Eigen-Sushi

Van Pham

November 2025

1 Introduction

Principal Component Analysis (PCA) is an algorithm used to reduce the dimensionality of data. It enables visualization of high-dimensional data, extracts important features, identifies patterns, and reduces input size for machine learning models.

Given the original data that has m variables, an observation (or data point) can be represented as a vector $\mathbf{v} \in \mathbb{R}^m$. PCA finds k orthogonal vectors (known as principal components), with $k < m$, that span a k-dimensional space. Each observation is then projected onto this space, resulting in a lower-dimensional vector $u \in \mathbb{R}^k$ that preserves as much information of the original data as possible. Preserving information in this context is equivalent to maximizing variance between data points, because directions with closely clustered tend to correspond to noise rather than signal (Roughgarden and Valiant, 2024). Consider two lines of data points. In the first line, 10 points are tightly clustered together. Since these points are very similar, each point contributes little unique information, so removing one point does not significantly affect the overall structure. In the second line, 10 points are more spread out. Here, each point contains more unique information.

This report explains the mechanics of PCA and analyzes both its runtime and its correctness. It then introduces an alternative approach for performing PCA and illustrates how this method can be applied in practice

2 Preliminaries

1. The covariance matrix is a symmetric matrix that represents the covariances between all pairs of variables. It describes the linear relationships between them. It can be expressed as:
   $C = \frac{1}{n}\sum_{i=1}^{n}(X_i - \mu)(X_i - \mu)^T$
   where each X_i is an observation $\in \mathbb{R}^m$, and $\mu \in \mathbb{R}^m$ is the mean vector. The product of $(X_i - \mu)(X_i - \mu)^T$ produces a $m \times m$ matrix for the i^{th} observation. Summing over all n and divide by n gives the average covariance across the dataset. Another way of expressing the covariance matrix is: $C = \mathbb{E}[(X_i - \mu)(X_i - \mu)^T]$.

2. Variance is how much the data spread out from their mean. Given a random variable X, the variance of X can be denoted as:
   $Var(X) = \mathbb{E}[(X - \mathbb{E}[X])^2]$

3. Eigenvector is a vector that when multiplied by a matrix A, its direction does not change but scaled by a constant, known as the eigenvalue. This can be denoted as: $A\bar{v} = \lambda\bar{v}$ where $\bar{v}$ is the eigenvector, and λ is the corresponding eigenvalue.

4. Given two matrices A and B, $(AB)^\top = B^\top A^\top$

5. Eigendecomposition is a technique used to find the eigenvectors and eigenvalues of a symmetric matrix. A symmetric matrix A can be decomposed into: $A = Q\Lambda Q^\top$ where Q is a matrix of eigenvectors,Λ is a diagonal matrix containing the eigenvalues. The i^{th} eigenvalue in Λ is associated with the i^{th} eigenvector in Q (i^{th} column).

6. Singular value decomposition is a technique for factorizing a matrix, and is especially useful for non-square ones (number of rows and columns is unequal). A decomposition of a $n \times m$ matrix A, can be broken into 3 matrices: $A = U\Sigma V^\top$,where U is a $n \times n$ matrix, containing left singular vectors. Σ is a $n \times m$ rectangular diagonal matrix, containing singular values. V is a $m \times m$ matrix that contains the right singular vectors. Besides, the columns of U and V are orthonormal, meaning $v_1 \in U, v_2 \in U, v_1^\top v_2 = 0$ if $v_1 \neq v_2$, and $v_1^\top v_2 = 1$ if $v_1 = v_2$. This leads to a nice property: $U^\top U = I$, and $V^\top V = I$, where I is the identity matrix.

3 Algorithm

3.1 Method

This section goes over how principal component analysis is performed. It is a quite simple algorithm with the following steps:

1. Center the data: For each variable (a column) in the dataset, subtract its mean from every observation so that the resulting variable has a mean of 0. To do this, first go through column i, with n observations and find the mean (denote this as $\bar{x_i}$). Then, go through each entry in column i and subtract it with $\bar{x_i}$. Repeat this process for all columns, and denote this centered matrix as X.

2. We want a $m \times m$ matrix, which captures covariance of the variables. This covariance matrix is computed by $C = \frac{1}{n} X^\top X$.

