Linear Algebra Approaches and Algorithms for Data Analysis

Jingyuan Sun
ChatGPT Claude

Dec, 2024

1 Linear Algebra Recap

1.1 Eigenvalues and Eigenvectors

Let A be an $n \times n$ matrix. The eigenvalues of A are defined as the roots of:

$$\det(A - \lambda I) = 0, \tag{1}$$

where λ is an eigenvalue of A.

For an eigenvalue λ, there exists a vector $\mathbf{x}$ such that:

$$A\mathbf{x} = \lambda\mathbf{x}. \tag{2}$$

The vector $\mathbf{x}$ is called the eigenvector of A corresponding to the eigenvalue λ.

1.2 Matrix Representation of Eigenvectors and Eigenvalues

Suppose A is an $n \times n$ matrix with eigenvectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ and eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$. Then:

$$A\mathbf{x}_i = \lambda_i \mathbf{x}_i \quad \text{for } i = 1, \ldots, n. \tag{3}$$

In matrix format:

$$A\mathbf{X} = \mathbf{X}\Lambda, \tag{4}$$

where:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \ldots & \mathbf{x}_n \end{bmatrix}, \quad \Lambda = \begin{bmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \lambda_n \end{bmatrix}. \tag{5}$$

1.3 Singular Value Decomposition (SVD)

The Singular Value Decomposition of a matrix $A \in \mathbb{R}^{m \times n}$ is given by:

$$A = U\Sigma V^T, \tag{6}$$

where:

- $U \in \mathbb{R}^{m \times m}$ is an orthogonal matrix ($U^T U = I$), containing the left singular vectors of A.
- $\Sigma \in \mathbb{R}^{m \times n}$ is a diagonal matrix containing the singular values $\sigma_1, \sigma_2, \ldots$, with $\sigma_1 \geq \sigma_2 \geq \ldots \geq 0$.
- $V \in \mathbb{R}^{n \times n}$ is an orthogonal matrix ($V^T V = I$), containing the right singular vectors of A.

The singular values σ_i are the square roots of the eigenvalues of $A^T A$:

$$\sigma_i = \sqrt{\lambda_i(A^T A)}, \quad i = 1, \ldots, \min(m, n). \tag{7}$$

1.4 Relationship Between SVD and PCA

- For a covariance matrix $C = A^T A$, the eigenvalues λ_i correspond to the squared singular values σ_i^2 of A.
- The eigenvectors of C are the right singular vectors of A.
- In PCA, the principal components are obtained from the top singular values and their corresponding singular vectors.

1.5 Properties of SVD

1. The rank of A is equal to the number of non-zero singular values.
2. The Frobenius norm of A is related to its singular values:

$$\|A\|_F = \sqrt{\sum_{i=1}^{r} \sigma_i^2}, \tag{8}$$

   where $r = \text{rank}(A)$.

3. The 2-norm of A is the largest singular value:

$$\|A\|_2 = \sigma_1. \tag{9}$$

1.6 Applications of SVD

- **Dimensionality Reduction:** Use top k singular values and corresponding singular vectors to approximate the matrix A:

$$A_k = U_k \Sigma_k V_k^T, \tag{10}$$

  where U_k and V_k contain the top k singular vectors, and Σ_k contains the top k singular values.

- **Data Compression:** Approximation A_k requires less storage than A while retaining most of the data's variance.
- **Noise Reduction:** Low-rank approximations remove small singular values, effectively filtering noise.

2 Principal Component Analysis

PCA is a linear transformation technique aiming to project data onto a lower-dimensional space that captures most of the variance. Let us start with the fundamental definitions and theorems.

2.1 Preliminaries

Consider a dataset $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ where each $\mathbf{x}_i \in \mathbb{R}^d$. We assume the data is mean-centered:

$$\frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i = \mathbf{0}.$$

If not, we can always shift the data by subtracting the sample mean.

Definition 1 (Covariance Matrix)**.** *Given mean-centered data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, the sample covariance matrix $\mathbf{C}$ is defined as*

$$\mathbf{C} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top.$$

This is a $d \times d$ symmetric and positive semi-definite matrix.

