3D Geometry for Computer Graphics

Lesson 2: PCA & SVD
Last week - eigendecomposition

- We want to learn how the matrix $A$ works:
Last week - eigendecomposition

- If we look at arbitrary vectors, it doesn’t tell us much.
Spectra and diagonalization

- If $A$ is symmetric, the eigenvectors are orthogonal (and there’s always an eigenbasis).

$$A = U \Lambda U^T$$

$$\Lambda = \begin{pmatrix} \lambda_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \lambda_n \end{pmatrix}$$

$$A u_i = \lambda_i u_i$$
Why SVD…

- Diagonalizable matrix is essentially a scaling.
- Most matrices are not diagonalizable – they do other things along with scaling (such as rotation)

- So, to understand how general matrices behave, only eigenvalues are not enough
- SVD tells us how general linear transformations behave, and other things…
The plan for today

- First we’ll see some applications of PCA – Principal Component Analysis that uses spectral decomposition.
- Then look at the theory.
- SVD
  - Basic intuition
  - Formal definition
  - Applications
PCA – the general idea

- PCA finds an orthogonal basis that best represents given data set.

- The sum of distances\(^2\) from the \(x'\) axis is minimized.
PCA – the general idea

- PCA finds an orthogonal basis that best represents given data set.

- PCA finds a best approximating plane (again, in terms of $\sum \text{distances}^2$)
Application: finding tight bounding box

- An axis-aligned bounding box: agrees with the axes
Application: finding tight bounding box

- Oriented bounding box: we find better axes!
Application: finding tight bounding box

- This is not the optimal bounding box
Application: finding tight bounding box

- Oriented bounding box: we find better axes!
Usage of bounding boxes (bounding volumes)

- Serve as very simple “approximation” of the object
- Fast collision detection, visibility queries
- Whenever we need to know the dimensions (size) of the object

- The models consist of thousands of polygons
- To quickly test that they don’t intersect, the bounding boxes are tested
- Sometimes a hierarchy of BB’s is used
- The tighter the BB – the less “false alarms” we have

Sample
Scanned meshes
Point clouds

- Scanners give us raw point cloud data
- How to compute normals to shade the surface?
Point clouds

- Local PCA, take the third vector
Notations

- Denote our data points by \( \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^d \)

\[
\mathbf{x}_1 = \begin{pmatrix}
  x_1^1 \\
  x_1^2 \\
  \vdots \\
  x_1^d
\end{pmatrix}, \quad \mathbf{x}_2 = \begin{pmatrix}
  x_2^1 \\
  x_2^2 \\
  \vdots \\
  x_2^d
\end{pmatrix}, \quad \ldots, \quad \mathbf{x}_n = \begin{pmatrix}
  x_n^1 \\
  x_n^2 \\
  \vdots \\
  x_n^d
\end{pmatrix}
\]
The origin of the new axes

- The origin is zero-order approximation of our data set (a point)
- It will be the center of mass:
  \[ m = \frac{1}{n} \sum_{i=1}^{n} x_i \]
- It can be shown that:
  \[ m = \arg\min_{x} \sum_{i=1}^{n} \| x_i - x \|^2 \]
Scatter matrix

- Denote $y_i = x_i - m$, $i = 1, 2, \ldots, n$

\[
S = YY^T
\]

where $Y$ is $d \times n$ matrix with $y_k$ as columns ($k = 1, 2, \ldots, n$)

\[
S = \begin{pmatrix}
y_1^1 & y_2^1 & \cdots & y_n^1 \\
y_1^2 & y_2^2 & \cdots & y_n^2 \\
\vdots & \vdots & \ddots & \vdots \\
y_1^d & y_2^d & \cdots & y_n^d \\
\end{pmatrix}
\begin{pmatrix}
y_1^1 \\
y_2^1 \\
\vdots \\
y_n^1 \\
\end{pmatrix}
\begin{pmatrix}
y_1^2 \\
y_2^2 \\
\vdots \\
y_n^2 \\
\end{pmatrix}
\begin{pmatrix}
y_1^d \\
y_2^d \\
\vdots \\
y_n^d \\
\end{pmatrix}
\]

\[
Y \quad Y^T
\]
Variance of projected points

- In a way, $S$ measures variance (= scatterness) of the data in different directions.
- Let’s look at a line $L$ through the center of mass $m$, and project our points $x_i$ onto it. The variance of the projected points $x'_i$ is:

$$\text{var}(L) = \frac{1}{n} \sum_{i=1}^{n} \| x'_i - m \|^2$$
Variance of projected points

