrix} \vec{v}_1 & \cdots & \vec{v}_\ell \end{bmatrix} \qquad \Sigma_\ell := \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_\ell \end{bmatrix} \tag{4}$$

then $A_\ell = U_\ell \Sigma_\ell V_\ell^\top$. Under this notation $A_r = U_r \Sigma_r V_r^\top$ is the compact SVD of A, so $A_r = A$.

The approximation of A by A_ℓ is justified by the *Eckart-Young-Mirsky theorem* (sometimes just *Eckart-Young theorem*), which says that this is the best rank-ℓ approximation in terms of the Frobenius norm.

Theorem 3 (Eckart-Young-Mirsky Theorem)
Let $A \in \mathbb{R}^{m\times n}$ have rank $r \leq \min\{m, n\}$. For $\ell \leq r$ and A_ℓ as defined above, we have that

$$A_\ell \in \operatorname*{argmin}_{B\in\mathbb{R}^{m\times n}} \quad \|A - B\|_F^2 \tag{5}$$

$$\text{s.t.} \quad \operatorname{rank}(B) = \ell. \tag{6}$$

See Appendix A for the proof. Note that the exact proof of this theorem is out of scope. The theorem is also true if we use the *relaxed* constraint $\operatorname{rank}(B) \leq \ell$, but the proof is harder and out of scope.

3 Principal Component Analysis

Suppose we have collected some noisy data points, which are represented as vectors $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^d$. Let

$$A := \begin{bmatrix} \vec{x}_1 & \cdots & \vec{x}_n \end{bmatrix}, \tag{7}$$

be the so-called *data matrix*, i.e., arranging the data points as the *columns* of A.

The first step in PCA is always to center the data around its mean $\frac{1}{n} \cdot A\vec{1}_n$. That is, we replace A, our uncentered data matrix, with $D := A(I_n - \frac{1}{n}\vec{1}_n\vec{1}_n^\top)$. More concretely, we subtract the column mean from each data point in A(note: the symbol $\vec{1}_n \in \mathbf{R}^{n\times 1}$ is a vector with all elements being 1).

Suppose $\operatorname{rank}(D) = r \leq \min\{n, d\}$. Further suppose the so-called "*ground truth*" data (i.e., data without the noise) actually would span a ℓ-dimensional subspace S_{gt} of $\mathbb{R}^d$, with $\ell \leq r$. The remaining $r - \ell$ dimensions, in this case, would be the product of noise. The goal of principal component analysis (PCA) is to find, from the noisy data, the relevant ℓ-dimensional subspace S_{gt} which the dataset spans.

One general approach to solving problems of this type is:

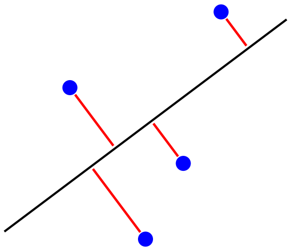
1. Make an *objective function* which quantifies the property you want to optimize.
2. Optimize the objective function.

This seems very abstract. For our problem, we want to estimate the "best" ℓ-dimensional subspace that our ground truth data would span. We would then ask what the meaning of "best" is; a quantifiable notion of "best subspace" is "the one closest to all the points". From here we can develop an objective function that we can minimize. The objective function would take in a subspace S, and evaluate how close it is to all the points; it would be the following:

$$\sum_{i=1}^{n} \|\vec{x}_i - \operatorname{proj}_S(\vec{x}_i)\|^2. \tag{8}$$

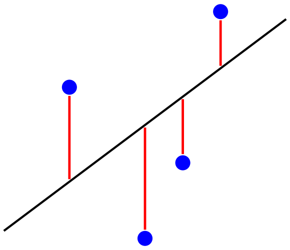
For some intuition, if $n = 4$ and $d = 2$ (i.e., our data set is 4 points in $\mathbb{R}^2$), this objective function would be the sum of the squared lengths of the red lines in this picture, where the subspace S is a black line and the data points $\vec{x}_1, \ldots, \vec{x}_4$ are blue dots:

Figure 1: Objective function visualization for PCA.



Contrast this to the least squares objective function, which computes the sum of squares of the *vertical* residuals (instead of the orthogonal residuals):

Figure 2: Objective function visualization for least squares.



Now, we explore how to compute this objective function. If $W \in \mathbb{R}^{d\times\ell}$ has orthonormal columns which span S, i.e., $W^\top W = I_\ell$ and $\mathrm{Col}(W) = S$, then

$$\mathrm{proj}_S(\vec{x}_i) = WW^\top \vec{x}_i. \tag{9}$$

The above equation follows directly from the least squares derivation of a projection matrix in the case in which the columns of this projection matrix are orthonormal.

$$Q(Q^\top Q)^{-1}Q^\top = QQ^\top \tag{10}$$

which means that the matrix $QQ^\top$ projects a vector onto $\mathrm{Col}(Q)$.

This means that our objective function takes the form

$$\sum_{i=1}^{n} \left\| \vec{x}_i - WW^\top \vec{x}_i \right\|^2. \tag{11}$$

And we would try to minimize this over subspaces S, i.e., over matrices $W \in \mathbb{R}^{d\times\ell}$ with orthonormal columns such that $W^\top W = I_\ell$. This leads to the following optimization problem, which can be solved via the SVD.

Theorem 4 (Principal Component Analysis)
Let $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^d$ be data points. If $A = \begin{bmatrix} \vec{x}_1 & \cdots & \vec{x}_n \end{bmatrix}$ is the data matrix, with SVD $A = U\Sigma V^\top$, then

$$U_\ell \in \operatorname*{argmin}_{W \in \mathbb{R}^{d \times \ell}} \sum_{i=1}^{n} \left\| \vec{x}_i - WW^\top \vec{x}_i \right\|^2 \tag{12}$$

$$\text{s.t.} \quad W^\top W = I_\ell. \tag{13}$$

where $U_\ell = \begin{bmatrix} \vec{u}_1 & \cdots & \vec{u}_\ell \end{bmatrix}$ is the first ℓ columns of U.

See Appendix B for the proof.

We may also phrase this result geometrically, in terms of subspaces instead of matrices.