3. Find the eigenvalues and eigenvectors of the covariance matrix through eigendecomposition.

4. We will claim this for now (which will be proven in section 3.3): the eigenvector of the largest eigenvalue is the direction that captures the most variance. This eigenvector is known as the first principal component. If the goal to reduce the dimensionality of the original data to $\mathbb{R}^k$, choose $\bar{v_1}, \bar{v_2}, ..., \bar{v_k}$ such that $\lambda_1 \geq \lambda_2 \geq, ..., \geq \lambda_k$, where λ_i is the eigenvalue of $\bar{v_i}$ eigenvector.

3.2 Runtime

1. The centering of the data matrix goes through all entries in a column (n), and does that for all columns (m), so the runtime is $O(nm)$.

2. An entry in the covariance matrix ($X^\top X$) is computed by doing dot multiplication of a column in X with a row in $X^\top$, which is n operations. Since the dimension of the covariance matrix is $m \times m$, we need to compute each entry m^2 times, so the runtime is $O(m^2 n)$

3. Eigendecomposition of a $m \times m$ matrix takes $O(m^3)$ time.

The total runtime of PCA is $O(m^2 n + m^3)$

3.3 Correctness

This section answers the question why the eigenvector with the largest eigenvalue is the direction that captures the most variance.

1. We first look at how the first principal component is found, which will be the basis for finding the $2^{nd}, ..., k^{th}$ principal components. We denote this first principal component as $v \in \mathbb{R}^m$. Since the direction of v is what we care about, we limit its magnitude to be a unit vector, such that $||v|| = 1$, or $v^\top v = 1$

2. We project each observation in the dataset onto the direction of v. This gives a scalar value $v^\top X$, where X is a random observation vector. The resulting projected value is defined as $Y = v^\top X$.

3. We would like to find the variance of Y, which according to the formula is equal to:
$Var(Y) = \mathbb{E}[(Y - \mathbb{E}[Y])^2]$

4. We first find the expected value of Y:
$\mathbb{E}[Y] = \mathbb{E}[v^\top X] = v^\top \mathbb{E}[X]$ (since $v^\top$ is a vector, the values are constant and can be brought outside).
$\mathbb{E}[X] = \mu$, where μ is the mean vector of each variable. Therefore $\mathbb{E}[v^\top X] = v^\top \mu$

5. Derive from the variance formula (step 3), we get:
$Var(Y) = \mathbb{E}[(Y - \mathbb{E}[Y])^2]$
$Var(Y) = \mathbb{E}[(v^\top X - v^\top \mu)^2]$
$Var(Y) = \mathbb{E}[(v^\top (X - \mu))^2]$
$Var(Y) = \mathbb{E}[(v^\top (X - \mu))(v^\top (X - \mu))]$

6. Since $(v^\top (X - \mu)$ is a scalar
$(v^\top (X - \mu)) = (v^\top (X - \mu))^\top = (X - \mu)^\top v$
Plug this back in the above equation, we get:
$Var(Y) = \mathbb{E}[(v^\top (X - \mu))((X - \mu)^\top v)]$
Since $v^\top$ and v are constants, they can be taken out, and we have:
$v^\top \mathbb{E}[(X - \mu)(X - \mu)^\top] v$
The expectation part is the definition of a covariance matrix, hence we can define $Var(Y) = v^\top C v$.

7. We need to find v, with the constraint $||v|| = 1$, such that $v^\top C v$ is maximized. In other words, this is finding the maxima of the function. To incorporate the constraint, a technique called Lagrange multiplier is used, and the following equation is obtained:
$v^\top C v + \alpha(1 - v^\top v)$, where α is a constant (Bishop, 2016)

8. Taking the derivative with respect to v and setting it to 0, we get:
$Cv = \alpha v$.
Multiply $v^\top$ to the left:
$v^\top C v = v^\top \alpha v$.
Since α is a constant, move it to the left.
$v^\top C v = \alpha v^\top v$
$v^\top C v = \alpha$ (since $v^\top v = 1$)
$Var(Y) = \alpha$

9. To maximize the variance, find the vector v such that $v^\top C v$ produces the largest α. Now it is obvious that v should be an eigenvector; moreover, it should be the eigenvector with the largest eigenvalue. Supposed λ is the largest eigenvalue, and v is the corresponding eigenvector:
$v^\top (Cv) = \alpha$
$v^\top \lambda v = \alpha$
$\lambda v^\top v = \alpha$
$\lambda = \alpha$

4 Alternative Method

There exists another way of computing the principal components without the need of building a covariance

matrix and performing eigendecomposition. The matrix is directly decomposed through singular value decomposition. In fact, this method is considered to be better than the original, and is widely implemented by many modern libraries. This is due to it being more numerically stable since matrix multiplication (for the covariance matrix) is expensive and prone to errors. The following proves why SVD is equivalent to performing eigendecomposition on a covariance matrix.

