Today's Agenda

- Singular Value Decomposition (SVD)
- Optimization on Least-squares (not covered in Final)

SVD

Derivation

- The singular values of A are the nonnegative square roots of the eigenvalues of $A^T A$.
- Every square Hermitian matrix is unitarily similar to a diagonal matrix, so

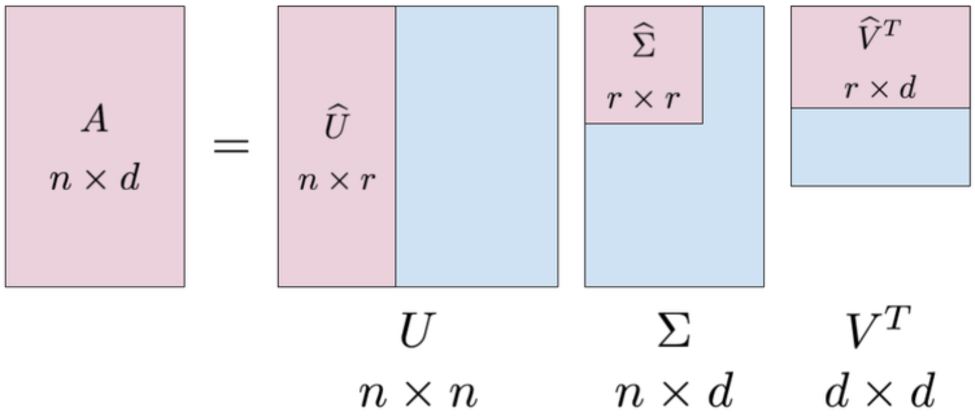
$$A^T A = QDQ^{-1}$$

- The diagonal elements of D are nonnegative

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r > 0 = \lambda_{r+1} = \cdots = \lambda_n$$

- Rank of the matrix A is r (numerical rank by SVD)

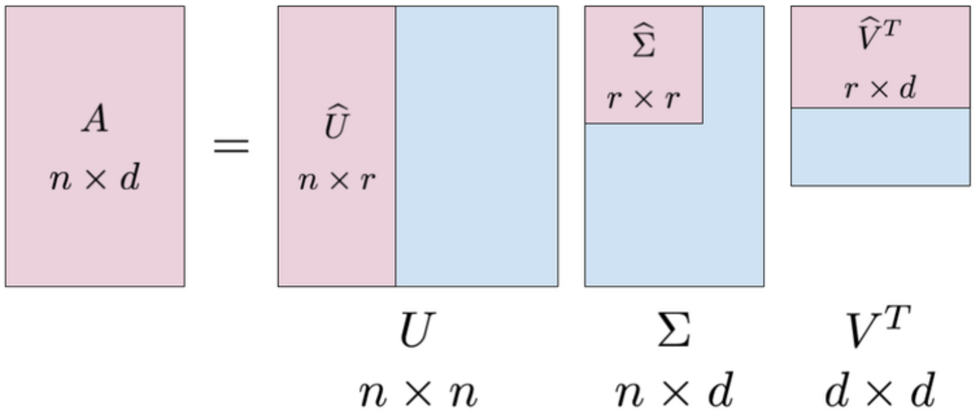
SVD definition



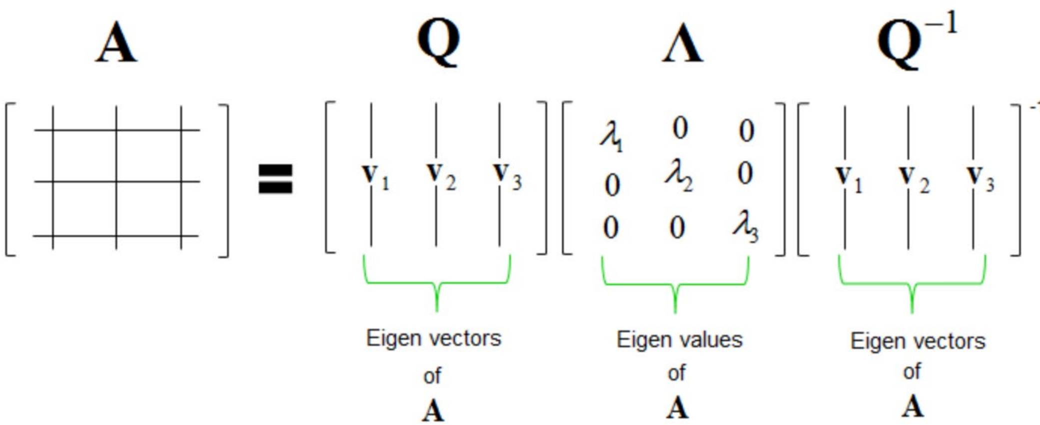
Unitary matrix: U, V
Diagonal: Σ

Economic SVD: [U,S,V]=svd(A, ‘econ’);

SVD v.s. eigen-decomp.