- Given a direction $\mathbf{v}$, $||\mathbf{v}|| = 1$, the projection of $\mathbf{x}_i$ onto $L = \mathbf{m} + \mathbf{v}t$ is:

$$||\mathbf{x}'_i - \mathbf{m}|| = \langle \mathbf{v}, \mathbf{x}_i - \mathbf{m} \rangle / ||\mathbf{v}|| = \langle \mathbf{v}, \mathbf{y}_i \rangle = \mathbf{v}^T \mathbf{y}_i$$
Variance of projected points

So,

\[
\text{var}(L) = \frac{1}{n} \sum_{i=1}^{n} \| x_i' - m \|^2 = \frac{1}{n} \sum_{i=1}^{n} (v^T y_i)^2 = \frac{1}{n} \| v^T Y \|^2 = \frac{1}{n} \| Y^T v \|^2 = \frac{1}{n} < Y^T v, Y^T v > = \frac{1}{n} v^T YY^T v = \frac{1}{n} v^T S v = \frac{1}{n} < S v, v >
\]
Directions of maximal variance

- So, we have: $\text{var}(L) = \langle S\mathbf{v}, \mathbf{v} \rangle$

- **Theorem:**

  Let $f : \{\mathbf{v} \in \mathbb{R}^d \mid \|\mathbf{v}\| = 1\} \rightarrow \mathbb{R}$,
  $$f(\mathbf{v}) = \langle S\mathbf{v}, \mathbf{v} \rangle$$ (and $S$ is a symmetric matrix).

  Then, the extrema of $f$ are attained at the eigenvectors of $S$.

- So, eigenvectors of $S$ are directions of maximal/minimal variance!
Summary so far

- We take the centered data vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n \in \mathbb{R}^d$
- Construct the scatter matrix $S = XX^T$
- $S$ measures the variance of the data points
- Eigenvectors of $S$ are directions of maximal variance.
Scatter matrix - eigendecomposition

- $S$ is symmetric

$\Rightarrow$ $S$ has eigendecomposition: $S = V \Lambda V^T$

The eigenvectors form orthogonal basis
Principal components

- Eigenvectors that correspond to big eigenvalues are the directions in which the data has strong components (= large variance).
- If the eigenvalues are more or less the same – there is no preferable direction.
- Note: the eigenvalues are always non-negative.
Principal components

- There's no preferable direction
- $S$ looks like this:

$$ V \begin{pmatrix} \lambda & \lambda \\ \lambda & \lambda \end{pmatrix} V^T $$

- Any vector is an eigenvector

- There is a clear preferable direction
- $S$ looks like this:

$$ V \begin{pmatrix} \lambda & \mu \\ \mu & \mu \end{pmatrix} V^T $$

- $\mu$ is close to zero, much smaller than $\lambda$. 
How to use what we got

- For finding oriented bounding box – we simply compute the bounding box with respect to the axes defined by the eigenvectors. The origin is at the mean point $m$. 
For approximation

This line segment approximates the original data set

The projected data set approximates the original data set
For approximation

- In general dimension $d$, the eigenvalues are sorted in descending order:

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$$

- The eigenvectors are sorted accordingly.

- To get an approximation of dimension $d' < d$, we take the $d'$ first eigenvectors and look at the subspace they span ($d' = 1$ is a line, $d' = 2$ is a plane...
For approximation

To get an approximating set, we project the original data points onto the chosen subspace:

\[ \mathbf{x}_i = \mathbf{m} + \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \ldots + \alpha_{d'} \mathbf{v}_{d'} + \ldots + \alpha_d \mathbf{v}_d \]

Projection:

\[ \mathbf{x}_i' = \mathbf{m} + \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \ldots + \alpha_{d'} \mathbf{v}_{d'} + 0 \cdot \mathbf{v}_{d'+1} + \ldots + 0 \cdot \mathbf{v}_d \]
SVD
We want to know what a linear transformation $A$ does

Need some simple and “comprehendible” representation of the matrix of $A$.

Let’s look what $A$ does to some vectors

Since $A(\alpha \mathbf{v}) = \alpha A(\mathbf{v})$, it’s enough to look at vectors $\mathbf{v}$ of unit length
The geometry of linear transformations

- A linear (non-singular) transform $A$ always takes hyper-spheres to hyper-ellipses.
The geometry of linear transformations

- Thus, one good way to understand what $A$ does is to find which vectors are mapped to the “main axes” of the ellipsoid.

