Dimensionality Reduction

Machine Learning

Overview

Overview

1. Singular Value Decomposition (SVD).

Overview

1. Singular Value Decomposition (SVD).
2. Moore-Penrose Pseudoinverse.

Overview

1. Singular Value Decomposition (SVD).
2. Moore-Penrose Pseudoinverse.
3. Least-squares and least-norm problems using Moore-Penrose pseudoinverse.

Overview

1. Singular Value Decomposition (SVD).
2. Moore-Penrose Pseudoinverse.
3. Least-squares and least-norm problems using Moore-Penrose pseudoinverse.
4. Principal Component Analysis (PCA).

Overview

1. Singular Value Decomposition (SVD).
2. Moore-Penrose Pseudoinverse.
3. Least-squares and least-norm problems using Moore-Penrose pseudoinverse.
4. Principal Component Analysis (PCA).
5. An alternate perspective on PCA.

Overview

1. Singular Value Decomposition (SVD).
2. Moore-Penrose Pseudoinverse.
3. Least-squares and least-norm problems using Moore-Penrose pseudoinverse.
4. Principal Component Analysis (PCA).
5. An alternate perspective on PCA.
6. Other dimensionality reduction algorithms.

Singular Value Decomposition

Definition (Singular Value Decomposition)

Singular Value Decomposition

Definition (Singular Value Decomposition)

Given a matrix $M \in \mathbb{C}^{n \times d}$, the **(full) SVD** of M writes:

Singular Value Decomposition

Definition (Singular Value Decomposition)

Given a matrix $M \in \mathbb{C}^{n \times d}$, the **(full) SVD** of M writes:

$$M$$

Singular Value Decomposition

Definition (Singular Value Decomposition)

Given a matrix $M \in \mathbb{C}^{n\times d}$, the **(full) SVD** of M writes:

$$M = U\Sigma V^*.$$

Singular Value Decomposition

Definition (Singular Value Decomposition)

Given a matrix $M \in \mathbb{C}^{n \times d}$, the **(full) SVD** of M writes:

$$M = U \Sigma V^*.$$

- $U \in \mathrm{SU}(n)$ is the matrix of eigenvectors of MM^* (ordered by size of eigenvalue).

Singular Value Decomposition

Definition (Singular Value Decomposition)

Given a matrix $M \in \mathbb{C}^{n\times d}$, the **(full) SVD** of M writes:

$$M = U\Sigma V^*.$$

- $U \in \mathrm{SU}(n)$ is the matrix of eigenvectors of MM^* (ordered by size of eigenvalue).
- $V \in \mathrm{SU}(d)$ is the matrix of eigenvectors of M^*M (ordered by size of eigenvalue).

Singular Value Decomposition

Definition (Singular Value Decomposition)

Given a matrix $M \in \mathbb{C}^{n \times d}$, the **(full) SVD** of M writes:

$$M = U \Sigma V^*.$$

- $U \in \mathrm{SU}(n)$ is the matrix of eigenvectors of MM^* (ordered by size of eigenvalue).
- $V \in \mathrm{SU}(d)$ is the matrix of eigenvectors of M^*M (ordered by size of eigenvalue).
- $\Sigma \in \mathbb{C}^{n \times d}$ is the matrix which has zeros outside the diagonal $(i = j)$ and *square roots of* corresponding eigenvalues on the diagonal.

Singular Value Decomposition

Definition (Singular Value Decomposition)

Given a matrix $M \in \mathbb{C}^{n\times d}$, the **(full) SVD** of M writes:

$$M = U\Sigma V^*.$$

- $U \in \mathrm{SU}(n)$ is the matrix of eigenvectors of MM^* (ordered by size of eigenvalue).
- $V \in \mathrm{SU}(d)$ is the matrix of eigenvectors of M^*M (ordered by size of eigenvalue).
- $\Sigma \in \mathbb{C}^{n\times d}$ is the matrix which has zeros outside the diagonal $(i = j)$ and *square roots of* corresponding eigenvalues on the diagonal. (These are called **singular values**.)

Definition (Singular Value Decomposition)

Given a matrix $M \in \mathbb{C}^{n\times d}$, the **(full) SVD** of M writes:

$$M = U\Sigma V^*.$$

- $U \in \mathrm{SU}(n)$ is the matrix of eigenvectors of MM^* (ordered by size of eigenvalue).
- $V \in \mathrm{SU}(d)$ is the matrix of eigenvectors of M^*M (ordered by size of eigenvalue).
- $\Sigma \in \mathbb{C}^{n\times d}$ is the matrix which has zeros outside the diagonal $(i = j)$ and *square roots of* corresponding eigenvalues on the diagonal. (These are called **singular values**.)

Singular Value Decomposition

Definition (Singular Value Decomposition)

Given a matrix $M \in \mathbb{C}^{n \times d}$, the **(full) SVD** of M writes:

$$M = U \Sigma V^*.$$

- $U \in \mathrm{SU}(n)$ is the matrix of eigenvectors of MM^* (ordered by size of eigenvalue).
- $V \in \mathrm{SU}(d)$ is the matrix of eigenvectors of M^*M (ordered by size of eigenvalue).
- $\Sigma \in \mathbb{C}^{n \times d}$ is the matrix which has zeros outside the diagonal $(i = j)$ and *square roots of* corresponding eigenvalues on the diagonal. (These are called **singular values**.)

- The *#1 most important* idea in data science.

Singular Value Decomposition

Definition (Singular Value Decomposition)

Given a matrix $M \in \mathbb{C}^{n\times d}$, the **(full) SVD** of M writes:

$$M = U\Sigma V^*.$$

- $U \in \mathrm{SU}(n)$ is the matrix of eigenvectors of MM^* (ordered by size of eigenvalue).
- $V \in \mathrm{SU}(d)$ is the matrix of eigenvectors of M^*M (ordered by size of eigenvalue).
- $\Sigma \in \mathbb{C}^{n\times d}$ is the matrix which has zeros outside the diagonal $(i = j)$ and *square roots of* corresponding eigenvalues on the diagonal. (These are called **singular values**.)

