SVD and PCA Theory

Joseph S. Choi

03/2025

Singular Value Decomposition (SVD)

(Source: Gilbert Strang, MIT course; other online sources on linear algebra)

1. SVD Definition and Meaning:
   - An arbitrary matrix A, of size m x n (m rows, n columns), can be written as a Singular Value Decomposition (SVD) where $A = U\Sigma V^T$.
   - U and V are orthogonal, or orthonormal, matrices (i.e., $UU^T = I$, or $U^{-1} = U^T$ and similarly for V).
   - Note that SVD implies $AV = U\Sigma$, so V is an orthonormal basis set in the Row Space, that maps to an orthonormal basis set U in the Column Space scaled by the Singular Values Σ, when transformed by the matrix A!
     - $U^T A = \Sigma V^T$, so U^T transforms the columns of A, so it is termed *Column Space* (See Jonathon Shlens, https://arxiv.org/abs/1404.1100).
     - $V^T A^T = \Sigma^T U^T$, so V^T transforms the rows of A, so V is termed to be in the *Row Space* (See Jonathon Shlens, https://arxiv.org/abs/1404.1100).
   - Geometrically, we can consider U and V to be rotations, and Σ to be scaling. So every matrix can be represented as a rotation, followed by scalar stretching, then another rotation.
2. SVD Proof:
   0. $A = U\Sigma V^T$ can be shown by looking at the matrix $A^T A$ and AA^T. Clearly both are symmetric matrices (i.e., it is its own transpose), and are square (n x n, and m x m in size, respectively).
   1. First let's look at $A^T A$. Since it is square, we can find a set of eigenvectors and eigenvalues such that $(A^T A)V = V\Lambda_v$, where V is the matrix whose columns are the eigenvectors, and Λ_v is the diagonal matrix with the corresponding eigenvalues.
   2. Since $A^T A$ is symmetric, V is an orthogonal (orthornormal) matrix (or can be chosen to be so) since the eigenvectors of symmetric matrices are orthogonal and can be chosen to be orthonormal.
   (Source: Spectral Theorem states that a symmetric matrix can be written as $Q\Lambda_u Q^T$

where Q is an orthogonal matrix. See how symmetric matrices have an orthonormal eigenbasis: dept.math.lsa.umich.edu/~speyer/LinearAlgebraVideos/Lecture10a.pdf)

3. Also $\Lambda_{v,u}$ is Real because $A^T A$ is symmetric. (Source: https://ocw.mit.edu/courses/18-06-linear-algebra-spring-2010/resources/lecture-25-symmetric-matrices-and-positive-definiteness/ shows that eigenvalues are real for symmetric matrices.)

4. In fact, $A^T A$ is also positive semi-definite (PSD), i.e. its eigenvalues are positive or equal to 0.
    - Proof: For any vector v, $v^T A^T Av = (Av)^T(Av) = ||Av||_2^2 \geq 0$ by definition of the l_2 norm. Now suppose v is any eigenvector of $A^T A$, with eigenvalue λ. Then $0 \leq v^T A^T Av = v^T(A^T Av) = \lambda v^T v = \lambda ||v||_2^2$. Since $||v||_2^2 \geq 0$, λ must necessarily ≥ 0 as well. Q.E.D. (Proof for $A^T A$ being PSD was not explicitly found in the MIT Strang lecture videos, but the initial part of the proof was found at https://statisticaloddsandends.wordpress.com/2018/01/31/xtx-is-always-positive-semidefinite/)

5. Intermediate Summary:
    - Putting this all together, we have an n x n matrix
    $$(A^T A) = V\Lambda_v V^T,$$
    where V is orthogonal, and Λ_v is diagonal and contains the eigenvalues corresponding to V that are ≥ 0.
    - Similarly, the m x m matrix
    $$(AA^T) = U\Lambda_u U^T,$$
    where U is orthogonal, and Λ_u is diagonal and contains the eigenvalues corresponding to U that are ≥ 0.
    - Note that the eigenvalues of Λ_u and Λ_v are the same except for that one may have zero(s) that is(are) not present in the other. These contribute to the Null Space.

6. Defining Σ:
    - If $m = n$, then $\Sigma \equiv \Lambda_{v,u}$, since the 2 matrices should be the same.
    - If $m \neq n$, to accommodate the $\Lambda_{v,u}$ different sizes, let's define Σ to be m x n in size. Its top-left part will be the square root of $\Lambda_{v,u}$, whichever is smaller in dimensions. The rest will be filled with 0. (Note: Here in taking the square root, we have used the fact that $\Lambda_{v,u} \geq 0$, which comes from $A^T A$ or AA^T being positive semi-definite.)
    - Now note that $\Sigma\Sigma^T = \Lambda_u$, and is m x m in size. Similarly $\Sigma^T\Sigma = \Lambda_v$, and is n x n in size.

7. Putting pieces together for SVD:
   - Recall $(A^TA)V = V\Lambda$, which will be our starting point. This can be written as $(A^TA) = V\Lambda_v V^T$, since V is orthogonal.
   - Now insert Σ: $(A^TA) = V\Lambda_v V^T = V\Sigma^T\Sigma V^T$.
   - Since U is orthogonal, $U^TU = I_m$, , where I_m is the m x m Identity Matrix.
   - Now insert I_m: $(A^TA) = V\Sigma^T\Sigma V^T = V\Sigma^T I_m \Sigma V^T = V\Sigma^T U^T U\Sigma V^T$.
   - Factor: $(A^TA) = V\Sigma^T U^T U\Sigma V^T = (U\Sigma V^T)^T(U\Sigma V^T)$
   $$\therefore A = (U\Sigma V^T)$$
   .