Remark 1. *Intuitively, the covariance matrix captures how different dimensions vary with respect to each other. Its eigenstructure reveals directions of greatest variance.*

2.2 Mathematical Formulation of PCA

PCA seeks a direction $\mathbf{w} \in \mathbb{R}^d$ that captures the maximum variance of the projected data. Formally:

Definition 2 (First Principal Component)**.** *The first principal component $\mathbf{w}_1$ is given by the solution to the optimization problem*

$$\max_{\mathbf{w}\in\mathbb{R}^d, \|\mathbf{w}\|=1} \mathbf{w}^\top \mathbf{C}\mathbf{w}.$$

Theorem 1. *The vector $\mathbf{w}_1$ that maximizes $\mathbf{w}^\top\mathbf{C}\mathbf{w}$ under the unit norm constraint is the eigenvector of $\mathbf{C}$ corresponding to its largest eigenvalue.*

Proof. Since $\mathbf{C}$ is symmetric and positive semi-definite, it can be eigen-decomposed as

$$\mathbf{C} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top,$$

where $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d]$ is orthonormal and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_d)$ with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \geq 0$. For any unit vector $\mathbf{w}$,

$$\mathbf{w}^\top\mathbf{C}\mathbf{w} = \mathbf{w}^\top\mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top\mathbf{w} = (\mathbf{V}^\top\mathbf{w})^\top\mathbf{\Lambda}(\mathbf{V}^\top\mathbf{w}).$$

Let $\mathbf{u} = \mathbf{V}^\top\mathbf{w}$. Since $\mathbf{V}$ is orthonormal, $\|\mathbf{u}\| = 1$. Thus,

$$\mathbf{w}^\top\mathbf{C}\mathbf{w} = \sum_{i=1}^{d} \lambda_i u_i^2.$$

This quadratic form is maximized when all of the weight is placed on the largest eigenvalue, i.e., $u_1 = 1$ and $u_2 = \cdots = u_d = 0$. This corresponds to $\mathbf{w} = \mathbf{v}_1$, the eigenvector of $\mathbf{C}$ with the largest eigenvalue λ_1. □

Subsequent principal components are defined similarly but under the constraint that they are orthogonal to the previously found components. The k-th principal component $\mathbf{w}_k$ is the eigenvector corresponding to the k-th largest eigenvalue λ_k.

2.3 Dimensionality Reduction via PCA

To reduce the dimension from d to $r < d$, we select the top r eigenvectors $\mathbf{w}_1, \ldots, \mathbf{w}_r$ and form a projection matrix:

$$\mathbf{W}_r = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_r].$$

We then map each original data point $\mathbf{x}_i$ to a lower-dimensional vector

$$\mathbf{y}_i = \mathbf{W}_r^\top \mathbf{x}_i.$$

Remark 2. *This projection maximizes the variance in the r-dimensional subspace and often reveals simpler structures in the data, making subsequent tasks (like clustering or classification) easier or more efficient.*

2.4 Example: PCA on Simple 2D Data

Consider a dataset in $\mathbb{R}^2$:

$$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}, \quad \mathbf{x}_i = \begin{pmatrix} x_{i1} \\ x_{i2} \end{pmatrix}.$$

If we compute the covariance matrix $\mathbf{C}$ and find its eigenvectors $\mathbf{v}_1$ and $\mathbf{v}_2$, projecting the data onto $\mathbf{v}_1$ (the direction of maximum variance) might reveal a linear trend. By then discarding $\mathbf{v}_2$, we reduce our data to one dimension while preserving as much information (variance) as possible.

In practice, one might do the following steps:

1. Mean-center the data: subtract the mean vector from each data point.
2. Compute the covariance matrix.
3. Find the eigen-decomposition of the covariance matrix.
4. Select the eigenvector(s) associated with the largest eigenvalue(s).
5. Project the data onto these eigenvectors to obtain a lower-dimensional representation.

3 Linear Algebra and Optimisation approaches to classification:

All optimization Note is contained in comp 331 Note.

3.1 Linear System Approaches

Perfect Separation ($\alpha = 0$):

- Linear systems aim to find the best decision boundary $A_i X^* = b$, where:
  - $A_i X^* < b$: Classifies as "Bad".
  - $A_i X^* > b$: Classifies as "Good".
- Example: If $A_1 X^* = 7.2$, it is classified as "Bad", whereas $A_2 X^* = 10.4$ is "Good".

Overlapping Classification Boundaries ($\alpha > 0$):

- In cases of overlapping groups, the decision boundary shifts:

$$\begin{aligned} A_i X^* &= b + \alpha \quad \text{(Adjusted boundary for "Good")}, \\ A_i X^* &= b - \alpha \quad \text{(Adjusted boundary for "Bad")}. \end{aligned}$$

- The adjustment ensures groups with overlaps are optimally separated based on predefined α values.