- The *#1 most important* idea in data science.
- Eigenvectors tell us a lot of information about the structure of the matrix.

Singular Value Decomposition

Definition (Singular Value Decomposition)

Given a matrix $M \in \mathbb{C}^{n\times d}$, the **(full) SVD** of M writes:

$$M = U\Sigma V^*.$$

- $U \in \mathrm{SU}(n)$ is the matrix of eigenvectors of MM^* (ordered by size of eigenvalue).
- $V \in \mathrm{SU}(d)$ is the matrix of eigenvectors of M^*M (ordered by size of eigenvalue).
- $\Sigma \in \mathbb{C}^{n\times d}$ is the matrix which has zeros outside the diagonal $(i = j)$ and *square roots of* corresponding eigenvalues on the diagonal. (These are called **singular values**.)

- The *#1 most important* idea in data science.
- Eigenvectors tell us a lot of information about the structure of the matrix.
- Generalization of diagonalization for non-square matrices.

Singular Value Decomposition

Singular Value Decomposition

Singular Value Decomposition

- For a matrix $M \in \mathbb{C}^{n \times d}$ with $\operatorname{rank}(M) = r \leq \min\{n, d\}$, we know that MM^* and M^*M might have some zero eigenvalues.

Singular Value Decomposition

- For a matrix $M \in \mathbb{C}^{n \times d}$ with $\mathrm{rank}(M) = r \leq \min\{n, d\}$, we know that MM^* and M^*M might have some zero eigenvalues. (In particular, exactly r *nonzero* eigenvalues.)

Singular Value Decomposition

- For a matrix $M \in \mathbb{C}^{n \times d}$ with $\operatorname{rank}(M) = r \leq \min\{n, d\}$, we know that MM^* and M^*M might have some zero eigenvalues. (In particular, exactly r *nonzero* eigenvalues.)
- Suppose $U = \begin{bmatrix} U_r & U_{n-r} \end{bmatrix}$ (taking first r columns) and $V = \begin{bmatrix} V_r & V_{d-r} \end{bmatrix}$, so

Singular Value Decomposition

- For a matrix $M \in \mathbb{C}^{n\times d}$ with $\operatorname{rank}(M) = r \leq \min\{n, d\}$, we know that MM^* and M^*M might have some zero eigenvalues. (In particular, exactly r *nonzero* eigenvalues.)
- Suppose $U = \begin{bmatrix} U_r & U_{n-r} \end{bmatrix}$ (taking first r columns) and $V = \begin{bmatrix} V_r & V_{d-r} \end{bmatrix}$, so

$$M$$

Singular Value Decomposition

- For a matrix $M \in \mathbb{C}^{n\times d}$ with $\operatorname{rank}(M) = r \leq \min\{n, d\}$, we know that MM^* and M^*M might have some zero eigenvalues. (In particular, exactly r *nonzero* eigenvalues.)
- Suppose $U = \begin{bmatrix} U_r & U_{n-r} \end{bmatrix}$ (taking first r columns) and $V = \begin{bmatrix} V_r & V_{d-r} \end{bmatrix}$, so

$$M = U\Sigma V^*$$

Singular Value Decomposition

- For a matrix $M \in \mathbb{C}^{n \times d}$ with $\operatorname{rank}(M) = r \leq \min\{n, d\}$, we know that MM^* and M^*M might have some zero eigenvalues. (In particular, exactly r *nonzero* eigenvalues.)
- Suppose $U = \begin{bmatrix} U_r & U_{n-r} \end{bmatrix}$ (taking first r columns) and $V = \begin{bmatrix} V_r & V_{d-r} \end{bmatrix}$, so

$$M = U\Sigma V^* = \begin{bmatrix} U_r & U_{n-r} \end{bmatrix} \begin{bmatrix} \Sigma_r & 0_{r\times(d-r)} \\ 0_{(n-r)\times r} & 0_{(n-r)\times(d-r)} \end{bmatrix} \begin{bmatrix} V_r^* \\ V_{d-r}^* \end{bmatrix}$$

Singular Value Decomposition

- For a matrix $M \in \mathbb{C}^{n\times d}$ with $\operatorname{rank}(M) = r \leq \min\{n, d\}$, we know that MM^* and M^*M might have some zero eigenvalues. (In particular, exactly r *nonzero* eigenvalues.)
- Suppose $U = \begin{bmatrix} U_r & U_{n-r} \end{bmatrix}$ (taking first r columns) and $V = \begin{bmatrix} V_r & V_{d-r} \end{bmatrix}$, so

$$\begin{aligned} M &= U\Sigma V^* = \begin{bmatrix} U_r & U_{n-r} \end{bmatrix} \begin{bmatrix} \Sigma_r & 0_{r\times(d-r)} \\ 0_{(n-r)\times r} & 0_{(n-r)\times(d-r)} \end{bmatrix} \begin{bmatrix} V_r^* \\ V_{d-r}^* \end{bmatrix} \\ &= U_r \Sigma_r V_r^* \end{aligned}$$

Singular Value Decomposition

- For a matrix $M \in \mathbb{C}^{n\times d}$ with $\operatorname{rank}(M) = r \leq \min\{n, d\}$, we know that MM^* and M^*M might have some zero eigenvalues. (In particular, exactly r *nonzero* eigenvalues.)
- Suppose $U = \begin{bmatrix} U_r & U_{n-r} \end{bmatrix}$ (taking first r columns) and $V = \begin{bmatrix} V_r & V_{d-r} \end{bmatrix}$, so

$$
\begin{aligned}
M &= U\Sigma V^* = \begin{bmatrix} U_r & U_{n-r} \end{bmatrix} \begin{bmatrix} \Sigma_r & 0_{r\times(d-r)} \\ 0_{(n-r)\times r} & 0_{(n-r)\times(d-r)} \end{bmatrix} \begin{bmatrix} V_r^* \\ V_{d-r}^* \end{bmatrix} \\
&= U_r \Sigma_r V_r^*
\end{aligned}
$$

This is the **compact SVD** of M.

