Lecture 6. Operators in Euclidean space and singular value decomposition (SVD)

Alex Avdiushenko

Neapolis University Paphos

June 14, 2023

Plan for the lecture

- Recall Euclidean space: $\mathbb{R}^n$, linear operator: $T : \mathbb{R}^n \to \mathbb{R}^m$

Plan for the lecture

- Recall Euclidean space: $\mathbb{R}^n$, linear operator: $T: \mathbb{R}^n \to \mathbb{R}^m$
- Combine operators with inner product and space geometry:

Plan for the lecture

- Recall Euclidean space: $\mathbb{R}^n$, linear operator: $T : \mathbb{R}^n \to \mathbb{R}^m$
- Combine operators with inner product and space geometry:
  1. Orthogonal operators and motions ("rigid transformations") of space

Plan for the lecture

- Recall Euclidean space: $\mathbb{R}^n$, linear operator: $T : \mathbb{R}^n \to \mathbb{R}^m$
- Combine operators with inner product and space geometry:
  1. Orthogonal operators and motions ("rigid transformations") of space
  2. Self-adjoint operators (algebraic approach), which is actually scaling the space along the orthogonal directions

Plan for the lecture

- Recall Euclidean space: $\mathbb{R}^n$, linear operator: $T : \mathbb{R}^n \to \mathbb{R}^m$
- Combine operators with inner product and space geometry:
  1. Orthogonal operators and motions (“rigid transformations”) of space
  2. Self-adjoint operators (algebraic approach), which is actually scaling the space along the orthogonal directions
- Singular values decomposition and its applications

0. Orthogonal Projectors

An **orthogonal projector** is a linear operator P in a Euclidean space $\mathbb{R}^n$ that satisfies the following properties:

0. Orthogonal Projectors

An **orthogonal projector** is a linear operator P in a Euclidean space $\mathbb{R}^n$ that satisfies the following properties:

- $P^2 = P$, i.e., P is idempotent

0. Orthogonal Projectors

An **orthogonal projector** is a linear operator P in a Euclidean space $\mathbb{R}^n$ that satisfies the following properties:

- $P^2 = P$, i.e., P is idempotent
- $P^T = P$, i.e., P is self-adjoint

0. Orthogonal Projectors

An **orthogonal projector** is a linear operator P in a Euclidean space $\mathbb{R}^n$ that satisfies the following properties:

- $P^2 = P$, i.e., P is idempotent
- $P^T = P$, i.e., P is self-adjoint

Geometrically, the action of an orthogonal projector P on a vector x in the space results in a new vector $y = Px$ that lies on a subspace $U \subseteq \mathbb{R}^n$. The subspace U is the image of P, and the difference vector $x - y$ is orthogonal to U.
If U has an orthonormal basis $\{u_1, u_2, \ldots, u_k\}$, then the matrix representation of P can be given as:

$$P = \sum_{i=1}^{k} u_i u_i^T$$

Notation

$\mathrm{pr}_U \mathbb{R}^n$ — orthogonal projector on U along $U^\perp$

Notation

$\mathrm{pr}_U \mathbb{R}^n$ — orthogonal projector on U along $U^\perp$

- $U + U^\perp = \mathbb{R}^n$
- $U \cap U^\perp = 0$

Notation

$\text{pr}_U \mathbb{R}^n$ — orthogonal projector on U along $U^\perp$

- $U + U^\perp = \mathbb{R}^n$
- $U \cap U^\perp = 0$
- $\langle x, y \rangle = x^T y$
- $U = \text{span}\langle u_1, u_2, \dots, u_k \rangle$
- $A = (u_1 \mid \cdots \mid u_k)$
- $U^\perp = \{y \in \mathbb{R}^n \mid A^T y = 0\}$

Then $\text{pr}_U v = A(A^T A)^{-1} A^T v$

Least Squares Method (LSM)

Suppose we have system of linear equations $Ax = b \in \mathbb{R}^m,\ x \in \mathbb{R}^n$.

Least Squares Method (LSM)

Suppose we have system of linear equations $Ax = b \in \mathbb{R}^m, \ x \in \mathbb{R}^n$. As we've already known it may have solution and then $|Ax - b| = 0$ or may not.