Corollary 5 (Geometric Principal Component Analysis)
Let $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^d$ be data points. If $A = \begin{bmatrix} \vec{x}_1 & \cdots & \vec{x}_n \end{bmatrix}$ is the data matrix, with SVD $A = U\Sigma V^\top$, then

$$S_{\text{PCA}} \in \operatorname*{argmin}_{S \subseteq \mathbb{R}^d} \sum_{i=1}^{n} \left\| \vec{x}_i - \text{proj}_S(\vec{x}_i) \right\|^2 \tag{14}$$

$$\text{s.t.} \quad \dim(S) \leq \ell \tag{15}$$

where $S_{\text{PCA}} = \text{Span}(\vec{u}_1, \ldots, \vec{u}_\ell)$ is the span of the first ℓ columns of U, and the argmin is taken over subspaces of $\mathbb{R}^d$.

Definition 6 (Principal Components)
Let $A = \begin{bmatrix} \vec{x}_1 & \cdots & \vec{x}_n \end{bmatrix} = U\Sigma V^\top$ be a data matrix. The *principal components* of A are the vectors $\vec{u}_1, \ldots, \vec{u}_d$ in that order. (For instance, $\vec{u}_1$ is the first principal component, $\vec{u}_2$ is the second principal component, etc.)

Based on our knowledge of how $\vec{u}_i$'s are the eigenvectors of $AA^\top$, we have the following equivalent, alternate characterization, which tells us how to calculate the principal components without calculating the SVD.

Theorem 7 (Principal Components as Eigenvectors)
Let $A = \begin{bmatrix} \vec{x}_1 & \cdots & \vec{x}_n \end{bmatrix} \in \mathbb{R}^{d \times n}$ be a data matrix. The principal components of D are the eigenvectors of $AA^\top$, ordered in non-increasing order by the value of the corresponding eigenvalue, with ties broken arbitrarily[a].

[a] By ties, we mean cases in which two or more eigenvalues in the non-increasing sequence are exactly equal.

Putting this into an algorithm gets us the following methods to find principal components, depending if the data points are columns or rows.

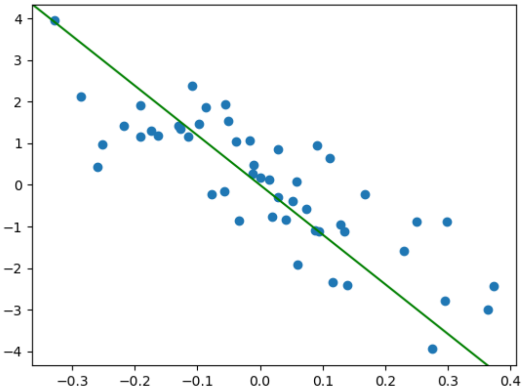
Here the $\textsc{Sort}(U, \Lambda)$ function sorts the columns of U in non-increasing order by the values on the diagonal of Λ, breaking ties arbitrarily.

Algorithm 8 Principal Component Analysis

```
function FINDPRINCIPALCOMPONENTS($A, \ell$)
    $A$ := Unnormalized Column Data
    $D = A(I_n - \frac{1}{n}\vec{1}_n\vec{1}_n^\top)$
    $(U, \Lambda) :=$ DIAGONALIZE($DD^\top$)
    return $U_\ell :=$ the first $\ell$ columns of SORT($U, \Lambda$)
end function
```

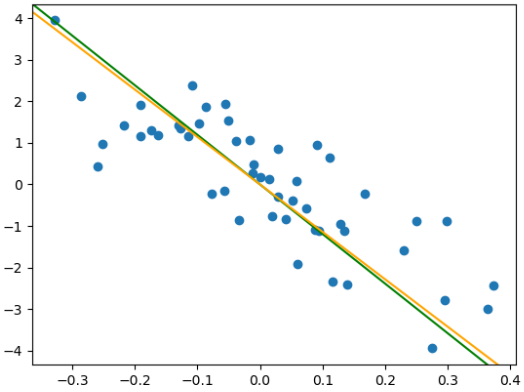
To give an idea of how effective this procedure is, suppose we have the following centered data (blue), with the "ground truth" subspace S_{gt} (green) also shown.

Figure 3: A sample dataset with $n = 50$, $d = 2$, and $\dim(S_{\mathrm{gt}}) = 1$ (green).



If we also plot the first principal component subspace (shown in orange), we get the following plot.

Figure 4: A sample dataset with $n = 50$, $d = 2$, $\dim(S_{\mathrm{gt}}) = 1$ (green), and $\dim(S_{\mathrm{PCA}}) = 1$ (orange).



The idea is: *the principal components learned from data, are very similar to the true low-rank subspace the ground truth data lie on!* This idea generalizes to many dimensions as well.

See Appendix C for the code that generates these plots.

Warning 9

All algorithms in this section work for *column data* – if we are working with *row data* of the form $A = \begin{bmatrix} \vec{x}_1^\top \\ \vdots \\ \vec{x}_n^\top \end{bmatrix} \in \mathbb{R}^{n \times d}$, which is very common in modern machine learning applications, we should take the transpose of the data matrix and then do PCA using the algorithms we already talked about.

4 Denoising and Dimensionality Reduction

Suppose we have a dataset $A \in \mathbb{R}^{d \times n}$ (where the data points are columns of A) which we perform PCA on, and get ℓ principal components $\vec{u}_1, \ldots, \vec{u}_\ell$. Suppose we are given a new noisy vector $\vec{x} \in \mathbb{R}^d$, and we know that the "ground truth" de-noised vector approximately lies in our low-rank ℓ-dimensional subspace. To recover an estimate for the original ground truth vector, we project onto this subspace, i.e., our principal component subspace. From what we know about projections, the formula to project onto $\mathrm{Span}(\vec{u}_1, \ldots, \vec{u}_\ell) = \mathrm{Col}(U_\ell)$ is

$$\mathrm{proj}_{\mathrm{Span}(\vec{u}_1,\ldots,\vec{u}_\ell)}(\vec{x}) = \mathrm{proj}_{\mathrm{Col}(U_\ell)}(\vec{x}) = U_\ell U_\ell^\top \vec{x}. \tag{16}$$

Since this process removes the residual noise and preserves the essential low-rank structure of $\vec{x}$, we call it *denoising*. This yields another formal algorithm:

Algorithm 10 Denoising using PCA

```
function PCADenoising($A, \vec{x}, \ell$)
    $U_\ell$ := FindPrincipalComponents($A, \ell$)
    return $\widehat{x} := U_\ell U_\ell^\top \vec{x}$
end function
```