Singular Value Decomposition

- For a matrix $M \in \mathbb{C}^{n \times d}$ with $\operatorname{rank}(M) = r \leq \min\{n, d\}$, we know that MM^* and M^*M might have some zero eigenvalues. (In particular, exactly r *nonzero* eigenvalues.)
- Suppose $U = \begin{bmatrix} U_r & U_{n-r} \end{bmatrix}$ (taking first r columns) and $V = \begin{bmatrix} V_r & V_{d-r} \end{bmatrix}$, so

$$\begin{aligned} M &= U\Sigma V^* = \begin{bmatrix} U_r & U_{n-r} \end{bmatrix} \begin{bmatrix} \Sigma_r & 0_{r\times(d-r)} \\ 0_{(n-r)\times r} & 0_{(n-r)\times(d-r)} \end{bmatrix} \begin{bmatrix} V_r^* \\ V_{d-r}^* \end{bmatrix} \\ &= U_r \Sigma_r V_r^* \end{aligned}$$

  This is the **compact SVD** of M.
- If we note that Σ_r is diagonal, we can write (for u_i, v_i i^{th} columns of U, V):

Singular Value Decomposition

- For a matrix $M \in \mathbb{C}^{n \times d}$ with $\operatorname{rank}(M) = r \leq \min\{n, d\}$, we know that MM^* and M^*M might have some zero eigenvalues. (In particular, exactly r *nonzero* eigenvalues.)
- Suppose $U = \begin{bmatrix} U_r & U_{n-r} \end{bmatrix}$ (taking first r columns) and $V = \begin{bmatrix} V_r & V_{d-r} \end{bmatrix}$, so

$$\begin{aligned} M &= U\Sigma V^* = \begin{bmatrix} U_r & U_{n-r} \end{bmatrix} \begin{bmatrix} \Sigma_r & 0_{r \times (d-r)} \\ 0_{(n-r)\times r} & 0_{(n-r)\times(d-r)} \end{bmatrix} \begin{bmatrix} V_r^* \\ V_{d-r}^* \end{bmatrix} \\ &= U_r \Sigma_r V_r^* \end{aligned}$$

  This is the **compact SVD** of M.
- If we note that Σ_r is diagonal, we can write (for u_i, v_i i^{th} columns of U, V):

$$M$$

Singular Value Decomposition

- For a matrix $M \in \mathbb{C}^{n\times d}$ with $\operatorname{rank}(M) = r \leq \min\{n, d\}$, we know that MM^* and M^*M might have some zero eigenvalues. (In particular, exactly r *nonzero* eigenvalues.)
- Suppose $U = \begin{bmatrix} U_r & U_{n-r} \end{bmatrix}$ (taking first r columns) and $V = \begin{bmatrix} V_r & V_{d-r} \end{bmatrix}$, so

$$\begin{aligned} M &= U\Sigma V^* = \begin{bmatrix} U_r & U_{n-r} \end{bmatrix} \begin{bmatrix} \Sigma_r & 0_{r\times(d-r)} \\ 0_{(n-r)\times r} & 0_{(n-r)\times(d-r)} \end{bmatrix} \begin{bmatrix} V_r^* \\ V_{d-r}^* \end{bmatrix} \\ &= U_r\Sigma_r V_r^* \end{aligned}$$

  This is the **compact SVD** of M.
- If we note that Σ_r is diagonal, we can write (for u_i, v_i i^{th} columns of U, V):

$$M = U_r\Sigma_r V_r^*$$

Singular Value Decomposition

- For a matrix $M \in \mathbb{C}^{n \times d}$ with $\operatorname{rank}(M) = r \le \min\{n, d\}$, we know that MM^* and M^*M might have some zero eigenvalues. (In particular, exactly r *nonzero* eigenvalues.)
- Suppose $U = \begin{bmatrix} U_r & U_{n-r} \end{bmatrix}$ (taking first r columns) and $V = \begin{bmatrix} V_r & V_{d-r} \end{bmatrix}$, so

$$\begin{aligned} M &= U \Sigma V^* = \begin{bmatrix} U_r & U_{n-r} \end{bmatrix} \begin{bmatrix} \Sigma_r & 0_{r \times (d-r)} \\ 0_{(n-r) \times r} & 0_{(n-r) \times (d-r)} \end{bmatrix} \begin{bmatrix} V_r^* \\ V_{d-r}^* \end{bmatrix} \\ &= U_r \Sigma_r V_r^* \end{aligned}$$

  This is the **compact SVD** of M.
- If we note that Σ_r is diagonal, we can write (for u_i, v_i i^{th} columns of U, V):

$$M = U_r \Sigma_r V_r^* = \sum_{i=1}^{r} \sigma_i u_i v_i^*$$

Singular Value Decomposition

- For a matrix $M \in \mathbb{C}^{n\times d}$ with $\operatorname{rank}(M) = r \le \min\{n, d\}$, we know that MM^* and M^*M might have some zero eigenvalues. (In particular, exactly r *nonzero* eigenvalues.)
- Suppose $U = \begin{bmatrix} U_r & U_{n-r} \end{bmatrix}$ (taking first r columns) and $V = \begin{bmatrix} V_r & V_{d-r} \end{bmatrix}$, so

$$
\begin{aligned}
M &= U\Sigma V^* = \begin{bmatrix} U_r & U_{n-r} \end{bmatrix} \begin{bmatrix} \Sigma_r & 0_{r\times(d-r)} \\ 0_{(n-r)\times r} & 0_{(n-r)\times(d-r)} \end{bmatrix} \begin{bmatrix} V_r^* \\ V_{d-r}^* \end{bmatrix} \\
&= U_r \Sigma_r V_r^*
\end{aligned}
$$

