Principal Component Analysis (PCA)

1 Introduction

Modern machine learning applications routinely encounter data that exists in extremely high-dimensional spaces, creating both computational and conceptual challenges. Consider the MNIST dataset of handwritten digits, where each image consists of $28 \times 28 = 784$ pixels, meaning every data point lives in a 784-dimensional space. Natural language data presents even greater challenges, with representations often requiring tens of thousands of dimensions corresponding to vocabulary size. In genomics, researchers work with gene expression data containing thousands of features for each sample, making direct analysis nearly impossible.

However, as the number of dimensions rises, we tend to see what is known as the **curse of dimensionality** as several problematic phenomena occur. Data points become increasingly sparse, with the volume of the space growing exponentially. This sparsity makes it difficult to identify patterns or clusters. Furthermore, the notion of euclidean distance becomes less meaningful in high dimensions, as the ratio between the minimum and maximum distances among points on a hypersphere approaches 1. Visualization, which is crucial for understanding data patterns and relationships, becomes impossible beyond three dimensions (for most of us).

However, most real-world high-dimensional data has much lower intrinsic dimensionality than the original space would suggest. This occurs because many features in real data are highly correlated with each other. For instance, in images, neighboring pixels tend to have similar values. In text, certain words frequently co-occur. This correlation structure means that the data effectively lies on or near a lower-dimensional manifold embedded in the high-dimensional space. Can we represent our original data with fewer features?

Principal Component Analysis is a ML approach that finds a new coordinate system for our data. This coordinate system consists of orthogonal directions called principal components. PCA has a few important properties: First, they capture the maximum possible variance in the data (keep as much information as possible). Second, they decorrelate the features, removing redundancy and simplifying subsequent analysis.

2 Mathematical Framework

2.1 Notation and Setup

We are given a data matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ where each row corresponds to one data point, so the i-th row is $\mathbf{x}^{(i)T}$.

The first step in PCA is centering the data around $\vec{0}$. We compute the mean vector $\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}^{(i)} \in \mathbb{R}^D$, which represents the centroid of our dataset in the D-dimensional space. Each data point is then centered by subtracting this mean: $\tilde{\mathbf{x}}^{(i)} = \mathbf{x}^{(i)} - \boldsymbol{\mu}$. This centering operation shifts the origin of our coordinate system to the center of the data cloud We collect these centered data points into a centered data matrix $\tilde{\mathbf{X}} \in \mathbb{R}^{N \times D}$.

We will work with the covariance matrix $\mathbf{\Sigma} \in \mathbb{R}^{D \times D}$. This matrix encodes all pairwise covariances between features and can be computed as:

$$\mathbf{\Sigma} = \frac{1}{N}\sum_{i=1}^{N} \tilde{\mathbf{x}}^{(i)}\tilde{\mathbf{x}}^{(i)T} = \frac{1}{N}\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}$$

The diagonal elements of $\mathbf{\Sigma}$ contain the variances of individual features, while off-diagonal elements contain covariances between pairs of features. This symmetric, positive semi-definite matrix completely characterizes the second-order statistics of our centered data.

3 The PCA Optimization Problem

3.1 Formulation 1: Maximum Variance

The first way to formulate PCA is as a variance maximization problem. We seek a unit vector $\mathbf{w} \in \mathbb{R}^D$ that defines a direction in the feature space. When we project our centered data points onto this direction, we obtain scalar projections $\{\mathbf{w}^T\tilde{\mathbf{x}}^{(i)}\}_{i=1}^{N}$. The variance of these projections measures how spread out the data is along this direction. Our goal is to find the direction that maximizes this spread:

$$\mathbf{w}_1 = \arg\max_{\|\mathbf{w}\|_2=1} \text{Var}\left(\{\mathbf{w}^T\tilde{\mathbf{x}}^{(i)}\}_{i=1}^{N}\right) \tag{1}$$

