Machine Learning in Computer Vision:
PCA: the Classic Version and Some Variants

Oren Freifeld
Computer Science, Ben-Gurion University

May 16, 2018
1. Spaces of Ordered Orthonormal Bases or Linear Subspaces

2. Principal Component Analysis (PCA)

3. Scalable PCA via EM-PCA

4. Robust PCA
   - Examples for CV Applications of RPCA
   - Traditional RPCA Methods

5. Scalable Robust PCA via Trimmed Grassmann Averages
Definition

$V_k(\mathbb{R}^n)$, the Stiefel manifold, of order $(n, k)$, where $k \in \{1, 2, \ldots, n\}$, is the set of orthonormal $n \times k$ matrices (i.e., “skinny-tall” when $k < n$); namely

$$V_k(\mathbb{R}^n) = \{ V \text{ such that } V \in \mathbb{R}^{n \times k} \text{ and } V^T V = I_{k \times k} \}$$
$V_k(\mathbb{R}^n) = \{V \text{ such that } V \in \mathbb{R}^{n \times k} \text{ and } V^T V = I_{k \times k}\}$

**Example ($k = 1$)**

$V_1(\mathbb{R}^n) = S^{n-1}$; namely, the $(n-1)$-dimensional unit sphere in $\mathbb{R}^n$:

$V_1(\mathbb{R}^n) = S^{n-1} \triangleq \{x \text{ such that } x \in \mathbb{R}^n \text{ and } x^T x = \|x\|^2 = 1\}$

**Example ($k = 1$, $n = 2$)**

$V_1(\mathbb{R}^2) = \left\{ \begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix} \text{ such that } \theta \in [0, 2\pi) \right\}$

Note that, e.g.,

$\begin{bmatrix} \cos 30^\circ \\ \sin 30^\circ \end{bmatrix}$ and $\begin{bmatrix} \cos 210^\circ \\ \sin 210^\circ \end{bmatrix}$

are viewed as two distinct elements of $V_1(\mathbb{R}^2) = S^1$ (although they of course span the same 1D linear subspace).
Spaces of Ordered Orthonormal Bases or Linear Subspaces

\[ V_k(\mathbb{R}^n) = \{ V \text{ such that } V \in \mathbb{R}^{n \times k} \text{ and } V^T V = I_{k \times k} \} \]

**Example \((k = n)\)**

\(V_n(\mathbb{R}^n)\) is the set of all \(n \times n\) matrices such that the \(n\) columns of each matrix form an (ordered) orthonormal basis of \(\mathbb{R}^n\); i.e., each point in \(V_n(\mathbb{R}^n)\) corresponds to an entire (ordered) \(n\)-dimensional basis of \(\mathbb{R}^n\).

**Exercise**

Show that, \(\forall a \in [-1, 1]\),

\[
\begin{bmatrix}
  a & -\sqrt{1-a^2} \\
  \sqrt{1-a^2} & a
\end{bmatrix}
\in V_2(\mathbb{R}^2).
\]

“Ordered” means, e.g., that, \(\forall a \in [-1, 1]\),

\[
\begin{bmatrix}
  a & -\sqrt{1-a^2} \\
  \sqrt{1-a^2} & a
\end{bmatrix} \text{ and } \begin{bmatrix}
  -\sqrt{1-a^2} & a \\
  a & \sqrt{1-a^2}
\end{bmatrix}
\]

(same columns, different order) are two distinct elements of \(V_n(\mathbb{R}^n)\).
**Definition**

$\text{Gr}(k, n)$, the Grassmann manifold (AKA the Grassmannian), of order $(n, k)$, where $k \in \{1, 2, \ldots, n\}$, is the set of all $k$-dimensional linear subspaces of $\mathbb{R}^n$. 
Example

Gr(1, n) is the set of all (unbounded-on-both-ends) 1D lines in $\mathbb{R}^n$ passing through the origin.
Example

Gr(2, 3) is the set of all 2D planes in $\mathbb{R}^3$ passing through the origin.
Example

$\text{Gr}(n, n)$ is $\{\mathbb{R}^n\}$; in effect, a set that contains a single element, and this element is $\mathbb{R}^n$, the only $n$-dimensional linear subspace of $\mathbb{R}^n$. 
Each $\mathcal{V} \in \text{Gr}(k, n)$ corresponds to a subset of $V_k(\mathbb{R}^n)$, i.e.,

$$\text{Gr}(k, n) \ni \mathcal{V} \leftrightarrow \{ V : V \in V_k(\mathbb{R}^n) \text{ such that the columns of } V \text{ span } \mathcal{V} \}$$
In other words, a point in $\text{Gr}(k, n)$ is what is called an equivalence class of elements in $V_k(\mathbb{R}^n)$. This is why we also write

$$\text{Gr}(k, n) \ni \mathcal{V} \leftrightarrow [V] \subset V_k(\mathbb{R}^n)$$

where $\mathcal{V}$ is some element of $V_k(\mathbb{R}^n)$ whose columns span $\mathcal{V}$.