  This is the **compact SVD** of M.
- If we note that Σ_r is diagonal, we can write (for u_i, v_i i^{th} columns of U, V):

$$M = U_r \Sigma_r V_r^* = \sum_{i=1}^{r} \sigma_i u_i v_i^*$$

  This is the **outer product form of SVD** for M (since $u_i v_i^* = u_i \otimes v_i$).

Moore-Penrose Pseudoinverse

Moore-Penrose Pseudoinverse

Moore-Penrose Pseudoinverse

- Suppose we have a system $Ax = b$ which we want to solve for x.

Moore-Penrose Pseudoinverse

- Suppose we have a system $Ax = b$ which we want to solve for x.
- If A has an inverse, then $x = A^{-1}b$ is the solution.

Moore-Penrose Pseudoinverse

- Suppose we have a system $Ax = b$ which we want to solve for x.
- If A has an inverse, then $x = A^{-1}b$ is the solution.
- If A is tall, then least squares gives the solution:

Moore-Penrose Pseudoinverse

- Suppose we have a system $Ax = b$ which we want to solve for x.
- If A has an inverse, then $x = A^{-1}b$ is the solution.
- If A is tall, then least squares gives the solution:

$$\widehat{x}$$

Moore-Penrose Pseudoinverse

- Suppose we have a system $Ax = b$ which we want to solve for x.
- If A has an inverse, then $x = A^{-1}b$ is the solution.
- If A is tall, then least squares gives the solution:

$$\widehat{x} = \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \|Ax - b\|_2^2 .$$

Moore-Penrose Pseudoinverse

- Suppose we have a system $Ax = b$ which we want to solve for x.
- If A has an inverse, then $x = A^{-1}b$ is the solution.
- If A is tall, then least squares gives the solution:

$$\widehat{x} = \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \|Ax - b\|_2^2 .$$

- What if A is wide? Then many solutions. The most useful is the one with minimum norm:

Moore-Penrose Pseudoinverse

- Suppose we have a system $Ax = b$ which we want to solve for x.
- If A has an inverse, then $x = A^{-1}b$ is the solution.
- If A is tall, then least squares gives the solution:

$$\widehat{x} = \operatorname*{argmin}_{x \in \mathbb{R}^d} \|Ax - b\|_2^2 .$$

- What if A is wide? Then many solutions. The most useful is the one with minimum norm:

$$\widehat{x}$$

Moore-Penrose Pseudoinverse

- Suppose we have a system $Ax = b$ which we want to solve for x.
- If A has an inverse, then $x = A^{-1}b$ is the solution.
- If A is tall, then least squares gives the solution:

$$\widehat{x} = \operatorname*{argmin}_{x \in \mathbb{R}^d} \|Ax - b\|_2^2 .$$

- What if A is wide? Then many solutions. The most useful is the one with minimum norm:

$$\widehat{x} = \operatorname*{argmin}_{\substack{x \in \mathbb{R}^d \\ Ax = b}} \|x\|_2^2 .$$

Moore-Penrose Pseudoinverse

- Suppose we have a system $Ax = b$ which we want to solve for x.
- If A has an inverse, then $x = A^{-1}b$ is the solution.
- If A is tall, then least squares gives the solution:

$$\widehat{x} = \operatorname*{argmin}_{x \in \mathbb{R}^d} \|Ax - b\|_2^2 .$$

- What if A is wide? Then many solutions. The most useful is the one with minimum norm:

$$\widehat{x} = \operatorname*{argmin}_{\substack{x \in \mathbb{R}^d \\ Ax = b}} \|x\|_2^2 .$$

- Let $A^\dagger = V\Sigma^\dagger U^* = V_r \Sigma_r^{-1} U_r^*$ be the **Moore-Penrose pseudoinverse**. Then $\widehat{x} = A^\dagger b$ is the "canonical" solution to $Ax = b$, and it works for all kinds of A. (Wide, tall, or square).

Moore-Penrose Pseudoinverse

- Suppose we have a system $Ax = b$ which we want to solve for x.
- If A has an inverse, then $x = A^{-1}b$ is the solution.
- If A is tall, then least squares gives the solution:

$$\widehat{x} = \operatorname*{argmin}_{x \in \mathbb{R}^d} \|Ax - b\|_2^2 .$$

- What if A is wide? Then many solutions. The most useful is the one with minimum norm:

$$\widehat{x} = \operatorname*{argmin}_{\substack{x \in \mathbb{R}^d \\ Ax = b}} \|x\|_2^2 .$$

- Let $A^\dagger = V\Sigma^\dagger U^* = V_r \Sigma_r^{-1} U_r^*$ be the **Moore-Penrose pseudoinverse**. Then $\widehat{x} = A^\dagger b$ is the "canonical" solution to $Ax = b$, and it works for all kinds of A. (Wide, tall, or square).
- Here, $\Sigma^\dagger = \begin{bmatrix} \Sigma_r & 0_{r\times(d-r)} \\ 0_{(n-r)\times r} & 0_{(n-r)\times(d-r)} \end{bmatrix}^\dagger = \begin{bmatrix} \Sigma_r^{-1} & 0_{r\times(n-r)} \\ 0_{(d-r)\times r} & 0_{(d-r)\times(n-r)} \end{bmatrix}$.