![Diagram of linear transformation]
Geometric analysis of linear transformations

- If we are lucky: \( A = V \Lambda V^T, \quad V \) orthogonal (true if \( A \) is symmetric)
- The eigenvectors of \( A \) are the axes of the ellipse
Symmetric matrix: eigen decomposition

- In this case $A$ is just a scaling matrix. The eigen decomposition of $A$ tells us which orthogonal axes it scales, and by how much:

\[ A = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \end{bmatrix} \begin{bmatrix} \lambda_1 & \lambda_2 & \cdots & \lambda_n \end{bmatrix}^{T} \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \end{bmatrix} \]

\[ A \mathbf{v}_i = \lambda_i \mathbf{v}_i \]
General linear transformations: SVD

- In general, $A$ will also contain rotations, not just scales:

$$A = U \Sigma V^T$$

$$A = \begin{bmatrix} u_1 & u_2 & \ldots & u_n \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \ldots \\ \sigma_n \end{bmatrix} \begin{bmatrix} v_1 & v_2 & \ldots & v_n \end{bmatrix}^T$$
General linear transformations: SVD

\[ AV = U \Sigma \]

\[ A \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix} = \begin{bmatrix} u_1 & u_2 & \cdots & u_n \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_n \end{bmatrix} \]

\[ A v_i = \sigma_i u_i, \quad \sigma_i \geq 0 \]
SVD more formally

- SVD exists for any matrix
- Formal definition:
  - For square matrices $A \in \mathbb{R}^{n \times n}$, there exist orthogonal matrices $U, V \in \mathbb{R}^{n \times n}$ and a diagonal matrix $\Sigma$, such that all the diagonal values $\sigma_i$ of $\Sigma$ are non-negative and
  $$A = U \Sigma V^T$$
The diagonal values of $\Sigma$ ($\sigma_1, \ldots, \sigma_n$) are called the singular values. It is accustomed to sort them: $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$.

The columns of $U (u_1, \ldots, u_n)$ are called the left singular vectors. They are the axes of the ellipsoid.

The columns of $V (v_1, \ldots, v_n)$ are called the right singular vectors. They are the preimages of the axes of the ellipsoid.

$$A = U \Sigma V^T$$
Reduced SVD

- For rectangular matrices, we have two forms of SVD. The reduced SVD looks like this:
  - The columns of $U$ are orthonormal
  - Cheaper form for computation and storage

\[
A = U \Sigma V^T
\]
Full SVD

- We can complete \( U \) to a full orthogonal matrix and pad \( \Sigma \) by zeros accordingly.

\[
A = U \Sigma V^T
\]
Some history

- SVD was discovered by the following people:

  - E. Beltrami (1835 – 1900)
  - M. Jordan (1838 – 1922)
  - J. Sylvester (1814 – 1897)
  - E. Schmidt (1876-1959)
  - H. Weyl (1885-1955)
SVD is the “working horse” of linear algebra

- There are numerical algorithms to compute SVD. Once you have it, you have many things:
  - Matrix inverse $\rightarrow$ can solve square linear systems
  - Numerical rank of a matrix
  - Can solve least-squares systems
  - PCA
  - Many more…
Matrix inverse and solving linear systems

- **Matrix inverse:** \( A = U \sum V^T \)

\[
A^{-1} = (U \sum V^T)^{-1} = (V^T)^{-1} \sum^{-1} U^{-1} =
\]

\[
= V \begin{bmatrix}
\frac{1}{\sigma_1} & & \\
& \ddots & \\
& & \frac{1}{\sigma_n}
\end{bmatrix} U^T
\]

- So, to solve \( Ax = b \)

\[
x = V \sum^{-1} U^T b
\]
Matrix rank

- The rank of $A$ is the number of non-zero singular values
Numerical rank

- If there are very small singular values, then $A$ is close to being singular. We can set a threshold $t$, so that $\text{numeric\_rank}(A) = \#\{\sigma_i | \sigma_i > t\}$

- If $\text{rank}(A) < n$ then $A$ is singular. It maps the entire space $\mathbb{R}^n$ onto some subspace, like a plane (so $A$ is some sort of projection).
Back to PCA

- We wanted to find principal components
PCA

- Move the center of mass to the origin

\[ p_i' = p_i - m \]
PCA

- Constructed the matrix $X$ of the data points.

$$X = \begin{bmatrix}
p'_1 & p'_2 & \cdots & p'_n \\
\end{bmatrix}$$

- The principal axes are eigenvectors of $S = XX^T$

$$XX^T = S = U \begin{bmatrix}
\lambda_1 \\
\vdots \\
\lambda_d \\
\end{bmatrix} U^T$$
PCA

We can compute the principal components by SVD of $X$:

$$X = U \Sigma V^T$$

$$XX^T = U \Sigma V^T (U \Sigma V^T)^T =$$

$$= U \Sigma V^T V \Sigma^T U^T = U \tilde{\Sigma}^2 U^T$$

Thus, the left singular vectors of $X$ are the principal components! We sort them by the size of the singular values of $X$. 