**Example**

If $\hat{x} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ then $\text{Gr}(1, 2) \ni \{\begin{bmatrix} x \\ 0 \end{bmatrix} : x \in \mathbb{R}\} \leftrightarrow [\hat{x}] = \{\hat{x}, -\hat{x}\} \subset V_1(\mathbb{R}^2)$
Fact

If

\[ U \in [V] \subset V_k(\mathbb{R}^n) \]

then

\[ UU^T = VV^T \in \mathbb{R}^{n \times n}. \]

Example

\[ V = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad U = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \Rightarrow \]

\[ VV^T = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \begin{bmatrix} -1 & 0 \end{bmatrix} = UU^T. \]
Exercise

Show that if $V \in V_k(\mathbb{R}^n)$ then:

$$\left(VV^T\right)^2 \triangleq (VV^T)(VV^T) = VV^T; \quad (1)$$

$$VV^T = I_{n \times n} \iff k = n; \quad (2)$$

$$\det(VV^T) \neq 0 \iff k = n. \quad (3)$$

Example ($k = 1, n = 2$)

$$V = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \Rightarrow VV^T = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

and indeed

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \neq I_{2 \times 2}. $$
In other words, if $k < n$, then, usually, $x \neq VV^T x$.

Equivalently, usually $(I_{n \times n} - VV^T)x \neq 0_{n \times 1}$. In fact (and trivially),

$x = VV^T x \iff x$ is in the null space of $I_{n \times n} - VV^T$.

**Example**

Let us again use $V = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. Then

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ 0 \end{bmatrix} = \begin{bmatrix} x \\ 0 \end{bmatrix} \quad \forall x \in \mathbb{R}$$

(4)

since $\begin{bmatrix} x \\ 0 \end{bmatrix}$ is in the null space of $I_{2 \times 2} - \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$.

**Exercise**

Perform similar analysis for $V = \begin{bmatrix} \cos \theta & \sin \theta \end{bmatrix}^T$. 
Spaces of Ordered Orthonormal Bases or Linear Subspaces

**Fact**

There is a one-to-one correspondence between $\text{Gr}(k, n)$ and rank-$k$ $n \times n$ idempotent matrices. In effect,

\[
\text{Gr}(k, n) \ni \mathcal{V} \leftrightarrow [\mathbf{V}] \subset V_k(\mathbb{R}^n) \iff \text{Gr}(k, n) \ni \mathcal{V} \leftrightarrow \mathbf{Q} = \mathbf{V} \mathbf{V}^T
\]
Optimization over the Stiefel and Grassmann Manifolds

Fact

Both Stiefel and the Grassmann manifolds are nonlinear spaces.

Fact (which most scientists and practitioners are still not aware of)

Many problems in Computer Vision and Machine Learning can be casted as an optimization problem over either a Stiefel manifold or a Grassmann manifold.

Despite the nonlinearity, it turns out that optimization in these spaces is not harder (nor easier) than standard optimization in $\mathbb{R}^k$. 
Optimization over the Stiefel and Grassmann Manifolds

We will only briefly touch upon few special and easy problems over such spaces; so easy that we can be ignorant about the underlying nonlinearities and geometry of these spaces. **If you like to know more:**

- Nicolas Boumal’s Matlab Toolbox Manopt at http://www.manopt.org/ (see also Jamie Townsend’s pyManopt)
- Edelman et al., 1998: “The geometry of algorithms with orthogonality constraints”.
  For EE/CS/CogSci students, one of the great things in this paper is that it is written in terms of linear algebra (as opposed to differential geometry)
- Nicolas Boumal’s tutorial slides from ICML ’16 Workshop on Geometry in Machine Learning https://sites.google.com/site/gimliworkshop/schedule
Eigenfaces

- The images on the left can be approximated using a linear combination of the images on the right.
- The images on right, viewed as vectors in $\mathbb{R}^{\#\text{pixels}}$, form an orthonormal basis of a 12-dim linear subspace of $\mathbb{R}^{\#\text{pixels}}$.
- This basis is optimal, at least two senses (wait a few slides).
- This is an example of linear dimensionality reduction.