Moore-Penrose Pseudoinverse

How does it work for least squares?

Moore-Penrose Pseudoinverse

How does it work for least squares?

Moore-Penrose Pseudoinverse

How does it work for least squares?

$$\hat{x}$$

Moore-Penrose Pseudoinverse

How does it work for least squares?

$$\widehat{x} = \operatorname*{argmin}_{x \in \mathbb{R}^d} \|Ax - b\|_2^2$$

How does it work for least squares?

$$\begin{aligned}\widehat{x} &= \operatorname*{argmin}_{x \in \mathbb{R}^d} \|Ax - b\|_2^2 \\ &= \operatorname*{argmin}_{x \in \mathbb{R}^d} \left\|U \Sigma V^* x - b\right\|_2^2\end{aligned}$$

How does it work for least squares?

$$
\begin{aligned}
\widehat{x} &= \operatorname*{argmin}_{x \in \mathbb{R}^d} \|Ax - b\|_2^2 \\
&= \operatorname*{argmin}_{x \in \mathbb{R}^d} \left\|U \Sigma V^* x - b\right\|_2^2 \\
&= \operatorname*{argmin}_{x \in \mathbb{R}^d} \left\|\Sigma V^* x - U^* b\right\|_2^2
\end{aligned}
$$

Moore-Penrose Pseudoinverse

How does it work for least squares?

$$
\begin{aligned}
\widehat{x} &= \underset{x\in\mathbb{R}^d}{\operatorname{argmin}} \|Ax - b\|_2^2 \\
&= \underset{x\in\mathbb{R}^d}{\operatorname{argmin}} \left\|U\Sigma V^* x - b\right\|_2^2 \\
&= \underset{x\in\mathbb{R}^d}{\operatorname{argmin}} \left\|\Sigma V^* x - U^* b\right\|_2^2 \\
&= \underset{x\in\mathbb{R}^d}{\operatorname{argmin}} \left\|\Sigma_r V_r^* x - U_r^* b\right\|_2^2 + \left\|U_{n-r}^* b\right\|_2^2
\end{aligned}
$$

How does it work for least squares?

$$
\begin{aligned}
\widehat{x} &= \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \|Ax - b\|_2^2 \\
&= \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \left\|U \Sigma V^* x - b\right\|_2^2 \\
&= \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \left\|\Sigma V^* x - U^* b\right\|_2^2 \\
&= \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \left\|\Sigma_r V_r^* x - U_r^* b\right\|_2^2 + \left\|U_{n-r}^* b\right\|_2^2 \\
&= \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \left\|\Sigma_r V_r^* x - U_r^* b\right\|_2^2
\end{aligned}
$$

Moore-Penrose Pseudoinverse

How does it work for least squares?

$$
\begin{aligned}
\widehat{x} &= \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \|Ax - b\|_2^2 \\
&= \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \left\|U \Sigma V^* x - b\right\|_2^2 \\
&= \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \left\|\Sigma V^* x - U^* b\right\|_2^2 \\
&= \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \left\|\Sigma_r V_r^* x - U_r^* b\right\|_2^2 + \left\|U_{n-r}^* b\right\|_2^2 \\
&= \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \left\|\Sigma_r V_r^* x - U_r^* b\right\|_2^2 \\
&= V_r \Sigma_r^{-1} U_r^* b
\end{aligned}
$$

Moore-Penrose Pseudoinverse

How does it work for least squares?

$$
\begin{aligned}
\widehat{x} &= \operatorname*{argmin}_{x \in \mathbb{R}^d} \|Ax - b\|_2^2 \\
&= \operatorname*{argmin}_{x \in \mathbb{R}^d} \left\| U \Sigma V^* x - b \right\|_2^2 \\
&= \operatorname*{argmin}_{x \in \mathbb{R}^d} \left\| \Sigma V^* x - U^* b \right\|_2^2 \\
&= \operatorname*{argmin}_{x \in \mathbb{R}^d} \left\| \Sigma_r V_r^* x - U_r^* b \right\|_2^2 + \left\| U_{n-r}^* b \right\|_2^2 \\
&= \operatorname*{argmin}_{x \in \mathbb{R}^d} \left\| \Sigma_r V_r^* x - U_r^* b \right\|_2^2 \\
&= V_r \Sigma_r^{-1} U_r^* b = V \Sigma^{\dagger} U^* b
\end{aligned}
$$

Moore-Penrose Pseudoinverse

How does it work for least squares?

$$\begin{aligned}
\widehat{x} &= \operatorname*{argmin}_{x\in\mathbb{R}^d} \|Ax - b\|_2^2 \\
&= \operatorname*{argmin}_{x\in\mathbb{R}^d} \left\|U\Sigma V^* x - b\right\|_2^2 \\
&= \operatorname*{argmin}_{x\in\mathbb{R}^d} \left\|\Sigma V^* x - U^* b\right\|_2^2 \\
&= \operatorname*{argmin}_{x\in\mathbb{R}^d} \left\|\Sigma_r V_r^* x - U_r^* b\right\|_2^2 + \left\|U_{n-r}^* b\right\|_2^2 \\
&= \operatorname*{argmin}_{x\in\mathbb{R}^d} \left\|\Sigma_r V_r^* x - U_r^* b\right\|_2^2 \\
&= V_r \Sigma_r^{-1} U_r^* b = V\Sigma^\dagger U^* b = A^\dagger b.
\end{aligned}$$

Moore-Penrose Pseudoinverse

How does it work for least squares?

$$
\begin{aligned}
\widehat{x} &= \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \|Ax - b\|_2^2 \\
&= \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \left\|U \Sigma V^* x - b\right\|_2^2 \\
&= \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \left\|\Sigma V^* x - U^* b\right\|_2^2 \\
&= \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \left\|\Sigma_r V_r^* x - U_r^* b\right\|_2^2 + \left\|U_{n-r}^* b\right\|_2^2 \\
&= \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \left\|\Sigma_r V_r^* x - U_r^* b\right\|_2^2 \\
&= V_r \Sigma_r^{-1} U_r^* b = V \Sigma^\dagger U^* b = A^\dagger b.
\end{aligned}
$$

Argument is mostly the same, though a bit linear algebraically complicated, for least-norm.