We may also discuss PCA from the viewpoint of *data compression* and *dimensionality reduction*. Namely, if $\ell \leq d$, then we can approximately represent points $\vec{x}$ in $\mathbb{R}^d$ in our dataset, by vectors $\vec{w}$ in $\mathbb{R}^\ell$; these vectors $\vec{w}$ in $\mathbb{R}^\ell$ precisely store the coefficients in the linear combination of $\vec{u}_1, \ldots, \vec{u}_\ell$ required to approximately generate $\vec{x}$. The closest point in the ℓ-dimensional principal component subspace to $\vec{x}$ is its projection, it has the formula

$$\mathrm{proj}_{\mathrm{Span}(\vec{u}_1,\ldots,\vec{u}_\ell)}(\vec{x}) = U_\ell U_\ell^\top \vec{x} = \sum_{i=1}^{\ell} \langle \vec{x},\, \vec{u}_i \rangle\, \vec{u}_i. \tag{17}$$

Then the coefficients w_i which make up our compressed vector $\vec{w}$ are precisely the inner products $\langle \vec{x},\, \vec{u}_i \rangle$, so we have

$$\vec{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_\ell \end{bmatrix} = \begin{bmatrix} \langle \vec{x},\, \vec{u}_1 \rangle \\ \vdots \\ \langle \vec{x},\, \vec{u}_\ell \rangle \end{bmatrix} = \begin{bmatrix} \vec{u}_1^\top \vec{x} \\ \vdots \\ \vec{u}_\ell^\top \vec{x} \end{bmatrix} = \begin{bmatrix} \vec{u}_1^\top \\ \vdots \\ \vec{u}_\ell^\top \end{bmatrix} \vec{x} = U_\ell^\top \vec{x}. \tag{18}$$

To summarize, we can represent a vector $\vec{x} \in \mathbb{R}^d$ in our ℓ-dimensional subspace by ℓ coordinates – that is a vector $U_\ell^\top \vec{x}$ in $\mathbb{R}^\ell$ whose entries are the inner products of $\vec{x}$ with the first ℓ principal components. Even if $\vec{x}$ is not in our ℓ-dimensional subspace, we can represent $\vec{x}$ by a vector in $\mathbb{R}^\ell$ which approximately generates

$\vec{x}$ (as opposed to exactly generating $\vec{x}$). To recover the original vector (or in the latter case, an estimate for it), we can just multiply by U_ℓ again, which reduces to Algorithm 10.

If $\ell \ll d$ then this is a big success for us, since we have compressed our data a lot, and distilled it to its most important directions of variation.

Algorithm 11 Dimensionality Reduction using PCA

```
function PCADIMENSIONALITYREDUCTION($A, \vec{x}, \ell$)
    $U_\ell :=$ FINDPRINCIPALCOMPONENTS($A, \ell$)
    return $\vec{w} := U_\ell^\top \vec{x}$
end function
```

Warning 12

All algorithms in this section work for *column data* – if we are working with *row data* of the form $A = \begin{bmatrix} \vec{x}_1^\top \\ \vdots \\ \vec{x}_n^\top \end{bmatrix} \in \mathbb{R}^{n \times d}$, which is very common in modern machine learning applications, we should take the transpose of the data matrix and then do PCA using the algorithms we already talked about.

5 Picking the Best Number of Principal Components

Throughout this note, we have generated a fixed number of principal components ℓ. This assumes that we know our ground truth data has an ℓ-dimensional low-rank structure, i.e., it lies on some ℓ-dimensional subspace S_{gt}. However, in real-world applications, we would not know the dimensionality of the true S_{gt}. There are a few things we can do about this:

1. If we have hardware constraints, like in the lab, and can only take the first few principal components, then we should just use those.

2. If we are not constrained by hardware, then one thing we can do is the following:

   (a) Separate our data into a training set $\vec{x}_1, \ldots, \vec{x}_{n_{\text{train}}}$ and a validation set $\vec{x}_{1;\text{val}}, \ldots, \vec{x}_{n_{\text{val}};\text{val}}$.

   (b) Obtain principal components using *only* the training data; i.e., find the eigenvectors of $A_{\text{train}} A_{\text{train}}^\top$ where $A_{\text{train}} := \begin{bmatrix} \vec{x}_1 & \cdots & \vec{x}_{n_{\text{train}}} \end{bmatrix}$.

   (c) Measure distance to the generated subspace using *only* the validation data, i.e., compute

   $$\sum_{i=1}^{n_{\text{val}}} \left\| \vec{x}_{i;\text{val}} - U_k U_k^\top \vec{x}_{i;\text{val}} \right\|^2. \tag{19}$$

   (d) Stop taking additional components when this distance plateaus, i.e., stays the same after adding one or two more components. Because the principal component subspace grows larger with every iteration, our performance never becomes strictly worse with subsequent iterations, even if we are using the validation set. So, we cannot stop when our performance gets strictly worse; this is the next best stopping rule.

This stops us from adding extraneous principal components which actually capture the effect of noise on the training set. Formally, this can be turned into an algorithm.

In the following algorithm, we include a "stopping parameter" ϵ. This corresponds to a minimum improvement in the objective function that the new principal component needs to give, or else we conclude that we are done adding principal components and stop there.

Algorithm 13 A validation algorithm for PCA.

```
function PCAVALIDATION($A_{\text{train}}, A_{\text{val}}, \epsilon$)
    $U :=$ FINDPRINCIPALCOMPONENTS($A_{\text{train}}, d$)
    $s_0 = +\infty$                                        ▷ Objective function value
    for $i \in \{1, \ldots, d\}$ do
        $U_i :=$ first $i$ columns of $U$
        Compute $s_i = \sum_{k=1}^{n_{\text{val}}} \left\| \vec{x}_{k;\text{val}} - U_i U_i^\top \vec{x}_{k;\text{val}} \right\|^2$
        if $s_i - s_{i-1} \leq \epsilon$ then                 ▷ Stopping threshold
            return $i$          ▷ Return the right number of principal components to use.
        end if
    end for
    return $d$
end function
```

Warning 14

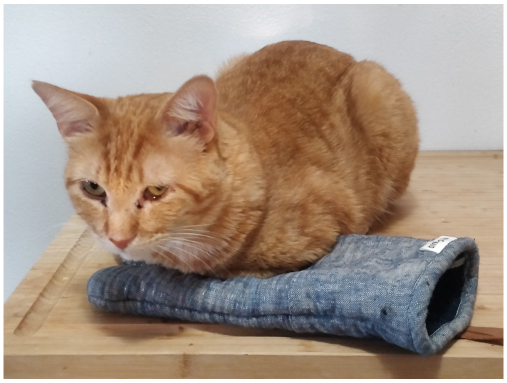
All algorithms in this section work for *column data* – if we are working with *row data* of the form $A = \begin{bmatrix} \vec{x}_1^\top \\ \vdots \\ \vec{x}_n^\top \end{bmatrix} \in \mathbb{R}^{n \times d}$, which is very common in modern machine learning applications, we should take the transpose of the data matrix and then do PCA using the algorithms we already talked about.