To derive a more tractable form, we note that the variance of the projections can be expressed in terms of the covariance matrix. Since the data is already centered, the variance of the projections equals their second moment:

$$
\begin{aligned}
\mathrm{Var}(\mathbf{w}^T\tilde{\mathbf{x}}) &= \mathbb{E}[(\mathbf{w}^T\tilde{\mathbf{x}})^2] && (2)\\
&= \mathbb{E}[\mathbf{w}^T\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T\mathbf{w}] && (3)\\
&= \mathbf{w}^T\mathbb{E}[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T]\mathbf{w} && (4)\\
&= \mathbf{w}^T\boldsymbol{\Sigma}\mathbf{w} && (5)
\end{aligned}
$$

This transformation reveals that maximizing the projected variance is equivalent to maximizing a quadratic form involving the covariance matrix:

$$\boxed{\mathbf{w}_1 = \arg\max_{\|\mathbf{w}\|_2=1} \mathbf{w}^T\boldsymbol{\Sigma}\mathbf{w}} \tag{6}$$

The constraint $\|\mathbf{w}\|_2 = 1$ is necessary because without it, we could arbitrarily increase the objective by scaling $\mathbf{w}$. In this formulation, we are finding the direction along which the data varies the most.

3.2 Formulation 2: Minimum Reconstruction Error

An alternative formulation of PCA focuses on finding a low-dimensional subspace that best approximates the original data in the least-squares sense. Suppose we want to find a K-dimensional subspace of $\mathbb{R}^D$ that best represents our data. We can represent this subspace by an orthonormal basis collected in a matrix $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_K] \in \mathbb{R}^{D\times K}$, where each column is a unit vector and the columns are mutually orthogonal, so $\mathbf{U}^T\mathbf{U} = \mathbf{I}_K$.

For each centered data point $\tilde{\mathbf{x}}^{(i)}$, we can compute its projection onto this subspace. The projection operation involves two steps. First, we compute the coordinates in the subspace: $\mathbf{z}^{(i)} = \mathbf{U}^T\tilde{\mathbf{x}}^{(i)} \in \mathbb{R}^K$. These K numbers represent the data point in the lower-dimensional space. Second, we can reconstruct an approximation of the original point: $\hat{\mathbf{x}}^{(i)} = \boldsymbol{\mu} + \mathbf{U}\mathbf{z}^{(i)} = \boldsymbol{\mu} + \mathbf{U}\mathbf{U}^T\tilde{\mathbf{x}}^{(i)}$. The matrix $\mathbf{U}\mathbf{U}^T$ acts as a projection operator onto the subspace spanned by the columns of $\mathbf{U}$.

The reconstruction error for a single point is $\|\tilde{\mathbf{x}}^{(i)} - \mathbf{U}\mathbf{U}^T\tilde{\mathbf{x}}^{(i)}\|_2^2$, which measures how much information is lost when we project to the lower-dimensional subspace and then reconstruct. The optimization problem seeks to minimize the average reconstruction error:

$$\boxed{\min_{\mathbf{U}^T\mathbf{U}=\mathbf{I}_K} \frac{1}{N}\sum_{i=1}^{N} \|\tilde{\mathbf{x}}^{(i)} - \mathbf{U}\mathbf{U}^T\tilde{\mathbf{x}}^{(i)}\|_2^2} \tag{7}$$

3.3 Equivalence of Formulations

A fundamental result in PCA is that these two seemingly different formulations—maximizing variance and minimizing reconstruction error—are mathematically equivalent. This equivalence stems from the Pythagorean theorem for orthogonal projections.