Principal Component Analysis

Principal Component Analysis

Principal Component Analysis

- Idea: a vector-valued random-variable x with $\mathbb{E}_x[x] = \mu$ has **covariance** equal to

Principal Component Analysis

- Idea: a vector-valued random-variable x with $\mathbb{E}_x[x] = \mu$ has **covariance** equal to

$$\underset{x}{\mathrm{Cov}}(x)$$

Principal Component Analysis

- Idea: a vector-valued random-variable x with $\mathbb{E}_x[x] = \mu$ has **covariance** equal to

$$\underset{x}{\mathrm{Cov}}(x) = \underset{x}{\mathbb{E}}\left[(x - \mu)(x - \mu)^*\right].$$

Principal Component Analysis

- Idea: a vector-valued random-variable x with $\mathbb{E}_x[x] = \mu$ has **covariance** equal to

$$\underset{x}{\operatorname{Cov}}(x) = \underset{x}{\mathbb{E}}\left[(x-\mu)(x-\mu)^*\right].$$

- Say we have some data $s_X = \{x_1, \ldots, x_n\} \sim \mu_X$ (no labels/outputs needed).

Principal Component Analysis

- Idea: a vector-valued random-variable x with $\mathbb{E}_x[x] = \mu$ has **covariance** equal to

$$\underset{x}{\mathrm{Cov}}(x) \;= \underset{x}{\mathbb{E}}\left[(x-\mu)\,(x-\mu)^*\right].$$

- Say we have some data $s_X = \{x_1, \ldots, x_n\} \sim \mu_X$ (no labels/outputs needed).
- Then we can estimate the covariance of a variable distributed according to μ_X by

Principal Component Analysis

- Idea: a vector-valued random-variable x with $\mathbb{E}_x[x] = \mu$ has **covariance** equal to

$$\operatorname*{Cov}_x(x) = \mathop{\mathbb{E}}_x\left[(x-\mu)(x-\mu)^*\right].$$

- Say we have some data $s_X = \{x_1, \ldots, x_n\} \sim \mu_X$ (no labels/outputs needed).
- Then we can estimate the covariance of a variable distributed according to μ_X by

$$\widehat{\Sigma}$$

Principal Component Analysis

- Idea: a vector-valued random-variable x with $\mathbb{E}_x[x] = \mu$ has **covariance** equal to

$$\operatorname*{Cov}_x(x) = \mathop{\mathbb{E}}_x\left[(x-\mu)(x-\mu)^*\right].$$

- Say we have some data $s_X = \{x_1, \ldots, x_n\} \sim \mu_X$ (no labels/outputs needed).
- Then we can estimate the covariance of a variable distributed according to μ_X by

$$\widehat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \widehat{\mu})(x_i - \widehat{\mu})^*$$

Principal Component Analysis

- Idea: a vector-valued random-variable x with $\mathbb{E}_x[x] = \mu$ has **covariance** equal to

$$\operatorname*{Cov}_x(x) = \mathop{\mathbb{E}}_x\left[(x-\mu)(x-\mu)^*\right].$$

- Say we have some data $s_X = \{x_1, \ldots, x_n\} \sim \mu_X$ (no labels/outputs needed).
- Then we can estimate the covariance of a variable distributed according to μ_X by

$$\widehat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \widehat{\mu})(x_i - \widehat{\mu})^* \quad \text{where} \quad \widehat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

Principal Component Analysis

- Idea: a vector-valued random-variable x with $\mathbb{E}_x[x] = \mu$ has **covariance** equal to

$$\operatorname*{Cov}_x(x) = \mathbb{E}_x\big[(x-\mu)(x-\mu)^*\big].$$

- Say we have some data $s_X = \{x_1, \dots, x_n\} \sim \mu_X$ (no labels/outputs needed).
- Then we can estimate the covariance of a variable distributed according to μ_X by

$$\widehat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \widehat{\mu})(x_i - \widehat{\mu})^* \quad \text{where} \quad \widehat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

- Idea: can get covariance in direction v (unit vector) by $\langle \operatorname{Cov}_x(x)\, v, v\rangle$ or $\left\langle \widehat{\Sigma} v, v\right\rangle$.

Principal Component Analysis

Principal Component Analysis

Principal Component Analysis

- Idea: higher covariance in given direction = more information about the distribution of the data.

Principal Component Analysis

- Idea: higher covariance in given direction = more information about the distribution of the data.
- Want to find the (approximate) directions of highest covariance by computing:

Principal Component Analysis

- Idea: higher covariance in given direction = more information about the distribution of the data.
- Want to find the (approximate) directions of highest covariance by computing:

$$v_1$$

Principal Component Analysis

- Idea: higher covariance in given direction = more information about the distribution of the data.
- Want to find the (approximate) directions of highest covariance by computing:

$$v_1 = \underset{\substack{v \in \mathbb{R}^d \\ \|v\|_2^2 = 1}}{\operatorname{argmax}} \left\langle \widehat{\Sigma} v, v \right\rangle$$