6 Examples

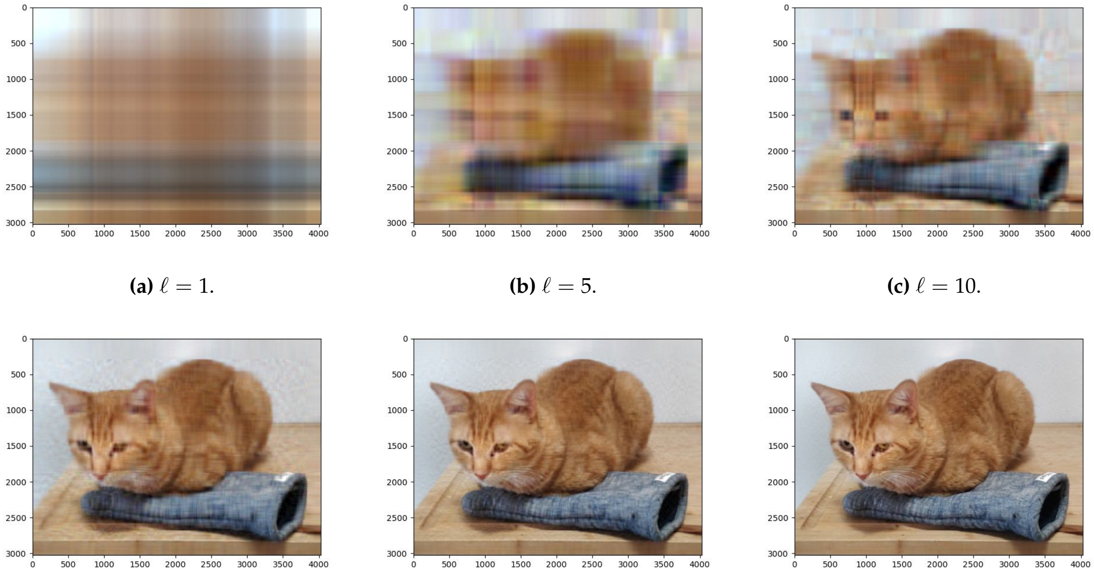
6.1 Low-Rank Approximation

The classical example of low-rank approximation is a image compression. More precisely, suppose we have an image like the below one:

Figure 5: The author's friend's cat Snyder.



It can be represented as three matrices $A_R, A_G, A_B \in \mathbb{R}^{4032\times 3024}$ corresponding to R, G, and B of the image. We perform a rank-ℓ approximation $A_R = U_{R;\ell}\Sigma_{R;\ell}V_{R;\ell}^{\top}$, $A_G = U_{G;\ell}\Sigma_{G;\ell}V_{G;\ell}^{\top}$, $A_G = U_{G;\ell}\Sigma_{G;\ell}V_{G;\ell}^{\top}$, and then compose an image out of them, for different values of ℓ. The results are shown below.



(a) $\ell = 1$.

(b) $\ell = 5$.

(c) $\ell = 10$.

(d) $\ell = 25$.

(e) $\ell = 50$.

(f) $\ell = 100$.

By rank 100 approximation, the image is almost perfect. Now, the original image had $3 \times 4032 \times 3024 = 36,578,304$ entries; at rank 100, we have $3 \times 100 \times (4032 + 3024 + 1) = 2,117,100$ entries, so we need to store around 5% of the original image. Not bad!

See Appendix D for some code showing how these images were created.

6.2 PCA

Suppose we, as course staff, have m students in our class and n assignments. Let $A \in \mathbb{R}^{m\times n}$ be a matrix, such that the i^{th} student's grade on the j^{th} assignment is A_{ij}.

- If we consider the assignments to be the data points, then A is a data matrix with column data. Computing the eigenvectors $\vec{u}_1, \ldots, \vec{u}_m$ of $AA^{\top}$ provides the principal components of the assignment data. One trend we could potentially see is the following:

  - The first principal component $\vec{u}_1$ might indicate whether the assignment has more theoretical or applied problems. That is, the projection coefficient of an assignment $\vec{x}$ onto the first principal component, $\langle \vec{x}, \vec{u}_1 \rangle = (U^\top \vec{x})_1$, would be a large positive number if the assignment $\vec{x}$ is purely theoretical, a large negative number if the assignment $\vec{x}$ is purely application-based, and somewhere in the middle if the assignment $\vec{x}$ has both types of problems.
  - The second principal component $\vec{u}_2$ might then indicate whether the assignment is long or short, in the same way (looking at the value of the projection coefficient $\langle \vec{x}, \vec{u}_2 \rangle = (U^\top \vec{x})_2$).
  - After a few more principal components, the data would be almost entirely captured by the principal component subspace. Any remaining residuals would be due to student performance on assignments not being perfectly consistent.

- If we consider the students to be the data points, then A is a data matrix with row data. Then the matrix $A^\top$ is a data matrix with column data. Computing the eigenvectors $\vec{v}_1, \ldots, \vec{v}_n$ of $(A^\top)(A^\top)^\top = A^\top A$ provides the principal components of the student data. We would expect something like:

  - The first principal component $\vec{v}_1$ might indicate whether the student prefers lots of applications or lots of theoretical problems. That is, the projection coefficient of a student $\vec{y}$ onto the first principal component, $\langle \vec{y}, \vec{v}_1 \rangle = (V^\top \vec{y})_1$, would be a large positive number if the student $\vec{y}$ purely favors theory, a large negative number if the student $\vec{y}$ purely favors applications, and somewhere in the middle if the student $\vec{y}$ doesn't strongly prefer one or the other.
  - The second principal component $\vec{v}_2$ might indicate whether the student is an underclassman or an upperclassman, in the same way (looking at the value of the projection coefficient $\langle \vec{y}, \vec{v}_2 \rangle = (V^\top \vec{y})_2$).
  - After a few more principal components, the data would be almost entirely captured by the principal component subspace, with any residuals being due to random variation.