Least Squares Method (LSM)

Suppose we have system of linear equations $Ax = b \in \mathbb{R}^m,\ x \in \mathbb{R}^n$.
As we've already known it may have solution and then $|Ax - b| = 0$ or may not.
The **Least Squares Method** is a widely used technique for fitting a model to data by minimizing the sum of the squares of the residuals (i.e., the differences between observed values and predicted values):

$$|Ax - b| \to \min$$

Least Squares Method (LSM)

Suppose we have system of linear equations $Ax = b \in \mathbb{R}^m$, $x \in \mathbb{R}^n$.
As we've already known it may have solution and then $|Ax - b| = 0$ or may not.
The **Least Squares Method** is a widely used technique for fitting a model to data by minimizing the sum of the squares of the residuals (i.e., the differences between observed values and predicted values):

$$|Ax - b| \to \min$$

Question

Ok, but where are the actual **squares**?

Least Squares Method (LSM)

Suppose we have system of linear equations $Ax = b \in \mathbb{R}^m, \ x \in \mathbb{R}^n$.
As we've already known it may have solution and then $|Ax - b| = 0$ or may not.
The **Least Squares Method** is a widely used technique for fitting a model to data by minimizing the sum of the squares of the residuals (i.e., the differences between observed values and predicted values):

$$|Ax - b| \to \min$$

Question

Ok, but where are the actual **squares**?

$$|x| = \sqrt{\langle x^T, x \rangle} = \sqrt{x_1^2 + \cdots + x_n^2}$$

The Least Squares solution can be found using various techniques, such as:

The Least Squares solution can be found using various techniques, such as:

- Normal equations — below

The Least Squares solution can be found using various techniques, such as:

- Normal equations — below
- Gradient descent (this is Machine Learning)

The Least Squares solution can be found using various techniques, such as:

- Normal equations — below
- Gradient descent (this is Machine Learning)
- Singular Value Decomposition (SVD) — at the end of the lecture

The Least Squares solution can be found using various techniques, such as:

- Normal equations — below
- Gradient descent (this is Machine Learning)
- Singular Value Decomposition (SVD) — at the end of the lecture

- $A = (A_1 \mid \cdots \mid A_n)$
- $x_1 A_1 + \cdots + x_n A_n = \mathrm{pr}_U\, b$ (geometric intuition and if columns of A are linearly independent)

The Least Squares solution can be found using various techniques, such as:

- Normal equations — below
- Gradient descent (this is Machine Learning)
- Singular Value Decomposition (SVD) — at the end of the lecture

- $A = (A_1 \mid \cdots \mid A_n)$
- $x_1 A_1 + \cdots + x_n A_n = \mathrm{pr}_U\, b$ (geometric intuition and if columns of A are linearly independent)
- $\mathrm{pr}_U\, b = \mathrm{pr}_U\, Ax = A\left(A^T A\right)^{-1} A^T b$
- $x = \left(A^T A\right)^{-1} A^T b$

LSM in machine learning world

LSM in machine learning world

Given a set of data points $\{(x_i, y_i)\}_{i=1}^m$ and a model function $f(x, \boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$, the goal is to find the optimal values of $\boldsymbol{\theta}$ that minimize the objective function:

$$Q(\boldsymbol{\theta}) = \sum_{i=1}^{m} (y_i - f(x_i, \boldsymbol{\theta}))^2$$

Given a set of data points $\{(x_i, y_i)\}_{i=1}^{m}$ and a model function $f(x, \boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$, the goal is to find the optimal values of $\boldsymbol{\theta}$ that minimize the objective function:

$$Q(\boldsymbol{\theta}) = \sum_{i=1}^{m} (y_i - f(x_i, \boldsymbol{\theta}))^2$$

In the context of linear regression, the model function is a linear combination of the parameters:

$$f(x, \boldsymbol{\theta}) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

Regularization

Regularization

It is a technique used to prevent overfitting in machine learning models by adding a penalty term to the objective function. This helps to reduce the complexity of the model and improve generalization.

Regularization

It is a technique used to prevent overfitting in machine learning models by adding a penalty term to the objective function. This helps to reduce the complexity of the model and improve generalization.
The objective function with regularization can be written as:

$$Q(\boldsymbol{\theta}) = \sum_{i=1}^{m} (y_i - f(x_i, \boldsymbol{\theta}))^2 + \lambda R(\boldsymbol{\theta})$$

where λ is the regularization parameter and $R(\boldsymbol{\theta})$ is the regularization term. Common regularization techniques include:

- **Lasso (L1) regularization**: $R(\boldsymbol{\theta}) = \sum_{j=1}^{n} |\theta_j|$