Principal Component Analysis

- Idea: higher covariance in given direction = more information about the distribution of the data.
- Want to find the (approximate) directions of highest covariance by computing:

$$v_1 = \underset{\substack{v \in \mathbb{R}^d \\ \|v\|_2^2 = 1}}{\operatorname{argmax}} \left\langle \widehat{\Sigma} v, v \right\rangle$$

$$v_2$$

Principal Component Analysis

- Idea: higher covariance in given direction = more information about the distribution of the data.
- Want to find the (approximate) directions of highest covariance by computing:

$$
\begin{aligned}
v_1 &= \operatorname*{argmax}_{\substack{v \in \mathbb{R}^d \\ \|v\|_2^2 = 1}} \left\langle \widehat{\Sigma} v, v \right\rangle \\
v_2 &= \operatorname*{argmax}_{\substack{v \in \mathbb{R}^d \\ \|v\|_2^2 = 1 \\ v \perp \operatorname{span}(v_1)}} \left\langle \widehat{\Sigma} v, v \right\rangle
\end{aligned}
$$

Principal Component Analysis

- Idea: higher covariance in given direction = more information about the distribution of the data.
- Want to find the (approximate) directions of highest covariance by computing:

$$v_1 = \operatorname*{argmax}_{\substack{v \in \mathbb{R}^d \\ \|v\|_2^2 = 1}} \left\langle \widehat{\Sigma} v, v \right\rangle$$

$$v_2 = \operatorname*{argmax}_{\substack{v \in \mathbb{R}^d \\ \|v\|_2^2 = 1 \\ v \perp \operatorname{span}(v_1)}} \left\langle \widehat{\Sigma} v, v \right\rangle$$

$$\ldots$$

Principal Component Analysis

- Idea: higher covariance in given direction = more information about the distribution of the data.
- Want to find the (approximate) directions of highest covariance by computing:

$$
\begin{aligned}
v_1 &= \underset{\substack{v \in \mathbb{R}^d \\ \|v\|_2^2 = 1}}{\operatorname{argmax}} \left\langle \widehat{\Sigma} v, v \right\rangle \\
v_2 &= \underset{\substack{v \in \mathbb{R}^d \\ \|v\|_2^2 = 1 \\ v \perp \operatorname{span}(v_1)}}{\operatorname{argmax}} \left\langle \widehat{\Sigma} v, v \right\rangle \\
\cdots & \qquad \cdots
\end{aligned}
$$

Principal Component Analysis

- Idea: higher covariance in given direction = more information about the distribution of the data.
- Want to find the (approximate) directions of highest covariance by computing:

$$
\begin{aligned}
v_1 &= \operatorname*{argmax}_{\substack{v \in \mathbb{R}^d \\ \|v\|_2^2 = 1}} \left\langle \widehat{\Sigma} v, v \right\rangle \\
v_2 &= \operatorname*{argmax}_{\substack{v \in \mathbb{R}^d \\ \|v\|_2^2 = 1 \\ v \perp \operatorname{span}(v_1)}} \left\langle \widehat{\Sigma} v, v \right\rangle \\
\cdots & \qquad \cdots \\
v_d &
\end{aligned}
$$

Principal Component Analysis

- Idea: higher covariance in given direction = more information about the distribution of the data.
- Want to find the (approximate) directions of highest covariance by computing:

$$
\begin{aligned}
v_1 &= \operatorname*{argmax}_{\substack{v \in \mathbb{R}^d \\ \|v\|_2^2 = 1}} \left\langle \widehat{\Sigma} v, v \right\rangle \\
v_2 &= \operatorname*{argmax}_{\substack{v \in \mathbb{R}^d \\ \|v\|_2^2 = 1 \\ v \perp \mathrm{span}(v_1)}} \left\langle \widehat{\Sigma} v, v \right\rangle \\
\cdots & \quad \cdots \\
v_d &= \operatorname*{argmax}_{\substack{v \in \mathbb{R}^d \\ \|v\|_2^2 = 1 \\ v \perp \mathrm{span}(v_1, \ldots, v_{d-1})}} \left\langle \widehat{\Sigma} v, v \right\rangle
\end{aligned}
$$

Principal Component Analysis

- Idea: higher covariance in given direction = more information about the distribution of the data.
- Want to find the (approximate) directions of highest covariance by computing:

$$
\begin{aligned}
v_1 &= \operatorname*{argmax}_{\substack{v \in \mathbb{R}^d \\ \|v\|_2^2 = 1}} \left\langle \widehat{\Sigma} v, v \right\rangle \\
v_2 &= \operatorname*{argmax}_{\substack{v \in \mathbb{R}^d \\ \|v\|_2^2 = 1 \\ v \perp \operatorname{span}(v_1)}} \left\langle \widehat{\Sigma} v, v \right\rangle \\
\cdots & \qquad \cdots \\
v_d &= \operatorname*{argmax}_{\substack{v \in \mathbb{R}^d \\ \|v\|_2^2 = 1 \\ v \perp \operatorname{span}(v_1, \ldots, v_{d-1})}} \left\langle \widehat{\Sigma} v, v \right\rangle
\end{aligned}
$$

- But these are just the eigenvectors of $\widehat{\Sigma}$, ordered by their eigenvalues! Called **principal components** of the data.

Principal Component Analysis

- Idea: higher covariance in given direction = more information about the distribution of the data.
- Want to find the (approximate) directions of highest covariance by computing:

$$\begin{aligned}
v_1 &= \underset{\substack{v\in\mathbb{R}^d \\ \|v\|_2^2=1}}{\operatorname{argmax}} \left\langle \widehat{\Sigma} v, v \right\rangle \\
v_2 &= \underset{\substack{v\in\mathbb{R}^d \\ \|v\|_2^2=1 \\ v\perp \operatorname{span}(v_1)}}{\operatorname{argmax}} \left\langle \widehat{\Sigma} v, v \right\rangle \\
\cdots & \qquad \cdots \\
v_d &= \underset{\substack{v\in\mathbb{R}^d \\ \|v\|_2^2=1 \\ v\perp \operatorname{span}(v_1,\ldots,v_{d-1})}}{\operatorname{argmax}} \left\langle \widehat{\Sigma} v, v \right\rangle
\end{aligned}$$

- But these are just the eigenvectors of $\widehat{\Sigma}$, ordered by their eigenvalues! Called **principal components** of the data. For data matrix X we first de-mean X feature-by-feature, then compute $X = U\Sigma V^*$, then use columns of V.