For any vector $\tilde{\mathbf{x}}^{(i)}$ and orthogonal projection matrix $\mathbf{U}\mathbf{U}^T$, we can decompose the vector into two orthogonal components: the projection onto the subspace and the residual perpendicular to it. The Pythagorean theorem tells us:

$$\|\tilde{\mathbf{x}}^{(i)}\|^2 = \|\mathbf{U}\mathbf{U}^T\tilde{\mathbf{x}}^{(i)}\|^2 + \|(\mathbf{I} - \mathbf{U}\mathbf{U}^T)\tilde{\mathbf{x}}^{(i)}\|^2 \tag{8}$$

The left side, $\|\tilde{\mathbf{x}}^{(i)}\|^2$, is the total variance of the data point (its squared distance from the origin after centering). The first term on the right, $\|\mathbf{U}\mathbf{U}^T\tilde{\mathbf{x}}^{(i)}\|^2$, is the variance captured by the projection. The second term, $\|(\mathbf{I} - \mathbf{U}\mathbf{U}^T)\tilde{\mathbf{x}}^{(i)}\|^2$, is the reconstruction error.

Since the total variance is fixed regardless of our choice of $\mathbf{U}$, maximizing the captured variance is equivalent to minimizing the reconstruction error. This elegant relationship means that PCA simultaneously achieves two desirable goals: it finds the subspace that preserves the most variance and provides the best approximation to the original data.

4 Solution via Eigendecomposition

4.1 Finding the First Principal Component

To solve the optimization problem for the first principal component, we employ the method of Lagrange multipliers. We want to maximize $\mathbf{w}^T\boldsymbol{\Sigma}\mathbf{w}$ subject to the constraint $\mathbf{w}^T\mathbf{w} = 1$. The Lagrangian function combines the objective and constraint:

$$\mathcal{L}(\mathbf{w}, \lambda) = \mathbf{w}^T\boldsymbol{\Sigma}\mathbf{w} - \lambda(\mathbf{w}^T\mathbf{w} - 1) \tag{9}$$

Taking the gradient with respect to $\mathbf{w}$ and setting it equal to zero yields the first-order optimality condition:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 2\boldsymbol{\Sigma}\mathbf{w} - 2\lambda\mathbf{w} = 0 \tag{10}$$

This simplifies to the eigenvalue equation:

$$\boxed{\boldsymbol{\Sigma}\mathbf{w} = \lambda\mathbf{w}} \tag{11}$$

This critical equation tells us that the optimal direction $\mathbf{w}$ must be an eigenvector of the covariance matrix $\boldsymbol{\Sigma}$, with λ being the corresponding eigenvalue.

To determine which eigenvector to choose, we substitute the eigenvalue equation back into our objective function. Multiplying both sides by $\mathbf{w}^T$ from the left:

$$\mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w} = \lambda \mathbf{w}^T \mathbf{w} = \lambda \tag{12}$$

This reveals that the variance along direction $\mathbf{w}$ equals the eigenvalue λ. Therefore, to maximize the variance, we should choose the eigenvector corresponding to the largest eigenvalue. This eigenvector defines the first principal component—the direction of maximum variance in the data.

4.2 Finding All Principal Components

The complete solution to PCA involves finding all principal components, not just the first. A remarkable property of the covariance matrix, which is symmetric and positive semi-definite, is that its eigenvectors form an orthogonal basis for $\mathbb{R}^D$. This means we can find D orthogonal directions that completely span the space.

Let the eigendecomposition of the covariance matrix be:

$$\boldsymbol{\Sigma} = \mathbf{Q} \boldsymbol{\Lambda} \mathbf{Q}^T \tag{13}$$

where $\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_D]$ is an orthogonal matrix whose columns are the eigenvectors, and $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_D)$ is a diagonal matrix of eigenvalues. We order these such that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D \geq 0$. The non-negativity of eigenvalues follows from the positive semi-definiteness of $\boldsymbol{\Sigma}$.

The principal components are then:

$$\boxed{\mathbf{w}_k = \mathbf{q}_k, \quad k = 1, \ldots, D} \tag{14}$$

Each principal component $\mathbf{w}_k$ captures variance λ_k, and they are ordered by decreasing importance. The orthogonality of eigenvectors ensures that each principal component captures different aspects of the variation in the data, with no redundancy between components.