Any matrix has SVD with unitary matrices U and V.



Only square diagonalizable matrices have such eigen-decomp with an invertible matrix Q.

Eigenvalues v.s. singular values

- Identical for a square and symmetric matrix.
- It is often useful to identify eigenvalues to understand algorithm selection and convergence behavior
  - Spectral radius is defined to the leading eigenvalue.
- However, eigenvalues are often much more sensitive in terms of stability.

- Singular values, being more stable in general, are often more useful in understanding matrix behavior
  - Condition number is defined as the ratio of largest/smallest singular values: $\kappa(A) = \frac{\sigma_{\max}}{\sigma_{\min}}$

SVD and least-squares

- Linear system Ax = b
  - Consistent: there exists a solution
    i. Unique solution (square A)
    ii. Infinitely many (fat A)
  - Inconsistent: find x to minimize $||Ax - b||_2$
    iii. Square or tall A
- Least squares problem

$$\min_x ||Ax - b||_2^2$$

- The solution is given by $x = (A^T A)^{-1} A^T b$.
- This formula only works for "tall" A.

Under-determined system

- If the matrix A is "fat", $Ax = b$ is called under-determined linear system.
- There are infinitely many solutions to $Ax = b$.

- Least norm solution:

$$\min ||x||_2^2 \ \ s.t.\ Ax = b.$$

- Closed-form formula: $x = A^T(AA^T)^{-1}b$.

Pseudo-inverse

Penrose Properties of the Pseudo-Inverse

The pseudo-inverse A^+ for the matrix A has these four properties:

$$A = AA^+A \qquad A^+ = A^+AA^+$$
$$AA^+ = (AA^+)^T \qquad A^+A = (A^+A)^T$$

- Suppose SVD of $A = U\Sigma V^T$
- Pseudo-inverse: $A^+ = V\Sigma^+ U^T$

Optimization

Conjugate gradient

Gradient descent

- As a current point x_k, the function $\Phi(x)$ decreases most rapidly in the direction of negative gradient.
- Gradient descent algorithm

$$\begin{cases} \mathbf{p}_k & = -\nabla \phi(\mathbf{x}_{k-1}) \\ \mathbf{x}_k & = \mathbf{x}_{k-1} + \alpha_k \mathbf{p}_k , \end{cases}$$

  where α_k is a step size that can be fixed or updated iteratively.
- Finding a suitable stepsize is called linear search.

Quadratic programming

- Consider $\Phi(x) = \frac{1}{2}x^T A x - x^T b$ for a symmetric positive definite matrix A.

- $\min \Phi(x)$ is equivalent to solving $Ax = b$.

- The gradient of Φ is $\nabla\Phi(\mathrm{x}) = Ax - b$.