NOTE: None of this section was based off of actual student data analysis; it was just the author's guess at what could be large orthogonal directions of variance in this dataset. It is worth realizing that while PCA is useful for dimensionality reduction, denoising, and noticing patterns in data, the exact features extracted are not, in general, directly interpretable with respect to the original features of the data. This represents one limitation of PCA as a feature extraction method. While we can make a an informed guess as to the nature of the principal components, they are not necessarily explainable via the original data's features as you saw in the examples above.

7 Final Comments

Low-rank approximation and principal component analysis are both powerful tools for data compression and data analysis. They both help us distill data down to its essential linear character, and so make it easy to work with in future applications such as statistics and machine learning. PCA, in particular, is used as a powerful pre-processing step in data science.

A Proof of Theorem 3

Proof of Theorem 3. The proof proceeds in several steps. Let $A = U\Sigma V^\top$ throughout, and let $\operatorname{rank}(A) = r$. Recall that we want to show that

$$A_k \in \operatorname*{argmin}_{\substack{B \in \mathbb{R}^{m\times n} \\ \operatorname{rank}(B)=\ell}} \|A - B\|_F^2. \tag{20}$$

1. *Reduce the problem with A down to a problem with Σ.*

   We use the fact that multiplication by an orthonormal matrix does not change the Frobenius norm to get

$$\begin{aligned}\|A - B\|_F^2 &= \left\|U\Sigma V^\top - B\right\|_F^2 &(21)\\ &= \left\|U^\top (U\Sigma V^\top - B)V\right\|_F^2 &(22)\\ &= \left\|U^\top U\Sigma V^\top V - U^\top BV\right\|_F^2 &(23)\\ &= \left\|\Sigma - U^\top BV\right\|_F^2. &(24)\end{aligned}$$

   Now since U and V are orthonormal matrices, they are full rank, and so $\operatorname{rank}(B) = \operatorname{rank}(U^\top BV)$. Thus if we let $X = U^\top BV$, the problem reduces to showing that

$$\begin{bmatrix} \Sigma_k & 0_{k\times(n-k)} \\ 0_{(m-k)\times k} & 0_{(m-k)\times(n-k)} \end{bmatrix} \in \operatorname*{argmin}_{\substack{X \in \mathbb{R}^{m\times n} \\ \operatorname{rank}(X)=\ell}} \|\Sigma - X\|_F^2. \tag{25}$$

2. *Reduce the problem with Σ to a problem of finding the best matrix Q with orthonormal columns.*

   Write, where $\vec{e}_i$ is the i^{th} standard basis vector in $\mathbb{R}^m$,

$$\begin{aligned}\|\Sigma - X\|_F^2 &= \sum_{i=1}^{n} \|(\Sigma - X)_i\|^2 &(26)\\ &= \sum_{i=1}^{n} \|\sigma_i \vec{e}_i - \vec{x}_i\|^2. &(27)\end{aligned}$$

   Since $\operatorname{rank}(X) = \ell$, the columns of X lie on an ℓ-dimensional subspace, say S which is spanned by the orthonormal basis $Q \in \mathbb{R}^{m\times \ell}$. Within S, the closest $\vec{x}_i$ to $\sigma_i\vec{e}_i$ (i.e., the $\vec{x}_i$ which minimizes $\|\sigma_i\vec{e}_i - \vec{x}_i\|^2$) is the projection of $\sigma_i\vec{e}_i$ onto $S = \operatorname{Col}(Q)$, which is given by

$$\vec{x}_i = \operatorname{proj}_S(\sigma_i\vec{e}_i) = \operatorname{proj}_{\operatorname{Col}(Q)}(\sigma_i\vec{e}_i) = \sigma_i QQ^\top \vec{e}_i. \tag{28}$$

   Simplifying the terms in the summand of the objective, we have

$$\begin{aligned}\|\sigma_i\vec{e}_i - \vec{x}_i\|^2 &= \left\|\sigma_i\vec{e}_i - \sigma_i QQ^\top\vec{e}_i\right\|^2 &(29)\\ &= \sigma_i^2\left\|\vec{e}_i - QQ^\top\vec{e}_i\right\|^2. &(30)\end{aligned}$$

   Notice that, by the orthogonality principle, we have

$$0 = \langle \operatorname{proj}_S(\vec{e}_i), \vec{e}_i - \operatorname{proj}_S(\vec{e}_i)\rangle = \left\langle \operatorname{proj}_{\operatorname{Col}(Q)}(\vec{e}_i), \vec{e}_i - \operatorname{proj}_{\operatorname{Col}(Q)}(\vec{e}_i)\right\rangle = \left\langle QQ^\top\vec{e}_i, \vec{e}_i - QQ^\top\vec{e}_i\right\rangle. \tag{31}$$

Thus, expanding the squared norm,

$$\begin{aligned}
\left\|\vec{e}_i - QQ^\top \vec{e}_i\right\|^2 &= \left\langle \vec{e}_i - QQ^\top \vec{e}_i,\, \vec{e}_i - QQ^\top \vec{e}_i \right\rangle && (32)\\
&= \left\langle \vec{e}_i,\, \vec{e}_i - QQ^\top \vec{e}_i \right\rangle - \underbrace{\left\langle QQ^\top \vec{e}_i,\, \vec{e}_i - QQ^\top \vec{e}_i \right\rangle}_{=0} && (33)\\
&= \left\langle \vec{e}_i,\, \vec{e}_i - QQ^\top \vec{e}_i \right\rangle && (34)\\
&= \langle \vec{e}_i,\, \vec{e}_i \rangle - \left\langle \vec{e}_i,\, QQ^\top \vec{e}_i \right\rangle && (35)\\
&= \langle \vec{e}_i,\, \vec{e}_i \rangle - \left\langle Q^\top \vec{e}_i,\, Q^\top \vec{e}_i \right\rangle && (36)\\
&= \|\vec{e}_i\|^2 - \left\|Q^\top \vec{e}_i\right\|^2 && (37)\\
&= 1 - \left\|Q^\top \vec{e}_i\right\|^2. && (38)
\end{aligned}$$