Shape matching

- We have two objects in correspondence
- Want to find the rigid transformation that aligns them
Shape matching

- When the objects are aligned, the lengths of the connecting lines are small.
Shape matching – formalization

- Align two point sets
  \[ P = \{ p_1, \ldots, p_n \} \quad \text{and} \quad Q = \{ q_1, \ldots, q_n \}. \]

- Find a translation vector \( t \) and rotation matrix \( R \) so that:
  \[ \sum_{i=1}^{n} \left\| p_i - (Rq_i + t) \right\|^2 \quad \text{is minimized} \]
Shape matching – solution

- Turns out we can solve the translation and rotation separately.
- Theorem: if \((R, t)\) is the optimal transformation, then the points \(\{p_i\}\) and \(\{Rq_i + t\}\) have the same centers of mass.

\[
\begin{align*}
p &= \frac{1}{n} \sum_{i=1}^{n} p_i \\
q &= \frac{1}{n} \sum_{i=1}^{n} q_i
\end{align*}
\]

\[
p = \frac{1}{n} \sum_{i=1}^{n} (Rq_i + t)
\]

\[
\Downarrow
\]

\[
p = R \left( \frac{1}{n} \sum_{i=1}^{n} q_i \right) + t = Rq + t
\]

\[
t = p - Rq
\]
Finding the rotation $R$

- To find the optimal $R$, we bring the centroids of both point sets to the origin:

$$p'_i = p_i - p \quad q'_i = q_i - q$$

- We want to find $R$ that minimizes

$$\sum_{i=1}^{n} \left\| p'_i - R q'_i \right\|^2$$
Finding the rotation $R$

\[
\sum_{i=1}^{n} \|p'_i - R q'_i\|^2 = \sum_{i=1}^{n} (p'_i - R q'_i)^T (p'_i - R q'_i) =
\]

\[
= \sum_{i=1}^{n} \left( p'^T_i p'_i - p'^T_i R q'_i - q'^T_i R^T p'_i + q'^T_i R^T R q'_i \right)
\]

These terms do not depend on $R$, so we can ignore them in the minimization.
Finding the rotation \( R \)

\[
\min \sum_{i=1}^{n} \left( -p_i'^T R q_i' - q_i'^T R^T p_i' \right) = \max \sum_{i=1}^{n} \left( p_i'^T R q_i' + q_i'^T R^T p_i' \right).
\]

\( q_i'^T R^T p_i' = \left( q_i'^T R^T p_i' \right)^T = p_i'^T R q_i' \)

\[\Rightarrow \max \sum_{i=1}^{n} \left( 2p_i'^T R q_i' \right) = \max \sum_{i=1}^{n} \left( p_i'^T R q_i' \right)\]

this is a scalar
Finding the rotation $R$

\[
\sum_{i=1}^{n} \left( p_i^T R q_i' \right) = \text{Trace} \left( \sum_{i=1}^{n} R q_i' p_i^T \right) = \text{Trace} \left( R \sum_{i=1}^{n} q_i' p_i^T \right)
\]

\[
H = \sum_{i=1}^{n} q_i' p_i^T
\]

\[
\text{Trace}(A) = \sum_{i=1}^{n} A_{ii}
\]
Finding the rotation $R$

- So, we want to find $R$ that maximizes $\text{Trace}(RH)$

- Theorem: if $M$ is symmetric positive definite (all eigenvalues of $M$ are positive) and $B$ is any orthogonal matrix then $\text{Trace}(M) \geq \text{Trace}(BM)$

- So, let’s find $R$ so that $RH$ is symmetric positive definite. Then we know for sure that $\text{Trace}(RH)$ is maximal.
Finding the rotation $R$

- This is easy! Compute SVD of $H$:
  \[ H = U \Sigma V^T \]

- Define $R$:
  \[ R = VU^T \]

- Check $RH$:
  \[ RH = (VU^T)(U \Sigma V^T) = V \Sigma V^T \]

This is a symmetric matrix, its eigenvalues are $\sigma_i \geq 0$
So $RH$ is positive!
Summary of rigid alignment:

- Translate the input points to the centroids:
  \[ p_i' = p_i - p \quad q_i' = q_i - q \]

- Compute the “covariance matrix”
  \[ H = \sum_{i=1}^{n} q_i' p_i'^T \]

- Compute the SVD of \( H \):
  \[ H = U \Sigma V^T \]

- The optimal rotation is
  \[ R = V U^T \]

- The translation vector is
  \[ t = p - Rq \]
Complexity

- Numerical SVD is an expensive operation
- We always need to pay attention to the dimensions of the matrix we’re applying SVD to.
Small somewhat related example
The End