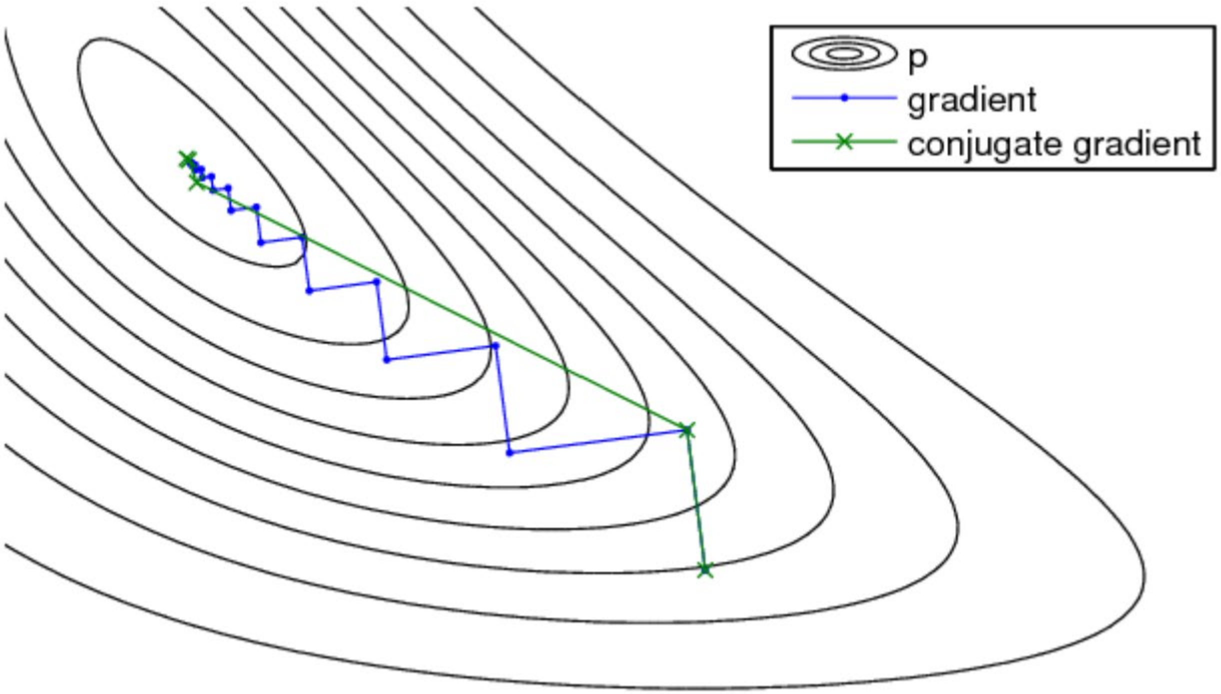
Steepest descent

- Recall $\begin{cases} \mathbf{p}_k &= -\nabla\phi(\mathbf{x}_{k-1}) \\ \mathbf{x}_k &= \mathbf{x}_{k-1} + \alpha_k \mathbf{p}_k, \end{cases}$
- $\nabla\Phi = Ax - b \;\Rightarrow\; \mathbf{p}_k = -\nabla\phi(\mathbf{x}_{k-1}) = \mathbf{b} - A\mathbf{x}_{k-1}.$
- Denote the residual $r_{k-1} := b - Ax_{k-1}$
- The optimal stepsize can be obtained exactly

$$\alpha_k = \frac{\mathbf{r}_{k-1}^T \mathbf{r}_{k-1}}{\mathbf{r}_{k-1}^T A \mathbf{r}_{k-1}}.$$

- This is called steepest descent (SD).

SD’s Drawback



Conjugate gradient

- The problem of SD is that $p_k = r_{k-1}$.
- Better to choose the descent directions as orthogonal as possible.
- Define A-conjugate: $u^T A v = 0$.
- Choose p_k be the closest vector to r_{k-1} that is A-conjugate to $p_1, p_2, \cdots, p_{k-1}$.

CG algorithm

- We can assume $p_k = r_{k-1} + \beta_k p_{k-1}$
- By A-conjugate, we can determine

$$\beta_k = -\frac{\mathbf{p}_{k-1}^T A \mathbf{r}_{k-1}}{\mathbf{p}_{k-1}^T A \mathbf{p}_{k-1}}.$$

- Residual recursion: $\mathbf{r}_{k-1} = \mathbf{r}_{k-2} - \alpha_k A \mathbf{p}_{k-1}$
- A simpler update $\beta_k = \mathbf{r}_{k-1}^T \mathbf{r}_{k-1} / \mathbf{r}_{k-2}^T \mathbf{r}_{k-2}$

Putting together

Starting from an initial

$$
\begin{cases}
\beta_k = \mathbf{r}_{k-1}^T \mathbf{r}_{k-1} / \mathbf{r}_{k-2}^T \mathbf{r}_{k-2} \\
\mathbf{p}_k = \mathbf{r}_{k-1} + \beta_k \mathbf{p}_{k-1} \\
\alpha_k = \mathbf{r}_{k-1}^T \mathbf{r}_{k-1} / \mathbf{p}_k^T A \mathbf{p}_k \\
\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_k \mathbf{p}_k \\
\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_k A \mathbf{p}_k
\end{cases}
$$