Figure from Szeliski’s textbook, 2010
Figure 14.14  Projection onto the linear subspace spanned by the eigenface images (Moghaddam and Pentland 1997) © 1997 IEEE. The distance from face space (DFFS) is the orthogonal distance to the plane, while the distance in face space (DIFS) is the distance along the plane from the mean image. Both distances can be turned into Mahalanobis distances and given probabilistic interpretations.

Figure from Szeliski’s textbook, 2010
Eigenflow (Learned Basis of Dense Optical Flow)

Figure from [Black, Yacoob and Jepson, 1997]
Principal Component Analysis (PCA)

Eigencontours

Figure from [Freifeld et al. 2010]
Principal Component Analysis (PCA)

**Data**

- Data:
  \[
  (y_i)_{i=1}^N \subset \mathbb{R}^d.
  \]

- Make it zero (sample-) mean:
  \[
  (x_i)_{i=1}^N = (y_i - \bar{y})_{i=1}^N
  \]

  where

  \[
  \bar{y} = \frac{1}{N} \sum_{i=1}^{N} y_i.
  \]

**Exercise**

Show that

\[
\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i = 0_{d \times 1}.
\]
Data

- zero-mean data matrix:

\[
X = \begin{bmatrix}
    x_1 & \ldots & x_N
\end{bmatrix} \subset \mathbb{R}^{d \times N}
\]

- An unnormalized sample covariance (or correlation, since it’s zero mean) of \(X\):

\[
XX^T = \sum_{i=1}^{N} x_i x_i^T \subset \mathbb{R}^{d \times d}
\]

Remark

If \(d > N\), or even if \(N \geq d\) but \(XX^T\) is rank-deficient (too many of the \(x_i\)’s are linear combinations of the others), then \(XX^T\) is singular.

Remark

If \(d\) is large, working with \(XX^T\) (or even just storing it in memory) might be too hard.
Principal Component Analysis (PCA)

Data

Exercise

Show that $XX^T \in \mathbb{R}^{d \times d}$ is symmetric.

Fact

- Eigenvalues of $XX^T$ are nonnegative (so it is SPSD).
- Eigenvalues of $XX^T$ are all positive (i.e., it’s SPD) $\iff$ it’s rank $d$. 

**Eigenvalue Decomposition**

- Eigenvalue decomposition of $XX^T \in \mathbb{R}^{d \times d}$:

$$XX^T = VSV^T = \sum_{i=1}^{d} s_i v_i v_i^T \quad V^T V = I_{d \times d}$$

$$V = \begin{bmatrix} v_1 & \ldots & v_d \end{bmatrix} \in \mathbb{R}^{d \times d} \quad (s_i)_{i=1}^{d} \subset \mathbb{R}_{\geq 0}$$

- $S = \text{diag}((s_i)_{i=1}^{d}) \in \mathbb{R}^{d \times d}$, i.e. $S$ is diagonal with $S_{ii} = s_i$, $i = 1, \ldots, d$.

**Exercise**

$$V^T V = I_{d \times d} \Rightarrow VV^T = I_{d \times d}$$

- $V$ is an orthogonal (square) matrix; namely, its columns are orthonormal and its rows are orthonormal.
Eigenvalue Decomposition

Exercise

Show that \((s_i)_{i=1}^d\) are the eigenvalues of \(XX^T\) and the columns of \(V\) are its right/left eigenvectors. In other words, show that, for \(j \in \{1, \ldots, d\}\),

\[
XX^T v_j = s_j v_j \quad \text{and} \quad v_j^T XX^T = s_j v_j^T
\]
Eigenvalue Decomposition

- We will assume the eigenvalues are sorted in a non-decreasing order:

\[ s_1 \geq s_2 \geq \ldots \geq s_d. \]

**Remark**

On a computer, when computing the eigenvalues/vectors of \( XX^T \), use a routine that exploits the symmetry of \( XX^T \); not just because of the efficiency, but since otherwise you will most likely get complex numbers due to numerical issues.

For example, use `np.linalg.eigh` and not `np.linalg.eig`.

- The corresponding eigenvectors are called the principal components.

**Remark**

Popular spelling mistake: principal versus principle.
Recall the sample mean of the data is zero.