5 The PCA Algorithm

Algorithm 1 Principal Component Analysis

```
1: **Input**: Data matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$, number of components $K$
2: **Training Phase:**
3: Compute mean: $\boldsymbol{\mu} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}^{(i)}$
4: Center data: $\tilde{\mathbf{X}} = \mathbf{X} - \mathbf{1}_N \boldsymbol{\mu}^T$
5: Compute covariance: $\boldsymbol{\Sigma} = \frac{1}{N} \tilde{\mathbf{X}}^T \tilde{\mathbf{X}}$
6: Eigendecomposition: $[\mathbf{Q}, \boldsymbol{\Lambda}] = \text{eig}(\boldsymbol{\Sigma})$
7: Sort by eigenvalues: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D$
8: Select top $K$ eigenvectors: $\mathbf{U} = [\mathbf{q}_1, \ldots, \mathbf{q}_K]$
9: **Projection Phase:**
10: For new data $\mathbf{x}$:
11:     Center: $\tilde{\mathbf{x}} = \mathbf{x} - \boldsymbol{\mu}$
12:     Project: $\mathbf{z} = \mathbf{U}^T \tilde{\mathbf{x}}$
13: **Output**: Projected data $\mathbf{z} \in \mathbb{R}^K$, basis $\mathbf{U}$, mean $\boldsymbol{\mu}$
```

6 Properties of PCA

6.1 Decorrelation

One of the most important properties of PCA is that it produces decorrelated features. When we project our data onto the principal components, the resulting features have zero covariance with each other. Mathematically, if we denote the projected data matrix as $\mathbf{Z}$, where each row contains the K-dimensional representation of a data point, the covariance matrix of these new features is:

$$\text{Cov}(\mathbf{Z}) = \mathbf{U}^T \boldsymbol{\Sigma} \mathbf{U} = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_K \end{bmatrix} \tag{15}$$

This diagonal structure means that the projected features are uncorrelated—knowing the value of one feature tells us nothing about the others. This decorrelation property is particularly valuable when the original features are highly correlated, as it removes redundancy and simplifies subsequent analysis. Each principal component captures a unique aspect of the variation in the data, making them ideal features for many machine learning algorithms that assume feature independence.

6.2 Variance Explained

Understanding how much information is preserved when we reduce dimensionality is crucial for practical applications of PCA. The eigenvalue λ_k associated with the k-th principal component tells us exactly how much variance is captured along that direction. The total variance in the original data equals the trace of the covariance matrix, which is the sum of all eigenvalues: $\text{trace}(\mathbf{\Sigma}) = \sum_{i=1}^{D} \lambda_i$.

When we retain only the first K principal components, we preserve $\sum_{k=1}^{K} \lambda_k$ units of variance. The proportion of variance explained by these components is:

$$\boxed{R^2(K) = \frac{\sum_{k=1}^{K} \lambda_k}{\sum_{i=1}^{D} \lambda_i}} \tag{16}$$

This ratio, often expressed as a percentage, provides a quantitative measure of how much information is retained in the reduced representation. For example, if $R^2(50) = 0.95$ for a dataset with $D = 1000$ features, we can represent the data using just 50 features while retaining 95% of the variance—a 20-fold reduction in dimensionality with minimal information loss.

6.3 Reconstruction Error

The flip side of variance explained is reconstruction error—how much information is lost when we use only K components. When we project data to a lower-dimensional subspace and then reconstruct it in the original space, the average squared reconstruction error is:

$$E(K) = \frac{1}{N} \sum_{i=1}^{N} \|\tilde{\mathbf{x}}^{(i)} - \mathbf{U}_K \mathbf{U}_K^T \tilde{\mathbf{x}}^{(i)}\|^2 = \sum_{j=K+1}^{D} \lambda_j \tag{17}$$

This elegant result shows that the reconstruction error equals the sum of the eigenvalues we discarded. Each eigenvalue represents the variance along its corresponding principal component, so by omitting components with small eigenvalues, we lose only the directions along which the data varies least. This is why PCA is optimal for linear dimensionality reduction—it minimizes reconstruction error by discarding the least important directions first.