1. Given that we already centered our dataset and got the matrix X, which can be decomposed into:
   $X = U\Sigma V^\top$
   The rank of X, denoted as $r = rank(X) \leq min(n, m)$, determines the number of linearly independent vectors in X; or in other words, X spans an r-dimension space. Vectors beyond this rank are redundant because they can be expressed as linear combinations of others. Therefore, in practice, we often compute a compacted SVD of X, keeping the only first r singular values and their corresponding vectors.
   $X = U_r\Sigma_r V_r^\top$
   U_r is a $n \times r$ matrix, Σ_r is a $r \times r$ square matrix, and V is a $m \times r$ matrix.

2. Earlier, the covariance matrix C is computed by:
   $C = \frac{1}{n}X^\top X$,
   With SVD, C can be rewritten as:
   $C = \frac{1}{n}(U_r\Sigma_r V_r^\top)^\top U_r\Sigma_r V_r^\top$

3. $(U_r\Sigma_r V_r^\top)^\top$ is equivalent to $V_r\Sigma_r^\top U_r^\top$.

4. $C = \frac{1}{n}V_r\Sigma_r^\top U_r^\top U_r\Sigma_r V_r^\top$. Since the vectors in U are orthonormal, $U^T U = I$.
   $C = \frac{1}{n}V_r\Sigma_r^\top\Sigma_r V_r^\top$.

5. Σ_r is a diagonal square matrix; therefore, $\Sigma_r^\top = \Sigma_r$.
   $C = \frac{1}{n}V_r\Sigma_r^2 V_r^\top$. Move $\frac{1}{n}$ to the right of V_r
   $C = V_r(\frac{1}{n}\Sigma_r^2)V^\top$

6. From derivation above, we obtain the covariance matrix in the form:
   $C = V_r(\frac{1}{n}\Sigma_r^2)V^\top$
   This shows that principal component analysis can be computed directly through SVD, such that right singular vectors V contains to the eigenvectors of the covariance matrix, and the eigenvalues are given by $\frac{1}{n}\Sigma_r^2$.

Instead of computing all r principal components from SVD, assume that only the top k components are required. In this case, truncated SVD can be used, and its runtime is $O(nmk)$, which is significantly faster than performing a full decomposition.

5 Demonstration

This section presents a simple experiment exploring how to reduce an image's dimensionality.

Figure 1: Original image

Figure 1 is the original image, which was taken at my favorite sushi restaurant in Vietnam. The size of it is 3,117 KB, and the number of columns is 4032.

Figure 2: Reduced image

Figure 2 shows the result of reducing the image's dimensionality from 4032 down to 300. The file size is significantly reduced to 550 KB; however, the image is no longer visually recognizable. For visualization, we reconstruct the reduced image back to its original size to see the variance captured by the principal components.

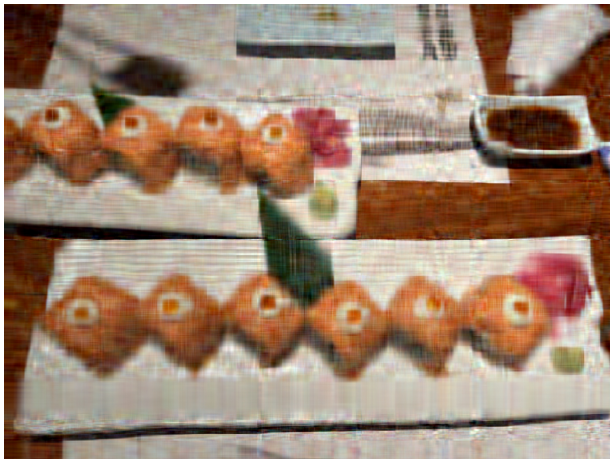

Figure 3: k = 25

Figure 4: k = 200

Figure 5: k = 300

It can be seen that with 25 principal components, the quality of the image is significantly reduced; however, the general coloring and structures of items (plates and food) are still captured. With $k = 200$ and $k = 300$, it looks identical to the original image.