Therefore, simplifying the original problem, we have

$$\begin{aligned}
\underset{\substack{Q \in \mathbb{R}^{m\times \ell} \\ Q^\top Q = I_\ell}}{\operatorname{argmin}} \sum_{i=1}^{n} \sigma_i^2 \left(\left\|\vec{e}_i - QQ^\top \vec{e}_i\right\|^2 \right) &= \underset{\substack{Q \in \mathbb{R}^{m\times \ell} \\ Q^\top Q = I_\ell}}{\operatorname{argmin}} \sum_{i=1}^{n} \sigma_i^2 \left(1 - \left\|Q^\top \vec{e}_i\right\|^2 \right) && (39)\\
&= \underset{\substack{Q \in \mathbb{R}^{m\times \ell} \\ Q^\top Q = I_\ell}}{\operatorname{argmax}} \sum_{i=1}^{n} \sigma_i^2 \left\|Q^\top \vec{e}_i\right\|^2 && (40)\\
&= \underset{\substack{Q \in \mathbb{R}^{m\times \ell} \\ Q^\top Q = I_\ell}}{\operatorname{argmax}} \sum_{i=1}^{n} \left\|Q^\top (\sigma_i \vec{e}_i)\right\|^2 && (41)\\
&= \underset{\substack{Q \in \mathbb{R}^{m\times \ell} \\ Q^\top Q = I_\ell}}{\operatorname{argmax}} \left\|Q^\top \Sigma\right\|_F^2. && (42)
\end{aligned}$$

since the squared Frobenius norm is the sum of the squared norms of the columns.

Thus, the following are equivalent problems:

(a) Find a rank-ℓ matrix X which minimizes $\|\Sigma - X\|_F^2$.

(b) Find a dimension-ℓ subspace S which minimizes $\sum_{i=1}^{n} \left\|\sigma_i \vec{e}_i - \operatorname{proj}_S(\sigma_i \vec{e}_i)\right\|$;

(c) Find a matrix $Q \in \mathbb{R}^{m\times \ell}$ such that $Q^\top Q = I_\ell$, which maximizes $\left\|Q^\top \Sigma\right\|_F^2$.

And so the third problem is the one we would like to solve.

3. *Reduce the problem of finding the best matrix Q with orthonormal columns to a problem with purely real numbers.*

Write the columns of Q as $Q = \begin{bmatrix} \vec{q}_1 & \cdots & \vec{q}_\ell \end{bmatrix}$. Then we can simplify the Frobenius norm as

$$\left\|Q^\top \Sigma\right\|_F^2 = \left\| \begin{bmatrix} \vec{q}_1^\top \\ \vdots \\ \vec{q}_\ell^\top \end{bmatrix} \begin{bmatrix} \sigma_1 \vec{e}_1 & \cdots & \sigma_n \vec{e}_n \end{bmatrix} \right\|_F^2 \tag{43}$$

$$= \left\| \begin{bmatrix} \sigma_1 \langle \vec{e}_1, \vec{q}_1 \rangle & \cdots & \sigma_n \langle \vec{e}_n, \vec{q}_1 \rangle \\ \vdots & \ddots & \vdots \\ \sigma_1 \langle \vec{e}_1, \vec{q}_\ell \rangle & \cdots & \sigma_n \langle \vec{e}_n, \vec{q}_\ell \rangle \end{bmatrix} \right\|_F^2 \tag{44}$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{\ell} \sigma_i^2 \langle \vec{e}_i, \vec{q}_j \rangle^2 \tag{45}$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{\ell} \sigma_i^2 Q_{ij}^2 \tag{46}$$

$$= \sum_{i=1}^{n} \sigma_i^2 \sum_{j=1}^{\ell} Q_{ij}^2 \tag{47}$$

$$= \sum_{i=1}^{n} \sigma_i^2 d_i \tag{48}$$

$$= \sum_{i=1}^{r} \sigma_i^2 d_i \tag{49}$$

where $d_i := \sum_{j=1}^{\ell} Q_{ij}^2$. The last simplification is because only the first r singular values are nonzero.

As the sum of squared numbers, $d_i \geq 0$. We show that $d_i \leq 1$. Indeed, let $\widehat{Q}$ be the extension of Q to an orthonormal basis of $\mathbb{R}^m$; in particular, define $\widehat{Q} = \begin{bmatrix} Q & \widetilde{Q} \end{bmatrix}$ where $\widetilde{Q} \in \mathbb{R}^{m \times (m-\ell)}$ has orthonormal columns, so that $\widehat{Q} \in \mathbb{R}^{m \times m}$ is an orthonormal square matrix. Then $\widehat{Q}$ has orthonormal rows (and columns), so each of its rows is unit norm, so

$$1 = \sum_{j=1}^{m} \widehat{Q}_{ij}^2 \tag{50}$$

$$= \sum_{j=1}^{\ell} \widehat{Q}_{ij}^2 + \sum_{j=\ell+1}^{m} \widehat{Q}_{ij}^2 \tag{51}$$

$$= \sum_{j=1}^{\ell} Q_{ij}^2 + \sum_{j=1}^{m-\ell} \widetilde{Q}_{ij}^2 \tag{52}$$

$$= d_i + \sum_{j=1}^{m-\ell} \underbrace{\widetilde{Q}_{ij}^2}_{\geq 0} \tag{53}$$

$$\implies d_i \leq 1. \tag{54}$$

Now, since Q is orthonormal, this also gives us a constraint on the d_i; indeed,

$$\ell = \sum_{j=1}^{\ell} \left\| \vec{q}_j \right\|^2 \tag{55}$$

$$= \sum_{j=1}^{\ell} \sum_{i=1}^{m} Q_{ij}^2 \tag{56}$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{\ell} Q_{ij}^2 \tag{57}$$

$$= \sum_{i=1}^{m} d_i. \tag{58}$$

This gives the problem:

$$\begin{aligned} \max_{d_1,\dots,d_m \in \mathbb{R}} \quad & \sum_{i=1}^{r} \sigma_i^2 d_i && (59) \\ \text{s.t.} \quad & d_i \geq 0 \qquad i \in \{1,\dots,m\} && (60) \\ & d_i \leq 1 \qquad i \in \{1,\dots,m\} && (61) \\ & \sum_{i=1}^{m} d_i = \ell. && (62) \end{aligned}$$

Once we solve this problem, we can convert its answer onto constraints on Q. Any solution to this problem which has a corresponding Q will surely maximize $\left\|\Sigma^\top Q\right\|_F^2$; it is left to solve this problem and prove that a solution has a corresponding Q.