Regularization

It is a technique used to prevent overfitting in machine learning models by adding a penalty term to the objective function. This helps to reduce the complexity of the model and improve generalization.
The objective function with regularization can be written as:

$$Q(\boldsymbol{\theta}) = \sum_{i=1}^{m} (y_i - f(x_i, \boldsymbol{\theta}))^2 + \lambda R(\boldsymbol{\theta})$$

where λ is the regularization parameter and $R(\boldsymbol{\theta})$ is the regularization term. Common regularization techniques include:

- **Lasso (L1) regularization**: $R(\boldsymbol{\theta}) = \sum_{j=1}^{n} |\theta_j|$
- **Ridge (L2) regularization**: $R(\boldsymbol{\theta}) = \sum_{j=1}^{n} \theta_j^2$

Regularization

It is a technique used to prevent overfitting in machine learning models by adding a penalty term to the objective function. This helps to reduce the complexity of the model and improve generalization.
The objective function with regularization can be written as:

$$Q(\boldsymbol{\theta}) = \sum_{i=1}^{m} (y_i - f(x_i, \boldsymbol{\theta}))^2 + \lambda R(\boldsymbol{\theta})$$

where λ is the regularization parameter and $R(\boldsymbol{\theta})$ is the regularization term. Common regularization techniques include:

- **Lasso (L1) regularization**: $R(\boldsymbol{\theta}) = \sum_{j=1}^{n} |\theta_j|$
- **Ridge (L2) regularization**: $R(\boldsymbol{\theta}) = \sum_{j=1}^{n} \theta_j^2$
- **Elastic Net**: A combination of L1 and L2 regularization

Regularization

It is a technique used to prevent overfitting in machine learning models by adding a penalty term to the objective function. This helps to reduce the complexity of the model and improve generalization.
The objective function with regularization can be written as:

$$Q(\boldsymbol{\theta}) = \sum_{i=1}^{m} (y_i - f(x_i, \boldsymbol{\theta}))^2 + \lambda R(\boldsymbol{\theta})$$

where λ is the regularization parameter and $R(\boldsymbol{\theta})$ is the regularization term. Common regularization techniques include:

- **Lasso (L1) regularization**: $R(\boldsymbol{\theta}) = \sum_{j=1}^{n} |\theta_j|$
- **Ridge (L2) regularization**: $R(\boldsymbol{\theta}) = \sum_{j=1}^{n} \theta_j^2$
- **Elastic Net**: A combination of L1 and L2 regularization

Regularization techniques can be applied to other machine learning models as well, such as logistic regression, neural networks, and support vector machines.

For Ridge (L2) regularization in matrix notation, we have:

$$x = \left(A^T A + \lambda E\right)^{-1} A^T b$$

It is easy to prove that all these three properties are equivalent. The first condition is algebraic and the other two are geometric.

It is easy to prove that all these three properties are equivalent. The first condition is algebraic and the other two are geometric.

- An operator U is a linear transformation that preserves the inner product, i.e., $\langle U\boldsymbol{x}, U\boldsymbol{y}\rangle = \langle \boldsymbol{x}, \boldsymbol{y}\rangle$ for all vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$

It is easy to prove that all these three properties are equivalent. The first condition is algebraic and the other two are geometric.

- An operator U is a linear transformation that preserves the inner product, i.e., $\langle U\boldsymbol{x}, U\boldsymbol{y}\rangle = \langle \boldsymbol{x}, \boldsymbol{y}\rangle$ for all vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$
- Linear operator preserves lengths and angles between vectors: $|U\boldsymbol{x}| = |\boldsymbol{x}|$, $\angle U\boldsymbol{x}, U\boldsymbol{y} = \angle \boldsymbol{x}, \boldsymbol{y}$

1. Orthogonal Operators and Motions

It is easy to prove that all these three properties are equivalent. The first condition is algebraic and the other two are geometric.

- An operator U is a linear transformation that preserves the inner product, i.e., $\langle U\boldsymbol{x}, U\boldsymbol{y}\rangle = \langle \boldsymbol{x}, \boldsymbol{y}\rangle$ for all vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$
- Linear operator preserves lengths and angles between vectors: $|U\boldsymbol{x}| = |\boldsymbol{x}|$, $\angle U\boldsymbol{x}, U\boldsymbol{y} = \angle \boldsymbol{x}, \boldsymbol{y}$
- Linear operator preserves lengths of vectors: $|U\boldsymbol{x}| = |\boldsymbol{x}|$

Example

For $\mathbb{R}^n$ and $\langle x, y\rangle = x^T y$ we have:

- $x \mapsto Ax$
- $xA^T Ay = (Ax)^T Ay = x^T y \Rightarrow \; A^T A = E$

Example

For $\mathbb{R}^n$ and $\langle x, y\rangle = x^T y$ we have:

- $x \mapsto Ax$
- $xA^T Ay = (Ax)^T Ay = x^T y \Rightarrow \; A^T A = E$

Orthogonal matrix

For $A \in \mathbb{R}^n$ the following statements are equivalent:

- $A^T A = E$ (orthonormal columns)
- $AA^T = E$ (orthonormal rows)
- $A^T = A^{-1}$

For any vector space with **orthonormal basis** it is general case of the motions' matrix.