7 Practical Considerations

7.1 Choosing the Number of Components

How many principal components should you use for your task? How many is enough to capture the information in your original dataset? It's a bit of the same question we asked for k-means and GMMs. The more PCs we keep around, the more information we keep around, but that wasn't the entire point. Each additional component provides less information than the last (diminishing returns). There are many ways to decide how many principal components to keep:

Method 1: Scree Plot Analysis. The scree plot visualizes eigenvalues λ_k against component index k. This plot typically shows a sharp decline in eigenvalues for the first few components, followed by a more gradual decrease. The "elbow" point, where the curve transitions from steep to shallow, suggests a natural cutoff. Components beyond this point often capture primarily noise rather than meaningful structure. However, identifying the elbow can be subjective, especially when the transition is gradual.

Method 2: Variance Threshold. A more quantitative approach sets a target for the cumulative proportion of variance explained. Common thresholds are 90% or 95% of total variance. We choose the smallest K such that $R^2(K) \geq 0.95$. This method provides a clear, objective criterion and ensures that most information is retained. The appropriate threshold depends on the application—visualization might tolerate 80% variance retention, while critical analysis might require 99%.

Method 3: Cross-Validation. When PCA serves as preprocessing for a supervised learning task, cross-validation can determine the optimal number of components. We evaluate the downstream task's performance (e.g., classification accuracy) for different values of K using held-out data. This approach directly optimizes for the ultimate goal rather than variance retention, sometimes revealing that fewer components yield better generalization by reducing overfitting.

7.2 Computational Approaches

7.2.1 Direct Eigendecomposition

The straightforward approach computes the covariance matrix explicitly and then performs eigendecomposition. This method works well when the number of features D is smaller than the number of samples N. The computational complexity breaks down as follows: computing the covariance matrix requires $O(ND^2)$ operations, as we must compute all D^2 entries, each requiring a sum over N samples. The eigendecomposition of the $D \times D$ covariance matrix requires $O(D^3)$ operations using standard algorithms. Thus, the total complexity is $O(ND^2 + D^3)$.

This approach becomes prohibitive for high-dimensional data. For instance, with $D = 10,000$ features, storing the covariance matrix alone requires 800 MB of memory (assuming double precision), and the eigendecomposition becomes computationally intensive.

7.2.2 Singular Value Decomposition (SVD)

SVD offers a more efficient and numerically stable approach to PCA, especially for high-dimensional data. Given the centered data matrix $\tilde{\mathbf{X}}$, we compute its SVD:

$$\tilde{\mathbf{X}} = \mathbf{U}\mathbf{S}\mathbf{V}^T \tag{18}$$

The columns of $\mathbf{V}$ are the principal components, and the eigenvalues are related to the singular values by $\lambda_i = s_i^2/N$. This approach never explicitly forms the covariance matrix, saving both memory and computation. Modern SVD algorithms can efficiently compute only the top K singular values and vectors, with complexity $O(NDK)$ for $K \ll \min(N, D)$. Additionally, SVD algorithms are numerically more stable than eigendecomposition, particularly for ill-conditioned matrices.

8 Limitations and Extensions

8.1 Limitations of PCA

Despite its widespread use, PCA has several fundamental limitations that restrict its applicability in certain scenarios.

Linearity represents perhaps the most significant limitation. PCA can only identify linear relationships between variables and finds only linear subspaces. If data lies on a nonlinear manifold—such as points on a spiral or sphere—PCA cannot discover this structure effectively. It will attempt to find the best linear approximation, potentially requiring many components to approximate what could be described simply in appropriate nonlinear coordinates.