- The unnormalized and normalized sample covariance matrices, $XX^T$ and $\frac{1}{N}XX^T$, have the same eigenvectors. The eigenvalues of $\frac{1}{N}XX^T$ are 

$$
\left( \frac{S_i}{N} \right)^d
$$

- $S_i/N$ is the (sample) variance of the (zero-mean) data when projected along the 1D subspace spanned by $\mathbf{v}_i$:

$$
\hat{\sigma}_i^2 = \frac{S_i}{N} = \frac{1}{N} \sum_{j=1}^{N} (\mathbf{v}_i^T \mathbf{x}_j)^2 = \mathbf{v}_i^T \left( \frac{1}{N} \sum_{j=1}^{N} \mathbf{x}_j \mathbf{x}_j^T \right) \mathbf{v}_i
$$
Principal Component Analysis

- $X \in \mathbb{R}^{d \times N}$
- $\text{rank}(X) = \text{rank}(XX^T) = |i : s_i > 0| \leq \min(N, d)$
- Let $k = \text{rank}(X)$. Suppose $k < d$. Then

$$XX^T = \sum_{i=1}^{d} s_i v_i v_i^T = \sum_{i=1}^{k} s_i v_i v_i^T$$

$$= \begin{bmatrix} v_1 \ldots v_k \end{bmatrix} S_{1:k,1:k} \begin{bmatrix} v_1 \ldots v_k \end{bmatrix}^T$$

where $S_{1:k,1:k} = \text{diag}((s_i)_{i=1}^{k})$ and $\begin{bmatrix} v_1 \ldots v_k \end{bmatrix} \in \mathbb{R}^{d \times k}$ consists of the first $k$ columns of $V$.

- The matrix $\begin{bmatrix} v_1 \ldots v_k \end{bmatrix} \in \mathbb{R}^{d \times k}$ is an element of the Stiefel Manifold $V_k(\mathbb{R}^n)$; i.e., $V^T V = I_{k \times k}$. 

May 16, 2018 29 / 73
Let $N \leq d$. The usually-rectangular $X \in \mathbb{R}^{d \times N}$ can be decomposed as:

$$
\begin{bmatrix}
\end{bmatrix}
= 
\begin{bmatrix}
\end{bmatrix}
\begin{bmatrix}
\end{bmatrix}
\begin{bmatrix}
\end{bmatrix}
$$

Figure: adapted from wikipedia.
Let $N \leq d$. $X \in \mathbb{R}^{d \times N}$. The SVD of $X$ is

$$
X = V \Lambda U^T \quad \Lambda \in \mathbb{R}^{d \times N}
$$

$$
U = \begin{bmatrix}
    u_1 & \ldots & u_N
\end{bmatrix} \in \mathbb{R}^{N \times N} \quad U^T U = I_{N \times N}
$$

$$
V = \begin{bmatrix}
    v_1 & \ldots & v_d
\end{bmatrix} \in \mathbb{R}^{d \times d} \quad V^T V = I_{d \times d}
$$

where:

- $\text{diag}(\Lambda) = (\sqrt{s_i})_{i=1}^N$;
- $\Lambda_{ij} = 0$ if $i \neq j$;
- $V$ and the $s_i$’s are the same ones from before.

The fact above means that the left singular vectors of $X$ are the eigenvectors of $XX^T$. The first $N$ singular values of $X$ are the square roots of the first $N$ eigenvalues of $XX^T$ (the trailing $d-N$ eigenvalues of $XX^T$ will be zeros). Indeed, $XX^T = V \Lambda U^T U \Lambda^T V^T = V S V^T$ (note the remark about the trailing $d-N$ eigenvalues).
Let $\mathbf{u}_i^T = [u_{i,1} \ldots u_{i,N}]$.

$$
X = V \Lambda U^T = V \Lambda \begin{bmatrix}
\mathbf{u}_1^T \\
\vdots \\
\mathbf{u}_N^T
\end{bmatrix} = V \Lambda 
\begin{bmatrix}
\mathbf{u}_{1,1} \ldots \mathbf{u}_{1,N} \\
\vdots \\
\mathbf{u}_{N,1} \ldots \mathbf{u}_{N,N}
\end{bmatrix}
$$

$$
\mathbf{x}_j = V \Lambda \begin{bmatrix}
u_{1,j} \\
\vdots \\
u_{N,j}
\end{bmatrix} = V \begin{bmatrix}
\sqrt{s_1} u_{1,j} \\
\vdots \\
\sqrt{s_N} u_{N,j} \\
\mathbf{0}_{(d-N) \times 1}
\end{bmatrix} = \sum_{i=1}^{N} \sqrt{s_i} u_{i,j} \mathbf{v}_i
$$

$$
\Rightarrow \mathbf{x}_j = \sum_{i=1}^{N} \sqrt{s_i} u_{i,j} \mathbf{v}_i \quad \text{(linear combination of the } \mathbf{v}_i \text{'s)}
$$
\[ N \leq d \]

\[ x_j = \sum_{i=1}^{N} \sqrt{s_i} u_{i,j} v_i \]

\[ X = V \Lambda U^T = \sum_{i=1}^{N} \sqrt{s_i} v_i u_i^T \]
Principal Component Analysis (PCA)