Eigenvalues and Eigenvectors of Motions

Let A be an orthogonal operator that represents a motion in a Euclidean space. Consider an eigenvector v and its corresponding eigenvalue λ such that:

$$Av = \lambda v$$

Eigenvalues and Eigenvectors of Motions

Let A be an orthogonal operator that represents a motion in a Euclidean space. Consider an eigenvector v and its corresponding eigenvalue λ such that:

$$Av = \lambda v$$

Since A is orthogonal, its columns form an orthonormal basis, and we have:

$$A^T A = AA^T = E$$

Eigenvalues and Eigenvectors of Motions

Let A be an orthogonal operator that represents a motion in a Euclidean space. Consider an eigenvector v and its corresponding eigenvalue λ such that:

$$Av = \lambda v$$

Since A is orthogonal, its columns form an orthonormal basis, and we have:

$$A^T A = AA^T = E$$

Multiplying the eigenvector equation by A^T from the left:

$$A^T Av = A^T(\lambda v)$$

Eigenvalues and Eigenvectors of Motions

Let A be an orthogonal operator that represents a motion in a Euclidean space. Consider an eigenvector v and its corresponding eigenvalue λ such that:

$$Av = \lambda v$$

Since A is orthogonal, its columns form an orthonormal basis, and we have:

$$A^T A = AA^T = E$$

Multiplying the eigenvector equation by A^T from the left:

$$A^T Av = A^T(\lambda v)$$

Using the property $A^T A = E$, we obtain:

$$v = \lambda A^T v$$

This shows that v is also an eigenvector of A with the same eigenvalue λ. Furthermore, since A is orthogonal, the eigenvalues satisfy the condition:

$$|\lambda|^2 = 1$$

This shows that v is also an eigenvector of A with the same eigenvalue λ. Furthermore, since A is orthogonal, the eigenvalues satisfy the condition:

$$|\lambda|^2 = 1$$

Therefore, the eigenvalues of a motion are either 1 or -1.

This shows that v is also an eigenvector of A with the same eigenvalue λ. Furthermore, since A is orthogonal, the eigenvalues satisfy the condition:

$$|\lambda|^2 = 1$$

Therefore, the eigenvalues of a motion are either 1 or -1.
The eigenvectors corresponding to the eigenvalue 1 represent points that remain fixed under the motion, while eigenvectors corresponding to the eigenvalue -1 represent points that are reflected through the origin.

Wonderful Remark

$$\det A = \pm 1$$

Examples of Motions in $\mathbb{R}^2$

- **Rotation around the origin:**

$$A_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

Rotates the plane counterclockwise by an angle θ

Examples of Motions in $\mathbb{R}^2$

- **Rotation around the origin:**

$$A_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

  Rotates the plane counterclockwise by an angle θ

- **Reflection across the line $y = x$:**

$$A_{y=x} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

  Reflects points across the line $y = x$

Examples of Motions in $\mathbb{R}^2$

- **Rotation around the origin:**

$$A_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

  Rotates the plane counterclockwise by an angle θ

- **Reflection across the line $y = x$:**

$$A_{y=x} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

  Reflects points across the line $y = x$

- **Reflection across the line $y = x$:**

$$A = \begin{pmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{pmatrix}$$

  Reflects points across the line $y = x\tan\frac{\theta}{2}$

Examples of Motions in $\mathbb{R}^3$

- **Rotation around the x-axis:**

$$A_{\theta,x} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}$$

Rotates the space counterclockwise by an angle θ around the x-axis

Examples of Motions in $\mathbb{R}^3$

- **Rotation around the x-axis:**

$$A_{\theta,x} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}$$

  Rotates the space counterclockwise by an angle θ around the x-axis

- **Reflection across the xy-plane:**

$$A_{xy} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

  Reflects points across the xy-plane

General Form of Matrix for Motions in $\mathbb{R}^n$

A motion in $\mathbb{R}^n$ can be represented by an orthogonal matrix with the following structure:

$$A = \begin{pmatrix} \pm 1 & & & & & & & \\ & \ddots & & & & & & \\ & & \pm 1 & & & & & \\ & & & \cos\theta_1 & -\sin\theta_1 & & & \\ & & & \sin\theta_1 & \cos\theta_1 & & & \\ & & & & & \ddots & & \\ & & & & & & \cos\theta_k & -\sin\theta_k \\ & & & & & & \sin\theta_k & \cos\theta_k \end{pmatrix}$$

where $k \leq \frac{n}{2}$ and the 2×2 blocks with sin and cos terms represent rotations in the corresponding 2-dimensional subspaces.
The ±1 entries on the diagonal can represent reflections and identity transformations in the corresponding 1-dimensional subspaces.

2. Self-Adjoint Operators

Consider an operator $A\colon V \to V$ on an inner product space V. By definition adjoint operator A^* holds:

$$\langle Ax, y\rangle = \langle x, A^* y\rangle, \quad \forall x, y \in V$$

For orthonormal basis

$$x^T A^T y = (Ax)^T y = x^T A^* y \;\Rightarrow\; A^* = A^T$$

Definition

A is called *self-adjoint* (or *Hermitian*) if it satisfies the following condition:

$$\langle Ax, y\rangle = \langle x, Ay\rangle, \quad \forall x, y \in V$$

Properties of self-adjoint operators

Properties of self-adjoint operators

- All eigenvalues are real: $\operatorname{Spec}_{\mathbb{C}} A = \operatorname{Spec}_{\mathbb{R}} A$

Properties of self-adjoint operators

- All eigenvalues are real: $\operatorname{Spec}_{\mathbb{C}} A = \operatorname{Spec}_{\mathbb{R}} A$
- Eigenvectors corresponding to distinct eigenvalues are orthogonal: $Av = \lambda v,\ Au = \mu u \ \Rightarrow \ v \perp u$

Properties of self-adjoint operators

- All eigenvalues are real: $\mathrm{Spec}_{\mathbb{C}} A = \mathrm{Spec}_{\mathbb{R}} A$
- Eigenvectors corresponding to distinct eigenvalues are orthogonal: $Av = \lambda v,\ Au = \mu u \ \Rightarrow\ v \perp u$
- Self-adjoint operators can be diagonalized by an orthogonal matrix: $\exists e_1, \dots e_n$ — orthonormal basis: $A = \mathrm{diag}(\lambda_1, \dots, \lambda_n)$

Properties of self-adjoint operators

- All eigenvalues are real: $\operatorname{Spec}_{\mathbb{C}} A = \operatorname{Spec}_{\mathbb{R}} A$
- Eigenvectors corresponding to distinct eigenvalues are orthogonal: $Av = \lambda v,\ Au = \mu u \ \Rightarrow\ v \perp u$
- Self-adjoint operators can be diagonalized by an orthogonal matrix: $\exists e_1, \ldots e_n$ — orthonormal basis: $A = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$

Remark

You should use Gram-Schmidt orthogonalization process to get orthonormal parts of basis, corresponding certain eigenvalue

Singular Value Decomposition (SVD)

Suppose $\phi : V \to U$ on the Euclidean spaces U, V. You need to find orthonormal bases to simplify the matrix A_ϕ.

The Singular Value Decomposition (SVD) is a factorization of a real or complex matrix $A \in \mathbb{R}^{m \times n}$. The SVD of A is given by:

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

Where:

Singular Value Decomposition (SVD)

Suppose $\phi : V \to U$ on the Euclidean spaces U, V. You need to find orthonormal bases to simplify the matrix A_ϕ.

The Singular Value Decomposition (SVD) is a factorization of a real or complex matrix $A \in \mathbb{R}^{m\times n}$. The SVD of A is given by:

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

Where:

- $U \in \mathbb{R}^{m\times m}$ is an orthogonal matrix containing the left singular vectors of A

Singular Value Decomposition (SVD)

Suppose $\phi : V \to U$ on the Euclidean spaces U, V. You need to find orthonormal bases to simplify the matrix A_ϕ.