The **orthogonality constraint** requiring principal components to be mutually orthogonal, while mathematically elegant, may not reflect the true structure of the data. In many real-world scenarios, the meaningful directions of variation are not orthogonal. For instance, in face images, variations due to lighting and pose might not be perpendicular in pixel space, yet PCA forces them to be orthogonal.

PCA captures **global structure** by finding directions that explain variance across the entire dataset. This global perspective can miss important local patterns. For instance, in a dataset containing multiple clusters, PCA might find components that span across clusters rather than components that describe variation within each cluster. This makes PCA less suitable for data with complex, multi-modal distributions.

Interpretability often suffers because principal components are dense linear combinations of all original features. The first principal component might involve contributions from every single original variable, making it difficult to understand what aspect of the data it represents. This lack of sparsity contrasts with the human preference for simple explanations involving only a few variables.

Finally, PCA exhibits **sensitivity to outliers** because the covariance matrix, which drives the entire analysis, can be heavily influenced by extreme values. A single outlier can significantly affect the estimated covariance, potentially rotating the principal components away from the directions that best describe the majority of the data.

8.2 Extensions

Kernel PCA extends PCA to capture nonlinear relationships by implicitly mapping data to a higher-dimensional feature space using the kernel trick. Instead of performing PCA on the original data, it operates in this transformed space where nonlinear patterns become linear. Common kernels include polynomial and radial basis function (RBF) kernels, each capturing different types of nonlinearity.

Sparse PCA addresses the interpretability issue by adding sparsity constraints to the optimization problem, encouraging principal components that involve only a subset of features. This makes components more interpretable—for instance, a sparse principal component might involve only genes from a specific biological pathway rather than small contributions from all genes.

Robust PCA reformulates the problem to handle outliers and corrupted observations. It decomposes the data matrix as $\mathbf{X} = \mathbf{L} + \mathbf{S}$, where $\mathbf{L}$ is a low-rank matrix capturing the clean data structure and $\mathbf{S}$ is a sparse matrix containing outliers and corruptions. This separation allows the method to identify the principal components of the clean data even in the presence of significant contamination.

Probabilistic PCA recasts PCA in a probabilistic framework, treating the observed data as noisy observations of a lower-dimensional latent variable. This Bayesian approach provides several advantages: it naturally handles missing data, provides uncertainty estimates for projections, and allows for principled model selection using marginal likelihood.

Incremental PCA adapts the algorithm for streaming data or datasets too large to fit in memory. Instead of computing the covariance matrix from all data at once, it updates the principal components incrementally as new data arrives, making it suitable for online learning scenarios.

8.3 Related Methods

Several related techniques share conceptual similarities with PCA while addressing different objectives or assumptions.

Factor Analysis also seeks a lower-dimensional representation but explicitly models observation noise. While PCA assumes all variance is signal, factor analysis separates variance into common factors and unique noise for each variable. This makes it more appropriate when measurements are known to be noisy.

Independent Component Analysis (ICA) goes beyond decorrelation to find statistically independent components. While PCA produces uncorrelated components (zero covariance), ICA seeks components that are statistically independent (no relationship of any kind). This stronger requirement makes ICA suitable for source separation problems, such as separating mixed audio signals.

t-SNE and UMAP are modern nonlinear dimensionality reduction techniques optimized specifically for visualization. Unlike PCA's global variance preservation, these methods focus on preserving local neighborhood structure, often revealing clusters and patterns that PCA cannot detect. However, they are typically used only for visualization, not for general dimensionality reduction.

Autoencoders represent the neural network approach to dimensionality reduction. The encoder network maps data to a lower-dimensional representation, while the decoder reconstructs the original data. Linear autoencoders with squared error loss recover the same solution as PCA, but nonlinear autoencoders can capture complex nonlinear relationships.