3.2 Multi-Criteria Linear Programming (MCLP)

Three-Group MC Model:

- Extends binary classification to handle multiple groups (G_1, G_2, G_3), each defined by adjusted boundaries:

$$\begin{aligned} b_1 - \alpha^1 < A_i X < b_1 + \alpha^1 \quad &\text{(Group } G_1\text{)}, \\ b_2 - \alpha^2 < A_i X < b_2 + \alpha^2 \quad &\text{(Group } G_2\text{)}. \end{aligned}$$

Key Parameters:

- α_i: Overlapping degree of group boundaries for a specific case (*external measurement*).
- α: Maximum overlapping across all groups.
- β_i: Distance of a case from its adjusted boundary (*internal measurement*).
- β: Minimum distance of all cases to their respective adjusted boundaries.
- h_i, k_i: Penalties assigned for misclassification related to α_i, β_i.

Optimization Problem:

- Minimize overlapping $(\sum \alpha_i)$.
- Maximize distance from boundaries $(\sum \beta_i)$.
- Subject to:

$$\begin{aligned} A_i X &= b + \alpha_i - \beta_i, \quad A_i \in G, \\ A_i X &= b - \alpha_i + \beta_i, \quad A_i \in B. \end{aligned}$$

Regret Function:

- Measures deviation from ideal values (α^*, β^*):

$$d_\alpha^+ = \sum \alpha_i + \alpha^* - \alpha^*,$$
$$d_\alpha^- = \alpha^* + \alpha_i - \sum \alpha_i,$$
$$d_\beta^+ = \sum \beta_i - \beta^*,$$
$$d_\beta^- = \beta^* - \sum \beta_i.$$

Objective:

$$\text{Minimize } (d_\alpha^+ + d_\alpha^-)^p + (d_\beta^+ + d_\beta^-)^p.$$

3.3 Algorithm for MCLP

1. Use `ReadCHD` to convert training and verifying data into a data matrix.
2. Use `GroupDef` to define groups $G_1, G_2, \ldots, G_s$.
3. Use `sGModel` to calculate the MCLP for the best s-group separation.
4. Evaluate training results using `Score`.
5. Predict classifications for verification data using `Predict`.

3.4 Multi-Criteria Non-Linear Programming (MCNLP)

Model Setup:

- Extends MCLP to include non-linear adjustments.
- Key variable: $\zeta_{i,j}$, the distance from case A_i to boundary b_j.
- Objective:

$$\text{Minimize } w_\alpha \sum_{i=1}^{n} \sum_{j=1}^{k} \alpha_{i,j} - w_\zeta \left(\sum_{i=1}^{n} \sum_{j=1}^{k-1} \zeta_{i,j} \right).$$

- Subject to:

$$A_i X = b_j + \alpha_{i,j} - \zeta_{i,j}, \quad 1 \leq j \leq k.$$

4 Support Vector Machines (SVM)

Support Vector Machines (SVMs) are powerful supervised learning methods used for classification and regression. This section covers the primal and dual formulations of SVM, their limitations, and the kernel trick.

4.1 Primal Formulation of SVM

The primal formulation seeks to minimize the loss while maximizing the margin between two classes. Given training data $\{(\mathbf{x}_i, y_i)\}$, where $\mathbf{x}_i$ is the feature vector and $y_i \in \{-1, +1\}$, the goal is to find a hyperplane:

$$\mathbf{w}^\top \mathbf{x} + b = 0$$

that separates the two classes.

Objective: The primal problem minimizes:

$$\min_{\mathbf{w}, b, \boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i$$

subject to:

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \ \forall i.$$

Here, ξ_i are slack variables to allow for margin violations, and $C > 0$ is a regularization parameter.

4.2 Dual Formulation of SVM

The dual formulation converts the primal problem into a quadratic programming (QP) problem, which is more efficient and allows the use of kernel functions for non-linear separations.

Objective: The dual problem is:

$$\max_{\alpha_i} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^\top \mathbf{x}_j)$$

subject to:

$$0 \leq \alpha_i \leq C, \quad \sum_{i=1}^{n} \alpha_i y_i = 0.$$

4.3 Limitations of Linear SVMs

Linear SVMs aim to find a hyperplane for separation. However:

- **Challenges in Non-Linear Data:** Not all datasets are linearly separable in the original feature space.
- **Example:** A circularly distributed dataset where classes are enclosed within one another cannot be separated by a straight hyperplane.
- **Need for Higher Dimensions:** Mapping data to a higher-dimensional space can enable separation using a hyperplane.