The Singular Value Decomposition (SVD) is a factorization of a real or complex matrix $A \in \mathbb{R}^{m\times n}$. The SVD of A is given by:

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

Where:

- $U \in \mathbb{R}^{m\times m}$ is an orthogonal matrix containing the left singular vectors of A
- $\Sigma \in \mathbb{R}^{m\times n}$ is a diagonal matrix with non-negative diagonal entries, called singular values, in non-increasing order $(\sigma_1 \geq \sigma_2 \geq \cdots \geq 0)$

Singular Value Decomposition (SVD)

Suppose $\phi : V \to U$ on the Euclidean spaces U, V. You need to find orthonormal bases to simplify the matrix A_ϕ.

The Singular Value Decomposition (SVD) is a factorization of a real or complex matrix $A \in \mathbb{R}^{m \times n}$. The SVD of A is given by:

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

Where:

- $U \in \mathbb{R}^{m \times m}$ is an orthogonal matrix containing the left singular vectors of A
- $\Sigma \in \mathbb{R}^{m \times n}$ is a diagonal matrix with non-negative diagonal entries, called singular values, in non-increasing order $(\sigma_1 \geq \sigma_2 \geq \cdots \geq 0)$
- $V \in \mathbb{R}^{n \times n}$ is an orthogonal matrix containing the right singular vectors of A

Explanations and applications

$$A = \begin{bmatrix} u_1 \,| & \dots & |\, u_n \end{bmatrix} \begin{bmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_r & & \\ & & & 0 & \\ & & & & \ddots \end{bmatrix} \begin{bmatrix} v_1^T \\ \vdots \\ v_n^T \end{bmatrix}$$

$$A = \sigma_1 u_1 v_1^T + \dots + \sigma_r u_r v_r^T$$

Explanations and applications

$$A = \left[u_1 \mid \quad \dots \quad \mid u_n \right] \begin{bmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_r & & \\ & & & 0 & \\ & & & & \ddots \end{bmatrix} \begin{bmatrix} v_1^T \\ \vdots \\ v_n^T \end{bmatrix}$$

$$A = \sigma_1 u_1 v_1^T + \dots + \sigma_r u_r v_r^T$$

- Data compression: keep only first k terms $(mn \mapsto k(m+n+1))$

Explanations and applications

$$A = \begin{bmatrix} u_1 \,|\, & \dots & \,|\, u_n \end{bmatrix} \begin{bmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_r & & \\ & & & 0 & \\ & & & & \ddots \end{bmatrix} \begin{bmatrix} v_1^T \\ \vdots \\ v_n^T \end{bmatrix}$$

$$A = \sigma_1 u_1 v_1^T + \dots + \sigma_r u_r v_r^T$$

- Data compression: keep only first k terms ($mn \mapsto k(m+n+1)$)
- Removing the background of a video stream from static camera

  Represent the frames as vectors and make up the matrix A from them all. Make SVD and zero first k terms to remove background.

Object-feature matrix and SVD

$$U^T \left[A_1 \mid \quad \dots \quad \mid A_n \right] = \begin{bmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_r & & \\ & & & 0 & \\ & & & & \ddots \end{bmatrix} V^T$$

Actually in V^T you've got new features with their importances.

Types of SVD and Storage Considerations

Depending on the application, there are different types of SVD and storage options:

- **Full SVD:**
$$A = U\Sigma V^T$$
  - Computes all singular values and vectors
  - Storage: $O(m^2 + mn + n^2)$

Types of SVD and Storage Considerations

Depending on the application, there are different types of SVD and storage options:

- **Full SVD:**

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

  - Computes all singular values and vectors
  - Storage: $O(m^2 + mn + n^2)$
- **Thin (Reduced) SVD:**

$$A = U_r \Sigma_r V_r^T$$

  - Computes only the first $r = \min(m, n)$ singular values and vectors
  - Storage: $O(mr + nr)$

- **Truncated SVD:**
$$A_k \approx U_k \Sigma_k V_k^T$$
    - Computes only the first k largest singular values and vectors ($k \ll \min(m,n)$)
    - Storage: $O(mk + nk)$
    - Often used for dimensionality reduction and approximation

Storage requirements vary depending on the type of SVD and the number of singular values and vectors needed for the application.

How to Find SVD?

To compute the SVD $A = U\Sigma V^T$ of a given matrix $A \in \mathbb{R}^{m \times n}$:

How to Find SVD?

To compute the SVD $A = U\Sigma V^T$ of a given matrix $A \in \mathbb{R}^{m \times n}$:

1. Calculate the eigenvectors and eigenvalues of AA^T.
   - Eigenvectors of AA^T: columns of U

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

How to Find SVD?