Principal Component Analysis for Dimensionality Reduction

**Definition**

Let $1 \leq k \leq \text{rank}(\mathbf{X}) = \min(d, N)$. $\mathbf{V}_k \triangleq \mathbf{V}_{:,1:k}$ is the $k$-dim PCA subspace of $\mathbf{X}$.

\[
\mathbb{R}^{d \times N} \ni \mathbf{X} \approx \sum_{i=1}^{k} \sqrt{s_i} \mathbf{v}_i \mathbf{u}_i^T \quad \text{(approximating all the data)}
\]

\[
\mathbb{R}^{d} \ni \mathbf{x}_j \approx \sum_{i=1}^{k} \sqrt{s_j} \mathbf{u}_{i,j} \mathbf{v}_i = \mathbf{V}_k \mathbf{c}_j \quad \text{(approximating a single datum)}
\]

\[
\mathbb{R}^{k} \ni \mathbf{c}_j \triangleq \begin{bmatrix}
\sqrt{s_1} u_{1,j} \\
\vdots \\
\sqrt{s_k} u_{k,j}
\end{bmatrix} = \mathbf{V}_k^T \mathbf{x}_j \quad \text{(Exercise: show this equality is true)}
\]

\[
\varepsilon_j \triangleq \mathbf{x}_j - \mathbf{V}_k \mathbf{c}_j = \mathbf{x}_j - \mathbf{V}_k \mathbf{V}_k^T \mathbf{x}_j = (\mathbf{I}_{d \times d} - \mathbf{V}_k \mathbf{V}_k^T) \mathbf{x}_j
\]
**Fact**

\[ V_k = \arg \min_{\tilde{V}: \tilde{V} \in \mathbb{R}^{d \times k}, \tilde{V}^T \tilde{V} = I_{k \times k}} \left\| X - \tilde{V} \tilde{V}^T X \right\|_F^2 \]

where, for \( A = [a_1 \ldots a_N] \subset \mathbb{R}^{d \times N} \),

\[ \| A \|_F = \sqrt{\sum_{i=1}^{N} \| a_i \|_{\ell_2}^2} \]

is the Frobenius norm (AKA Hilbert-Schmidt norm).

**Exercise**

Show that

\[ \| A \|_F = \| \text{vec}(A) \|_{\ell_2} = \sqrt{\sum_i \sum_j A_{ij}^2} = \sqrt{\text{trace}(A^T A)}. \]
Approximation error:

\[ X - V_k V_k^T X = (I_N \times N - V_k V_k^T)X = [\varepsilon_1 \ldots \varepsilon_N] \subset \mathbb{R}^{d \times N} \]

Recall:

\[ P \triangleq I_{d \times d} - V_k V_k^T \in \mathbb{R}^{d \times d} \] is a projection matrix; i.e., it is idempotent, namely

\[ P^2 \triangleq PP = P. \]
Some of the issues with learning PCA (here, “learning” simply means computing $V_k$ from $X$):

- $l_2$ is not robust
- Misalignment (e.g., of faces)
- Scalability
- While optimal under a Gaussian model, can be suboptimal otherwise
- Problematic for data in a nonlinear space – especially for synthesis
There are also problems with using the PCA (after we learned it); e.g.:

- Presumably, we want to use the low-dim $c \in \mathbb{R}^k$ for some task (e.g., classification). Perhaps a different low-dim subspace would have lead to better results in that task.

- Given a new point, $x \in \mathbb{R}^d$, the orthogonal projection operation,

$$\mathbb{R}^k \ni c = V_k^T x,$$

is not robust to the in-subspace outliers. In other words, the orthogonal projection operation itself can be viewed as a minimizer some $\ell_2$ loss (hence no robustness). This problem is not unique to PCA subspaces; it can happen whenever we use orthonormal projection on some fixed subspace.
We will discuss several variants of PCA, as well as other ways (e.g., more efficient) ways to compute it.
Principal Component Analysis for Dimensionality Reduction

Fact

\[ V_k^{(1)} \overset{1}{=} \arg \min_{\tilde{V} \in V_k(\mathbb{R}^d)} \sum_{j=1}^{N} \left\| \text{reconstruction error } j \right\|^2 \quad (\text{already told you}) \]

\[ V_k^{(2)} \overset{2}{=} \arg \max_{\tilde{V} \in V_k(\mathbb{R}^d)} \sum_{i=1}^{k} \left\| \text{unnormalized var in direction } i \right\| \quad (\text{telling you now}) \]