Principal Component Analysis

Principal Component Analysis

Principal Component Analysis

- Idea: by looking only at the directions of greatest covariance, we can keep most of the variability (hence information) in the data, while reducing dimension (sometimes a lot).

Principal Component Analysis

- Idea: by looking only at the directions of greatest covariance, we can keep most of the variability (hence information) in the data, while reducing dimension (sometimes a lot).
- Given an integer $k < d$, let $V_k = \begin{bmatrix} v_1 & \cdots & v_k \end{bmatrix}$.

Principal Component Analysis

- Idea: by looking only at the directions of greatest covariance, we can keep most of the variability (hence information) in the data, while reducing dimension (sometimes a lot).
- Given an integer $k < d$, let $V_k = \begin{bmatrix} v_1 & \cdots & v_k \end{bmatrix}$.
- Then project data point $x \in \mathbb{R}^d$ onto $\mathbb{R}^k$ by $V_k^* x$.

Principal Component Analysis

- Idea: by looking only at the directions of greatest covariance, we can keep most of the variability (hence information) in the data, while reducing dimension (sometimes a lot).
- Given an integer $k < d$, let $V_k = \begin{bmatrix} v_1 & \cdots & v_k \end{bmatrix}$.
- Then project data point $x \in \mathbb{R}^d$ onto $\mathbb{R}^k$ by $V_k^* x$.
- This gives an algorithm, which can be written down formally as:

Principal Component Analysis

- Idea: by looking only at the directions of greatest covariance, we can keep most of the variability (hence information) in the data, while reducing dimension (sometimes a lot).
- Given an integer $k < d$, let $V_k = \begin{bmatrix} v_1 & \cdots & v_k \end{bmatrix}$.
- Then project data point $x \in \mathbb{R}^d$ onto $\mathbb{R}^k$ by $V_k^* x$.
- This gives an algorithm, which can be written down formally as:

PCA

procedure TRAIN(s_{train})

$\quad \widehat{\mu} \leftarrow \frac{1}{|s_{\text{train}}|} \sum_{i=1}^{n} x_i$

$\quad \widehat{\Sigma} \leftarrow \frac{1}{|s_{\text{train}}|} \sum_{i=1}^{n} (x_i - \widehat{\mu})(x_i - \widehat{\mu})^*$

$\quad V \leftarrow \texttt{Eigenvectors}(\widehat{\Sigma})$

$\quad V_k \leftarrow V[:, :k]$

procedure INFERENCE(x)

$\quad$ **return** $V_k^* x$

Other Perspective on PCA

Other Perspective on PCA

Other Perspective on PCA

- The main idea is to keep most information of the data with very little storage.

Other Perspective on PCA

- The main idea is to keep most information of the data with very little storage.
- We would like to approximate the data matrix X using SVD/PCA.

Other Perspective on PCA

- The main idea is to keep most information of the data with very little storage.
- We would like to approximate the data matrix X using SVD/PCA.

Theorem (Eckart-Young-Mirsky Theorem)

SVD gives the best t-rank approximation to X, for all t.
To be precise, define the ***truncated SVD*** $X_t = \sum_{i=1}^{t} \sigma_i u_i v_i^*$ *(equivalently taking first t columns, like compact SVD but with t instead of r). Then*

$$\|X - X_t\|_2 = \inf_{\substack{Y \in \mathbb{C}^{n\times d} \\ \mathrm{rank}(Y) \le t}} \|X - Y\|_2 .$$

Furthermore, the approximation is also true in the operator norm $\|X\|_{\mathrm{op}}$ which is the maximum singular value of X:

$$\|X - X_t\|_{\mathrm{op}} = \inf_{\substack{Y \in \mathbb{C}^{n\times d} \\ \mathrm{rank}(Y) \le t}} \|X - Y\|_{\mathrm{op}} .$$

Other Dimensionality Reduction Algorithms

Other Dimensionality Reduction Algorithms

Other Dimensionality Reduction Algorithms

By no means an exhaustive list:

Other Dimensionality Reduction Algorithms

By no means an exhaustive list:

- Nonlinear/Robust PCA: coming soon!

Other Dimensionality Reduction Algorithms

By no means an exhaustive list:

- Nonlinear/Robust PCA: coming soon!
- Johnson-Lindenstrauss/Compressed Sensing: coming soon!

Other Dimensionality Reduction Algorithms

By no means an exhaustive list:

- Nonlinear/Robust PCA: coming soon!
- Johnson-Lindenstrauss/Compressed Sensing: coming soon!
- t-SNE: embeds to ensure close-together points are mapped close together with high probability.

Other Dimensionality Reduction Algorithms

By no means an exhaustive list:

- Nonlinear/Robust PCA: coming soon!
- Johnson-Lindenstrauss/Compressed Sensing: coming soon!
- t-SNE: embeds to ensure close-together points are mapped close together with high probability.
- TDA: embeds to make sure point clouds maintain their form.

Other Dimensionality Reduction Algorithms

By no means an exhaustive list:

- Nonlinear/Robust PCA: coming soon!
- Johnson-Lindenstrauss/Compressed Sensing: coming soon!
- t-SNE: embeds to ensure close-together points are mapped close together with high probability.
- TDA: embeds to make sure point clouds maintain their form.
- Autoencoders: embeds using neural networks to maximize information usable by the network.