To compute the SVD $A = U\Sigma V^T$ of a given matrix $A \in \mathbb{R}^{m \times n}$:

1. Calculate the eigenvectors and eigenvalues of AA^T.
   - Eigenvectors of AA^T: columns of U

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

2. Sort the eigenvalues in decreasing order, and rearrange the eigenvectors accordingly.
   - $\chi_{AA^T} = 0 \to \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r \geq 0$

How to Find SVD?

To compute the SVD $A = U\Sigma V^T$ of a given matrix $A \in \mathbb{R}^{m \times n}$:

1. Calculate the eigenvectors and eigenvalues of AA^T.
   - Eigenvectors of AA^T: columns of U

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

2. Sort the eigenvalues in decreasing order, and rearrange the eigenvectors accordingly.
   - $\chi_{AA^T} = 0 \to \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r \geq 0$
3. Form the diagonal matrix Σ using the square roots of the sorted eigenvalues.

$$\sigma_i = \sqrt{\lambda_i}$$

How to Find SVD?

To compute the SVD $A = U\Sigma V^T$ of a given matrix $A \in \mathbb{R}^{m\times n}$:

1. Calculate the eigenvectors and eigenvalues of AA^T.
   - Eigenvectors of AA^T: columns of U

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

2. Sort the eigenvalues in decreasing order, and rearrange the eigenvectors accordingly.
   - $\chi_{AA^T} = 0 \to \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r \geq 0$
3. Form the diagonal matrix Σ using the square roots of the sorted eigenvalues.

$$\sigma_i = \sqrt{\lambda_i}$$

4. $V:\ A^T = V\Sigma^T U^T \ \Rightarrow\ A^T U = V\Sigma^T \ \Rightarrow\ v_i = \frac{A^T u_i}{\sigma_i}$ for $i = 1, \ldots, r$

How to Find SVD?

To compute the SVD $A = U\Sigma V^T$ of a given matrix $A \in \mathbb{R}^{m\times n}$:

1. Calculate the eigenvectors and eigenvalues of AA^T.
   - Eigenvectors of AA^T: columns of U

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

2. Sort the eigenvalues in decreasing order, and rearrange the eigenvectors accordingly.
   - $\chi_{AA^T} = 0 \to \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r \geq 0$
3. Form the diagonal matrix Σ using the square roots of the sorted eigenvalues.
$$\sigma_i = \sqrt{\lambda_i}$$
4. $V: \; A^T = V\Sigma^T U^T \;\Rightarrow\; A^T U = V\Sigma^T \;\Rightarrow\; v_i = \frac{A^T u_i}{\sigma_i}$ for $i = 1, \ldots, r$
5. $\langle v_1, \ldots, v_r\rangle^\perp = \{y \mid Ay = 0\} \mapsto$ FSS, G.-Sh., norm $\mapsto y_1, \ldots, y_{n-r}$

Example

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Example

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \rightarrow AA^T = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

Example

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \rightarrow AA^T = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\chi_{AA^T}(t) = t^2 - 4t + 3 \mapsto \lambda_1 = 3, \lambda_2 = 1 \mapsto \sigma_1 = \sqrt{3}, \sigma_2 = 1$$

Example

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \rightarrow AA^T = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\chi_{AA^T}(t) = t^2 - 4t + 3 \mapsto \lambda_1 = 3, \lambda_2 = 1 \mapsto \sigma_1 = \sqrt{3}, \sigma_2 = 1$$

$$\lambda_1 : (AA^T - 3E)x = 0 \mapsto \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} x = 0 \mapsto \begin{bmatrix} 1 \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

Example

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \rightarrow AA^T = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\chi_{AA^T}(t) = t^2 - 4t + 3 \mapsto \lambda_1 = 3, \lambda_2 = 1 \mapsto \sigma_1 = \sqrt{3}, \sigma_2 = 1$$

$$\lambda_1 : (AA^T - 3E)x = 0 \mapsto \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} x = 0 \mapsto \begin{bmatrix} 1 \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

$$\lambda_2 : (AA^T - E)x = 0 \mapsto \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} x = 0 \mapsto \begin{bmatrix} -1 \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