- So PCA can be interpreted as the solution to two seemingly-different optimization problems.
Principal Component Analysis

- If $d \ll N$, eigenvalue decomposition of $XX^T$ is much better than SVD of $X$.
- If $N \ll d$, SVD of $X$ is much better than eigenvalue decomposition of $XX^T$.
- In both cases “much better” might still be very bad.
Two of the issues with learning PCA that we mentioned before are:

- $\ell_2$ is not robust
- Scalability: Eigenvalues decomposition on $XX^T$ works well only if $d$ is low (where the covariance can easily be stored in memory) while SVD on $X$ is impractical when either $d$ or $N$ is too large (SVD has cubic complexity)

Here, “learning” simply means computing $V_k$ from $X$
Scalable PCA

There are several methods for scalable PCA. A common choice is the EM PCA [Roweis, NIPS 1998] which computes each principal component by repeatedly iterating (till convergence)

\[
\tilde{v} \leftarrow \sum_{i=1}^{N} (x_i^T v_{[t-1]}) x_i \quad \text{and} \quad v_{[t]} = \frac{\tilde{v}}{\|\tilde{v}\|}
\]

Here, \(v_{[t]}\) is the estimate of the principal component at iteration \(t\).

Let \(v\) denote the converged solution.

Upon convergence, \(v\) is removed from the data, i.e., \(\tilde{X} = (I_{n \times n} - vv^T) X\), and the next principal component is computed in a similar way from \(\tilde{X}\), and so on, till \(k\) principal components are found.
Scalable PCA

- The algorithm can be derived in several different ways, including as an Expectation-Maximization algorithm, hence the name.
- This algorithm has linear complexity and low memory footprint, making it highly scalable.
- The algorithm, however, is not robust (as it is tied to an $\ell_2$ loss).
RPCA for Background Modeling
RPCA for Fixing Old Movies

Removing outliers at both pixel and frame level (e.g., due to random jumps in the illumination)
Robust PCA in Computer Vision

The two most known methods are

- [Del la Torre and Black, 1998]
- [Candes, Li, Ma, and Wright, 2003]
Del la Torre and Black Robust PCA, 1998

- Modulo some details we ignore here, they replaced

\[
V_k = \arg \min_{V \in V_k(\mathbb{R}^n)} \sum_{i=j}^{N} \left\| (I_{d \times d} - V \tilde{V}^T) x_j \right\|^2
\]

with

\[
V_k = \arg \min_{V \in V_k(\mathbb{R}^n)} \sum_{j=1}^{N} \rho \left( (I_{d \times d} - \tilde{V} \tilde{V}^T) x_j \right)
\]

where \(\rho : \mathbb{R}^n \to \mathbb{R}_{>0}, \rho : (\varepsilon_1, \ldots, \varepsilon_d) \mapsto \sum_{j=1}^{n} \rho_{\text{Geman-McClure}}(\varepsilon_1)\).

- They used a standard optimization method over \(\mathbb{R}^k\) with some good tricks to handle the orthonormality constraint on \(\tilde{V}\) (it would have been better to use optimization methods that work directly on the Stiefel manifold) and the lack of convexity.

- This does not scale well (their initialization required SVD)
Decomposed $X$ as $X = L + S$ where $L$ is low-rank and $S$ is sparse (i.e., it contains the outliers).

They elegantly showed that under very broad assumptions this can be solved as a convex problem, and provided several algorithms, where the so-called Inexact Augmented Lagrange Multiplier (ALM) method is most commonly used.

They repeatedly compute SVD – so this scales poorly.
PCA is a process that takes points (apples) as input and returns a subspace (oranges)

Wouldn’t it be better to start and end with the same type of data?
Observations by Hauberg et al., 2014

- Each non-zero data point defines a 1D subspace.

Example

Figure from Hauberg et al., 2014
The average subspace corresponds to a principal component.