6 Conclusion

The report begins with the foundational definition of PCA and the problems it addresses. It then details the traditional computation method, which involves calculating the covariance matrix and performing eigendecomposition. This section includes an analysis of the method's runtime complexity and a proof of its mathematical correctness. The focus then shifts to a more widely used technique: Singular Value Decomposition (SVD), which is numerically more stable by avoiding the costly and error-prone matrix multiplication needed to build the covariance matrix. It then demonstrates why SVD is equivalent to performing eigen-decomposition on a covariance matrix. The report concludes with a small experiment illustrating how PCA reduces dimensionality and retrains variance.

I have learned a lot throughout this report. On the technical side, I gained a deeper understanding of why eigenvectors capture variance, how matrices can be decomposed using eigen-decomposition and SVD, and how PCA can be applied in practice through the image experiment, where I observed the effect of dimensionality reduction and variance retention. Beyond the technical aspects, this project improved my research skills by teaching me how to efficiently skim papers and extract key information, and gave me valuable experience using Latex.

The experimental results show that using $k = 200$ principal components produces a reconstruction visually indistinguishable from $k = 300$. This suggests diminishing returns in the variance captured as k increases. Conceptually, this diminishing return behaves similarly to the $y = \log x$ function, where growth slows significantly as x becomes large. Therefore, an extension would be to investigate methods for selecting the optimal number of principal components.

Another possible extension is to explore variants of PCA, which will be briefly discussed. One such variant is sparse PCA, in which, after computing the principal components (each can be written as a linear combination $c_1m_1 + c_2m_2 + ... + c_mm_m$ where c is the weight and m is the variable), many weights become 0 through regularization or sparsity constraints. This approach improves interpretability by highlighting the most influential features and can also reduce computation, since zero weights remove corresponding multiplications during projection.

A more advanced extension is kernel PCA, which

overcomes the limitation that standard PCA only captures linear relationships in the data. If the underlying relationship is non-linear such as circular or curved, a linear line fails to capture the true shape. Kernel PCA uses a kernel function to map the data into a higher-dimensional space, where the relationship becomes linear, and PCA is performed on that transformed space.

7 References

1. Shlens, "A Tutorial on Principal Component Analysis," 1404.1100, Apr. 2014. [Online]. Available: https://arxiv.org/pdf/1404.1100

2. T. Roughgarden and G. Valiant, "CS168: The Modern Algorithmic Toolbox Lecture #7: Understanding and Using Principal Component Analysis (PCA)," 2024. [Online]. Available: https://web.stanford.edu/class/cs168/l/l7.pdf?

3. C. Bishop, Pattern Recognition and Machine Learning. 2006, Chapter 12.

4. E. Fetaya, J. Lucas, E. Andrews, and University of Toronto, "CSC 411 Lecture 12: Principle Components Analysis." [Online]. Available: https://www.cs.toronto.edu/~jlucas/teaching/csc411/lectures/lec12_handout.pdf?

5. Y. Qiu, "Large-Scale Eigenvalue Decomposition and SVD with RSpectra," Jul. 18, 2024. https://cran.r-project.org/web/packages/RSpectra/vignettes/introduction.html

6. NeuralNine, "Image compression using PCA in Python," YouTube. Jun. 08, 2022. [Online]. Available: https://www.youtube.com/watch?v=3aUshxvxGhY

7. Dr. Trefor Bazett, "Lagrange Multipliers — Geometric Meaning & Full example," YouTube. Nov. 27, 2019. [Online]. Available: https://www.youtube.com/watch?v=8mjcnxGMwFo

8. Caltech, "8.6 David Thompson (Part 6): Nonlinear Dimensionality Reduction: KPCA," YouTube. May 25, 2018. [Online]. Available: https://www.youtube.com/watch?v=HbDHohXPLnU

9. Libretexts, "20.4: Sparse principal component analysis," Biology LibreTexts, Mar. 17, 2021. Available: https://bio.libretexts.org/Bookshelves/Computational_Biology/Book%3A_Computational_Biology_-_Genomes_Networks_and_Evolution_(Kellis_et_al.)/20%3A_Networks_I-_Inference_Structure_Spectral_Methods/20.04%3A_Sparse_Principal_Component_Analysis