Example

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \rightarrow AA^T = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\chi_{AA^T}(t) = t^2 - 4t + 3 \mapsto \lambda_1 = 3, \lambda_2 = 1 \mapsto \sigma_1 = \sqrt{3}, \sigma_2 = 1$$

$$\lambda_1 : (AA^T - 3E)x = 0 \mapsto \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} x = 0 \mapsto \begin{bmatrix} 1 \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

$$\lambda_2 : (AA^T - E)x = 0 \mapsto \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} x = 0 \mapsto \begin{bmatrix} -1 \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

$$A = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \sqrt{3} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} V$$

$$v_1 = \frac{A^T u_1}{\sigma_1} = \frac{1}{\sqrt{6}} \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}, v_2 = \frac{A^T u_2}{\sigma_2} = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}$$

$$v_1 = \frac{A^T u_1}{\sigma_1} = \frac{1}{\sqrt{6}} \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}, v_2 = \frac{A^T u_2}{\sigma_2} = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}$$

$$y_1 = [1, -1, -1]^T \mapsto \frac{1}{\sqrt{3}}[1, -1, -1]^T$$

$$v_1 = \frac{A^T u_1}{\sigma_1} = \frac{1}{\sqrt{6}} \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}, v_2 = \frac{A^T u_2}{\sigma_2} = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}$$

$$y_1 = [1, -1, -1]^T \mapsto \frac{1}{\sqrt{3}}[1, -1, -1]^T$$

$$A = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \sqrt{3} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \end{bmatrix}$$

Remark

For “tall” matrices better to start from the V and $A^T A$ matrices

Low-Rank Matrix Approximation

The SVD allows us to approximate a given matrix $A \in \mathbb{R}^{m \times n}$ with a low-rank matrix.

Low-Rank Matrix Approximation

The SVD allows us to approximate a given matrix $A \in \mathbb{R}^{m \times n}$ with a low-rank matrix.

- Given the SVD of $A = U\Sigma V^T$, we can approximate A by a rank-k matrix, A_k:

$$A_k = U_k \Sigma_k V_k^T$$

where U_k, Σ_k, and V_k are truncated versions of U, Σ, and V

Low-Rank Matrix Approximation

The SVD allows us to approximate a given matrix $A \in \mathbb{R}^{m \times n}$ with a low-rank matrix.

- Given the SVD of $A = U\Sigma V^T$, we can approximate A by a rank-k matrix, A_k:

$$A_k = U_k \Sigma_k V_k^T$$

  where U_k, Σ_k, and V_k are truncated versions of U, Σ, and V
- $U_k \in \mathbb{R}^{m \times k}$ consists of the first k columns of U

Low-Rank Matrix Approximation

The SVD allows us to approximate a given matrix $A \in \mathbb{R}^{m \times n}$ with a low-rank matrix.

- Given the SVD of $A = U\Sigma V^T$, we can approximate A by a rank-k matrix, A_k:

$$A_k = U_k \Sigma_k V_k^T$$

  where U_k, Σ_k, and V_k are truncated versions of U, Σ, and V
- $U_k \in \mathbb{R}^{m \times k}$ consists of the first k columns of U
- $\Sigma_k \in \mathbb{R}^{k \times k}$ consists of the k largest singular values in Σ

Low-Rank Matrix Approximation

The SVD allows us to approximate a given matrix $A \in \mathbb{R}^{m \times n}$ with a low-rank matrix.

- Given the SVD of $A = U\Sigma V^T$, we can approximate A by a rank-k matrix, A_k:

$$A_k = U_k \Sigma_k V_k^T$$

  where U_k, Σ_k, and V_k are truncated versions of U, Σ, and V
- $U_k \in \mathbb{R}^{m \times k}$ consists of the first k columns of U
- $\Sigma_k \in \mathbb{R}^{k \times k}$ consists of the k largest singular values in Σ
- $V_k \in \mathbb{R}^{n \times k}$ consists of the first k columns of V

Low-Rank Matrix Approximation

The SVD allows us to approximate a given matrix $A \in \mathbb{R}^{m \times n}$ with a low-rank matrix.

- Given the SVD of $A = U\Sigma V^T$, we can approximate A by a rank-k matrix, A_k:
$$A_k = U_k \Sigma_k V_k^T$$
where U_k, Σ_k, and V_k are truncated versions of U, Σ, and V
- $U_k \in \mathbb{R}^{m \times k}$ consists of the first k columns of U
- $\Sigma_k \in \mathbb{R}^{k \times k}$ consists of the k largest singular values in Σ
- $V_k \in \mathbb{R}^{n \times k}$ consists of the first k columns of V
- The approximation error can be measured using the Frobenius norm:
$$\|A - A_k\|_F = \sqrt{\sum_{i=k+1}^{\min(m,n)} \sigma_i^2} = \sqrt{\operatorname{tr}(B^T B)}$$
where σ_i are the singular values of A