4.4 Kernel Trick

The kernel trick addresses the limitation of non-linear separability by computing the dot product of transformed features in a high-dimensional space.

Idea: Instead of explicitly transforming data into higher dimensions, the kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ computes:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j),$$

where ϕ is a mapping to a higher-dimensional space.

Decision Function: The decision function in the dual form depends on the dot product:

$$f(\mathbf{x}) = \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}),$$

where $K(\mathbf{x}_i, \mathbf{x})$ is the kernel function.

4.5 Applications of SVM

Support Vector Machines are widely used in:

- Text categorization and image classification.
- Bioinformatics for protein structure prediction.
- Financial forecasting for stock market trends.

4.6 Hard-Margin SVM: The Linearly Separable Case

Consider a labeled dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$. Assume the data is linearly separable, meaning there exists a hyperplane that can perfectly separate the two classes:

$$\mathbf{w}^\top \mathbf{x} + b = 0,$$

with no misclassifications.

Definition 3 (Functional Margin)**.** *For a hyperplane parameterized by* $(\mathbf{w}, b)$*, the functional margin of a data point* $(\mathbf{x}_i, y_i)$ *is defined as*

$$\hat{\gamma}_i = y_i(\mathbf{w}^\top \mathbf{x}_i + b).$$

The functional margin of the entire dataset is the minimum margin over all points:

$$\hat{\gamma} = \min_i \hat{\gamma}_i.$$

Definition 4 (Geometric Margin)**.** *The geometric margin, which is scale-invariant, is defined as*

$$\gamma = \min_i y_i \left(\frac{\mathbf{w}}{\|\mathbf{w}\|}^\top \mathbf{x}_i + \frac{b}{\|\mathbf{w}\|} \right).$$

This measures the distance from the closest point to the hyperplane $(\mathbf{w}, b)$*.*

Theorem 2 (SVM Maximum Margin Classifier)**.** *Among all hyperplanes that can separate the data, the one that maximizes the geometric margin is unique. This hyperplane can be found by solving:*

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \textit{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 \ \forall i.$$

Proof Sketch. To eliminate scale dependence, the constraints are fixed as $y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1$. Maximizing the geometric margin $\gamma = \frac{1}{\|\mathbf{w}\|}$ is equivalent to minimizing $\|\mathbf{w}\|^2$. The convex optimization problem has a unique solution due to strict convexity. The complete proof utilizes Lagrangian duality and the Karush-Kuhn-Tucker (KKT) conditions, showing a unique global minimum. □

4.7 Soft-Margin SVM: The Non-separable Case

Real-world data is often not perfectly separable. We introduce slack variables $\xi_i \geq 0$ to allow margin violations:

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0.$$

The optimization problem becomes

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i \quad \text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0,$$

where $C > 0$ is a penalty parameter balancing margin size and misclassification.

4.8 Kernels for Nonlinear Boundaries

If the data is not linearly separable in the original space, SVM can still find a linear separator in a high-dimensional feature space induced by a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$, where

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$$

for some mapping $\phi : \mathbb{R}^d \to \mathbb{R}^D$ (often $D \gg d$). This approach allows constructing nonlinear decision boundaries without explicitly computing the mapping ϕ.

4.9 Example: A Simple Linearly Separable Classification Task

Suppose we have a dataset in $\mathbb{R}^2$:

$$\{(\mathbf{x}_i, y_i)\}_{i=1}^n, \quad \mathbf{x}_i = \begin{pmatrix} x_{i1} \\ x_{i2} \end{pmatrix}, \ y_i \in \{-1, +1\},$$

and assume it is linearly separable. The steps to train a hard-margin SVM are:

1. Set up the optimization problem with constraints $y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1$.
2. Solve for $\mathbf{w}$ and b using quadratic programming methods.
3. Identify the support vectors: points that lie exactly on the margins $y_i(\mathbf{w}^\top \mathbf{x}_i + b) = 1$.
4. Use the resulting model $(\mathbf{w}, b)$ to classify new points by evaluating the sign of $(\mathbf{w}^\top \mathbf{x} + b)$.

If the data were not linearly separable, you would introduce slack variables and possibly use a kernel function for a nonlinear decision boundary.