4. *Solve the numerical problem and find a maximizer.*

   One can show by a so-called exchange argument, or by inspection, that one maximizer of this problem is $d_1^\star = \cdots = d_\ell^\star = 1$ and $d_{\ell+1}^\star = \cdots = d_m^\star = 0$, at which point the optimal value is

$$\sum_{i=1}^{r} \sigma_i^2 d_i^\star = \sum_{i=1}^{\ell} \sigma_i^2. \tag{63}$$

5. *Using the solution to the numerical optimization problem, find a $Q_\star$ which maximizes $\left\|Q^\top \Sigma\right\|_F^2$.*

   Note that the quantity d_i is the squared norm of the i^{th} *row* of Q. Thus we are looking for a matrix $Q_\star \in \mathbb{R}^{m \times \ell}$ which has:

   - orthonormal columns;
   - the 1st through ℓ^{th} rows have unit norm;
   - the $(\ell+1)^{\text{th}}$ through m^{th} rows are $\vec{0}^\top$.

   A $Q_\star$ which satisfies this is given by $Q_\star = \begin{bmatrix} I_\ell \\ 0_{(m-\ell)\times\ell} \end{bmatrix}$. Thus this $Q_\star$ maximizes $\left\|Q^\top \Sigma\right\|_F^2$ among all $Q \in \mathbb{R}^{m \times \ell}$ with orthonormal columns.

6. *Compute the original objective function using this $Q_\star$ and show that it reduces to $\|A - A_k\|_F^2$.*

   We have that

$$\begin{aligned} \min_{\substack{B \in \mathbb{R}^{m\times n} \\ \operatorname{rank}(B)=\ell}} \|A - B\|_F^2 &= \min_{\substack{X \in \mathbb{R}^{m\times n} \\ \operatorname{rank}(X)=\ell}} \|\Sigma - X\|_F^2 && (64) \\ &= \left\|\Sigma - Q_\star Q_\star^\top \Sigma\right\|_F^2 && (65) \\ &= \left\| \begin{bmatrix} \Sigma_r & 0_{r\times(n-r)} \\ 0_{(m-r)\times r} & 0_{(m-r)\times(n-r)} \end{bmatrix} - \begin{bmatrix} I_\ell \\ 0_{(m-\ell)\times\ell} \end{bmatrix} \begin{bmatrix} I_\ell \\ 0_{(m-\ell)\times\ell} \end{bmatrix}^\top \begin{bmatrix} \Sigma_r & 0_{r\times(n-r)} \\ 0_{(m-r)\times r} & 0_{(m-r)\times(n-r)} \end{bmatrix} \right\|_F^2 && (66) \\ &= \left\| \begin{bmatrix} \Sigma_r & 0_{r\times(n-r)} \\ 0_{(m-r)\times r} & 0_{(m-r)\times(n-r)} \end{bmatrix} - \begin{bmatrix} \Sigma_\ell & 0_{\ell\times(n-\ell)} \\ 0_{(m-\ell)\times\ell} & 0_{(m-\ell)\times(n-\ell)} \end{bmatrix} \right\|_F^2 && (67) \end{aligned}$$

$$= \left\| U\left(\begin{bmatrix} \Sigma_r & 0_{r\times(n-r)} \\ 0_{(m-r)\times r} & 0_{(m-r)\times(n-r)} \end{bmatrix} - \begin{bmatrix} \Sigma_\ell & 0_{\ell\times(n-\ell)} \\ 0_{(m-\ell)\times\ell} & 0_{(m-\ell)\times(n-\ell)} \end{bmatrix} \right) V^\top \right\|_F^2 \tag{68}$$

$$= \left\| U\Sigma V^\top - U \begin{bmatrix} \Sigma_\ell & 0_{\ell\times(n-\ell)} \\ 0_{(m-\ell)\times\ell} & 0_{(m-\ell)\times(n-\ell)} \end{bmatrix} V^\top \right\|_F^2 \tag{69}$$

$$= \|A - A_\ell\|_F^2. \tag{70}$$

Thus

$$A_\ell \in \operatorname*{argmin}_{\substack{B\in\mathbb{R}^{m\times n} \\ \operatorname{rank}(B)=\ell}} \|A-B\|_F^2 \tag{71}$$

and the proof is complete. □

B Proof of Theorem 4

Proof of Theorem 4. We first simplify each term in the objective function.

$$\left\|\vec{x}_i - WW^\top\vec{x}_i\right\|^2 = \left\langle \vec{x}_i - WW^\top\vec{x}_i,\ \vec{x}_i - WW^\top\vec{x}_i \right\rangle \tag{72}$$

$$= \langle \vec{x}_i,\ \vec{x}_i\rangle - \left\langle WW^\top\vec{x}_i,\ \vec{x}_i\right\rangle - \left\langle \vec{x}_i,\ WW^\top\vec{x}_i\right\rangle + \left\langle WW^\top\vec{x}_i,\ WW^\top\vec{x}_i\right\rangle \tag{73}$$

$$= \langle \vec{x}_i,\ \vec{x}_i\rangle - 2\left\langle WW^\top\vec{x}_i,\ \vec{x}_i\right\rangle + \left\langle WW^\top WW^\top\vec{x}_i,\ \vec{x}_i\right\rangle \tag{74}$$

$$= \langle \vec{x}_i,\ \vec{x}_i\rangle - 2\left\langle WW^\top\vec{x}_i,\ \vec{x}_i\right\rangle + \left\langle WW^\top\vec{x}_i,\ \vec{x}_i\right\rangle \tag{75}$$

$$= \langle \vec{x}_i,\ \vec{x}_i\rangle - \left\langle WW^\top\vec{x}_i,\ \vec{x}_i\right\rangle \tag{76}$$

$$= \langle \vec{x}_i,\ \vec{x}_i\rangle - \left\langle W^\top\vec{x}_i,\ W^\top\vec{x}_i\right\rangle \tag{77}$$

$$= \|\vec{x}_i\|^2 - \left\|W^\top\vec{x}_i\right\|^2. \tag{78}$$

Thus we can reduce the optimization of the objective to the simpler optimization

$$\operatorname*{argmin}_{\substack{W\in\mathbb{R}^{n\times\ell} \\ W^\top W = I_\ell}} \sum_{i=1}^n \left\|\vec{x}_i - WW^\top\vec{x}_i\right\|^2 = \operatorname*{argmin}_{\substack{W\in\mathbb{R}^{n\times\ell} \\ W^\top W = I_\ell}} \sum_{i=1}^n \left[\|\vec{x}_i\|^2 - \left\|W^\top\vec{x}_i\right\|^2 \right] \tag{79}$$

$$= \operatorname*{argmax}_{\substack{W\in\mathbb{R}^{n\times\ell} \\ W^\top W = I_\ell}} \sum_{i=1}^n \left\|W^\top\vec{x}_i\right\|^2. \tag{80}$$