Figure 1. *Left:* A zero-mean dataset represented as a set of points. *Center:* The same data represented as a set of one-dimensional subspaces. *Right:* The blue dotted subspace is the average.
The latter observation is super useful since

- Averaging subspaces is easy, fast, and efficient (linear complexity)
- Same goes for computing robust averages

Putting it differently, it is easier to compute sample means (robust or not) than sample covariances (robust or not).
Data Sampled from a 2D Gaussian
The Corresponding Symmetrized Directional Distribution

Remark

The subfield of statistics that handles distributions for sphere-valued data is called Directional Statistics.
Trimmed Grassmann Average (TGA) Demo

- Original data: \((x_i)_{i=1}^N \subset \mathbb{R}^2\) (more generally, \(\subset \mathbb{R}^n\))
- Assume zero (sample) mean: \(\frac{1}{N} \sum_{i=1}^N x_i = 0_{2 \times 1}\) (more generally, \(0_{n \times 1}\))
Trimmed Grassmann Average (TGA) Demo

**Representation**

- Each $x_i \neq 0_{2 \times 1}$ spans a 1D subspace of $\mathbb{R}^2$; i.e., $\{\alpha x_i\}_{\alpha \in \mathbb{R}}$
- More generally, $\mathbb{R}^n \ni x_i \neq 0_{n \times 1}$ spans a 1D subspace of $\mathbb{R}^n$
- The data in the example below contains outliers (the points in the upper-left part quadrant)
Replace the original data with these 1D subspaces

*i.e.* let’s view our data as $\left(\{\alpha x_i\}_{\alpha \in \mathbb{R}}\right)_{i=1}^N$
Trimmed Grassmann Average (TGA) Demo

*Representation*

- Represent the subspace associated with each $x_i$ via a pair of antipodal points: $[u_i] \triangleq \{u_i, -u_i\}$, where $u_i \triangleq x_i/\|x_i\|$.
- This highlights the fact that $x_i$ and $-x_i$ span the same 1D subspace.
- $u_i$ and $-u_i$ are unit vectors.
We discussed the representation, but have yet to discuss the model. Nonetheless, let us jump to the description of the inference algorithm, and we will return to the model later.
Trimmed Grassmann Average (TGA) Demo

\textit{Inference}

- we are looking for the “best” subspace, \([\nu] = \{\nu, -\nu\}\), where \(\nu\) is a unit vector
- We will later define what “best” means (i.e., what is the model?)
Trimmed Grassmann Average (TGA) Demo

Inference

- Initialize by picking one of the $u_i$’s at random: $v^{[t=0]} = u_i$ for some $i \in (1, \ldots, N)$.
- In the example below, we intentionally use the worst initialization (an outlier).
For each $[u_i] = \{u_i, -u_i\}$, pick the sign such that the vector is closer to $v[t=0]$:

$$\tilde{u}_i^{[t=1]} = \arg \min_{q \in \{u_i, -u_i\}} ||q - v[t=0]||$$
Inference

Sort the $\tilde{u}_i^{[t=1]}$'s by their relative angles (or distances) from $\nu^{[t=0]}$ and denote the ordered points by

$$\tilde{u}^{[t=1]}, \tilde{u}^{[t=1]}_{(1)}, \tilde{u}^{[t=1]}_{(2)}, \ldots, \tilde{u}^{[t=1]}_{(N)}$$
Trimmed Grassmann Average (TGA) Demo

Inference

- Set the updated estimate to the trimmed spherical average (there are several ways to define spherical averages, below is one of them) of the $\tilde{u}_i$’s and normalizing it:

\[
\mu^{[t=1]}_{TA} \triangleq \frac{1}{N - 2N_0} \sum_{i=N_0}^{N-N_0} \tilde{u}^{[t=1]}_i
\]

\[
\nu^{[t=1]} = \frac{\mu^{[t=1]}_{TA}}{\| \mu^{[t=1]}_{TA} \|}
\]
Trimmed Grassmann Average (TGA) Demo

Inference

$\nu^{[t=1]}$
Trimmed Grassmann Average (TGA) Demo

Inference

Now just repeat the process till convergence:

1. $\tilde{u}_i^t = \arg\min_{q \in \{u_i, -u_i\}} ||q - v^{[t-1]}||$

2. $(\tilde{u}^{(1)}_t, \tilde{u}^{(2)}_t, \ldots, \tilde{u}^{(N)}_t) \triangleq$ the $\tilde{u}_i^t$'s sorted by their relative angles from $v^{[t-1]}$

3. Set $\mu_{TA}^{[t]} \triangleq \frac{1}{N-2N_0} \sum_{i=N_0}^{N-N_0} \tilde{u}^{[t]}_{(i)}$ and $v^{[t]} = \mu_{TA}^{[t]} / \|\mu_{TA}^{[t]}\|$

The final estimated 1D subspace is $[v] \triangleq [v^{t_{final}}] = \{v^{t_{final}}, -v^{t_{final}}\}$
Inference

Now just repeat the process till convergence:

1. $\tilde{u}_i^t = \arg\min_{q \in \{u_i, -u_i\}} ||q - v^{t-1}||$
2. $(\tilde{u}_1^t, \tilde{u}_2^t, \ldots, \tilde{u}_N^t) \triangleq$ the $\tilde{u}_i^t$'s sorted by their relative angles from $v^{t-1}$
3. Set $\mu_{TA}^t = \frac{1}{N-2N_0} \sum_{i=N_0}^{N-N_0} \tilde{u}_i^t$ and $v^t = \mu_{TA}^t / ||\mu_{TA}^t||$

The final estimated 1D subspace is $[v] \triangleq [v^{t_{\text{final}}}] = \{v^{t_{\text{final}}}, -v^{t_{\text{final}}}\}$
Trimmed Grassmann Average (TGA) Demo

**Inference**

- Now just repeat the process till convergence:
  1. $\tilde{u}_i^{[t]} = \arg \min_{q \in \{u_i, -u_i\}} ||q - v^{[t-1]}||$
  2. $(\tilde{u}_1^{[t]}, \tilde{u}_2^{[t]}, \ldots, \tilde{u}_N^{[t]})$ $\triangleq$ the $\tilde{u}_i^{[t]}$’s sorted by their relative angles from $v^{[t-1]}$
  3. Set $\mu_{TA}^{[t]} \triangleq \frac{1}{N-2N_0} \sum_{i=N_0}^{N-N_0} \tilde{u}_i^{[t]}$ and $v^{[t]} = \mu_{TA}^{[t]} / ||\mu_{TA}^{[t]}||$

- The final estimated 1D subspace is $[v] \triangleq [v_{t_{final}}] = \{v_{t_{final}}, -v_{t_{final}}\}$
Trimmed Grassmann Average (TGA) Demo

**Inference**

- Now just repeat the process till convergence:
  1. $\tilde{u}_i^t = \arg\min_{q \in \{u_i, -u_i\}} ||q - v^{t-1}||$
  2. $(\tilde{u}^t_1, \tilde{u}^t_2, \ldots, \tilde{u}^t_N) \triangleq$ the $\tilde{u}_i^t$’s sorted by their relative angles from $v^{t-1}$
  3. Set $\mu_{TA}^t \triangleq \frac{1}{N-2N_0} \sum_{i=N_0}^{N-N_0} \tilde{u}^t_i$ and $v^t = \mu_{TA}^t / \|\mu_{TA}^t\|$

- The final estimated 1D subspace is $[v] \triangleq [v^{t_{\text{final}}}] = \{v^{t_{\text{final}}}, -v^{t_{\text{final}}}\}$
Trimmed Grassmann Average

Using Trimmed Grassmann Average for PCA

However, for this to be useful in the context PCA, we need a slight modification; i.e., weighting the points according to their magnitudes. The algorithm then becomes:

1. **initialize:**

   \[ w_i = \text{sign}(u^T_i v^{[t=0]}) \| x_i \| \text{ and } v^{[t=1]} = \frac{1}{\sum_{i=N_0}^{N-N_0} w_i} \sum_{i=N_0}^{N-N_0} w_i u(i) \]

   where the \( u(i) \)'s are sorted according to the distances between the \([u(i)]'s\) and \([v^{[t=0]}]\)

2. **Now iterate:**

   \[ w_i = \text{sign}(u^T_i v^{[t-1]}) \| x_i \| \text{ and } v^{[t]} = \frac{1}{\sum_{i=N_0}^{N-N_0} w_i} \sum_{i=N_0}^{N-N_0} w_i u(i) \]

   where the \( u(i) \)'s are sorted according to the distances between the \([u(i)]'s\) and \([v^{[t-1]}]\)

- To find additional principal components, continue by removing the first component and repeat the process till obtaining \( k \) principal components.
Trimmed Grassmann Average (TGA)

Model

- Suppose all the weights are 1, and suppose there is no trimming. Turns out the Grassmann average approach produces

$$v = \arg \max_{v \in S^{n-1}} \sum_{i=1}^{N} |x_i^T v|$$

- Compare this with the standard PCA:

$$\arg \max_{v \in S^{n-1}} \sum_{i=1}^{N} v^T x_i x_i^T v \underbrace{(x_i^T v)^2}_{(x_i^T v)^2}$$

So even before trimming the GA approach is expected to be more robust than PCA (since $\ell_1$ is more robust than $\ell_2$).
Relation of Grassmann Average to PCA and Gaussian Data

Fact

- For Gaussian data, the expected value of the GA coincides with the expected value of the first Principal Component.
- Although PCA was derived (and is applicable) without assuming a Gaussian model, it turns out that under a Gaussian model, PCA is optimal w.r.t. some information-theoretic criterion.
- Empirically, as the model becomes less and less Gaussian, it seems that GA deteriorates more gracefully than PCA.