Now expanding the squared norm in terms of the sum of squares of each entry, we have

$$\sum_{i=1}^n \left\|W^\top\vec{x}_i\right\|^2 = \sum_{i=1}^n\sum_{k=1}^\ell (\vec{w}_k^\top\vec{x}_i)^2 \tag{81}$$

$$= \sum_{i=1}^n\sum_{k=1}^\ell \vec{w}_k^\top\vec{x}_i\vec{x}_i^\top\vec{w}_k \tag{82}$$

$$= \sum_{k=1}^\ell\sum_{i=1}^n \vec{w}_k^\top\vec{x}_i\vec{x}_i^\top\vec{w}_k \tag{83}$$

$$= \sum_{k=1}^\ell \vec{w}_k^\top\left(\sum_{i=1}^n \vec{x}_i\vec{x}_i^\top\right)\vec{w}_k \tag{84}$$

$$= \sum_{k=1}^{\ell} \vec{w}_k^\top \left(AA^\top \right) \vec{w}_k. \tag{85}$$

Computing $AA^\top = U\Sigma\Sigma^\top U^\top$, we have

$$\sum_{i=1}^{n} \left\| W^\top \vec{x}_i \right\|^2 = \sum_{k=1}^{\ell} \vec{w}_k^\top \left(AA^\top \right) \vec{w}_k \tag{86}$$

$$= \sum_{k=1}^{\ell} \vec{w}_k^\top U\Sigma\Sigma^\top U^\top \vec{w}_k. \tag{87}$$

We introduce the change of coordinates $\vec{\tilde{w}}_k := U^\top \vec{w}_k$. Note that since the $\vec{w}_k$ are orthonormal, so too are the $\vec{\tilde{w}}_k$. Expanding out $\Sigma\Sigma^\top$, we have

$$\sum_{i=1}^{n} \left\| W^\top \vec{x}_i \right\|^2 = \sum_{k=1}^{\ell} \vec{w}_k^\top U\Sigma\Sigma^\top U^\top \vec{w}_k \tag{88}$$

$$= \sum_{k=1}^{\ell} \vec{\tilde{w}}_k^\top \Sigma\Sigma^\top \vec{\tilde{w}}_k \tag{89}$$

$$= \sum_{k=1}^{\ell} \vec{\tilde{w}}_k^\top \begin{bmatrix} \sigma_1^2 & & & & & \\ & \ddots & & & & \\ & & \sigma_r^2 & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix} \vec{\tilde{w}}_k \tag{90}$$

$$\tag{91}$$

Since the $\vec{\tilde{w}}_k$'s are orthonormal, a maximizing choice of $\vec{\tilde{w}}_1, \ldots, \vec{\tilde{w}}_\ell$ is $\vec{e}_1, \ldots, \vec{e}_\ell$ (i.e., the standard basis vectors)[1] – at which point the objective value is $\sum_{k=1}^{\ell} \sigma_k^2$. The corresponding $\vec{w}_k$ is given by

$$\vec{w}_k = U\vec{\tilde{w}}_k = U\vec{e}_k = \vec{u}_k \tag{92}$$

so a maximizing W is U_ℓ as desired. □

[1] There are other maximizers; for example, any permutation of the first ℓ standard basis vectors, or if some singular values are equal then those can also be hit by an $\vec{e}_k$.

C Code for PCA Plots

```
import numpy as np
import matplotlib.pyplot as plt
import pathlib

rng = np.random.RandomState(16)

n = 50
sigma = 0.1

s_vec = rng.randn(2, 1)  # (2, 1)
X = s_vec @ rng.randn(1, n) + sigma * rng.randn(2, n)  # (2, 50)
plt.axline((0, 0), s_vec.reshape(2, ), color='green')
plt.scatter(X[0], X[1])

plt.savefig(pathlib.Path("figures") / "pca_data.png")

U, S, Vh = np.linalg.svd(X)
plt.axline((0, 0), U[:, 0], color='orange')

plt.savefig(pathlib.Path("figures") / "pca.png")
```

D Code for Low-Rank Compression

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.image import imread
import pathlib


img = imread(pathlib.Path("figures") / "snyder.jpg")
A_R, A_G, A_B = img[:, :, 0], img[:, :, 1], img[:, :, 2]

# normalize to [0, 1]
A_R = A_R.astype(np.single) / 255
A_G = A_G.astype(np.single) / 255
A_B = A_B.astype(np.single) / 255

# Now take SVD and truncate to get approximations
U_R, S_R, Vh_R = np.linalg.svd(A_R)
U_G, S_G, Vh_G = np.linalg.svd(A_G)
U_B, S_B, Vh_B = np.linalg.svd(A_B)
```

```
Sigma_R, Sigma_G, Sigma_B = np.diag(S_R), np.diag(S_G), np.diag(S_B)
for l in (1, 5, 10, 25, 50, 100):
    img_l = np.zeros(shape=img.shape)
    img_l[:, :, 0] = U_R[:, :l] @ Sigma_R[:l, :l] @ Vh_R[:l]
    img_l[:, :, 1] = U_G[:, :l] @ Sigma_G[:l, :l] @ Vh_G[:l]
    img_l[:, :, 2] = U_B[:, :l] @ Sigma_B[:l, :l] @ Vh_B[:l]
    plt.imshow(img_l)
    plt.savefig(pathlib.Path("figures") / f"snyder_{l}.jpg")
```

Contributors:

- Druv Pai.
- Rahul Arya.
- Anant Sahai.
- Ayan Biswas.
- Ashwin Vangipuram.
- Kamyar Salahi.
- Matteo Guarrera.
- Aakarsh Vermani.
- Chancharik Mitra.