3. How to decompose A using SVD:
   - Using the proof of SVD above, we now know how to explicitly construct U, V, Σ such that $A = (U\Sigma V^T)$.
   - Solving V: Find eigen solutions of (A^TA), i.e., $(A^TA)V = V\Lambda_v$.
   - Solving U: Find eigen solutions of (AA^T), i.e., $(AA^T)U = U\Lambda_u$.
   - Solving Σ: Take the square root of $\Lambda_{v,u}$, fill with zeros to make it same size as A.
     - Note: For Principal Component Analysis (PCA), build U, V, Σ so that the singular values are in descending order from top to bottom, which is proportional to the "importance" of the components.

Principal Component Analysis (PCA)

0. References:
   1. Jonathon Shlens, "A Tutorial on Principal Component Analysis," arXiv:1404.1100 (2014) https://arxiv.org/abs/1404.1100
      - Shlens uses (n) instead of (n-1) for the denominator in the variances and covariances. The latter is typically used in other sources, likely because others consider sampling of a population, whereas Shlens may be assuming the whole population is used (Shlens does mention this "practice" in footnote 2). In my summary, I have corrected this to (n-1),
      - SNR defined by Shlens is the ratio of the Signal Variance to the Noise Variance. While this seems to be what is necessary for the toy spring example provided, I don't think this is widely accepted as the definition of SNR. Typically in engineering, SNR is defined as Signal/Noise = (Mean of Signal)/(Standard Deviation of Noise). Perhaps a different word than "SNR" may be appropriate for the PCA paper.

2. Jeff Jauregui, "Principal component analysis with linear algebra" (August 31, 2012) https://www.math.union.edu/~jauregui/PCA.pdf

1. PCA Goal:
    - "PCA provides a roadmap for how to reduce a complex data set to a lower dimension to reveal the sometimes hidden, simplified structures that often underlie it."
    - "The goal of principal component analysis is to identify the most meaningful basis to re-express a data set. The hope is that this new basis will filter out the noise and reveal hidden structure."
        - "Is there another basis, which is a linear combination of the original basis, that best re-expresses our data set?"
2. PCA Assumptions:
    - "By positing reasonably good measurements, quantitatively we assume that directions with largest variances in our measurement space contain the dynamics of interest."
        - This means that the features with larger variances are more important. This sometimes is not necessarily true, and at times require scaling to correct.
3. Covariance Matrix:
    - Suppose we have m features (original m measurement "basis" vectors), and n measurements (samples of measurements, for example, at different times).
    - Let X be an m x n matrix containing all measurement data, where each of the m rows represents all measurements for the mth feature (or measurement type). Also the mean of each row (of n measurements or samples) is subtracted from each row (so each row of X has 0 mean).
    - Covariance Matrix $C_X \equiv \frac{1}{n-1}XX^T$
        - C_X is an m x m matrix.
        - Diagonals of C_X are the variances- Large values correspond to interesting structure, per assumption.
        - Off-diagonals of C_X are the *covariances*- Larger values correspond to redundancy (since the corresponding features can predict each other, and hence one can be removed with little impact).
4. PCA and newly transformed covariance matrix, C_Y, from C_X:
    - Goal of PCA: Find a transformation of basis so that the diagonals (variances, or relevant signals) of the new covariance matrix is maximized, and the off-diagonals (covariances, or redundancy) are minimized.

- Explicitly, find an orthonormal transformation $Y = PX$ such that the covariance matrix in Y, i.e. C_Y, is a Diagonal Matrix.
- Also let's sort by importance, using variance, so that each successive dimension is ranked in order (top-left being most important, and bottom-right element being least important).

5. PCA Construction:

0. (Shlens' paper describes eigenvector decomposition and SVD separately, which was confusing and disjointed for me. So I have re-organized this using SVD.)
1. Let X be the original m x n measurement matrix (m features/types, n samples/measurements). (Note, this is opposite of the SVD notes above)
2. Let $Y \equiv \frac{1}{\sqrt{n-1}} X^T$ be the n x m matrix ($Y = A$ in the SVD notes, with size index flipped).
3. $Y^T Y = \frac{1}{n-1} X X^T = C_X$ (the m x m Covariance Matrix).
4. By SVD, $(Y^T Y) = V \Lambda_v V^T = C_X$ (V is an m x m matrix)
5. Rearranging, $\Lambda_v = V^T C_X V$.
6. So V transforms C_X into the diagonal matrix Λ_v as desired by PCA!
   - The columns of V are the orthonormal eigenvectors of C_X (since $C_X V = V \Lambda_v$). "Therefore, the columns of V are the principal components of X."
   - In other words, V spans the row space $Y \equiv \frac{1}{\sqrt{n-1}} X^T$, and hence spans the column space of X.
7. Converting SVD for $A = X^T$ to SVD for the new matrix $Y = \frac{1}{\sqrt{n-1}} X^T$:
   - U, V remain the same.
   - Only the singular values need to be scaled to $\Sigma \rightarrow \frac{1}{\sqrt{n-1}} \Sigma$.
   - This can be seen, or rather justified, from the definition of SVD: $A = U \Sigma V^T$.
   - Caution: It is worth confirming that the SVD algorithm used confirms this. For NumPy.linalg.svd in 03/2025, this was true.
   - Python code to check given below.

```
# Do SVD on Y = 1/\sqrt{n-1} A:
U, s, Vh = np.linalg.svd(A, full_matrices=True)
Y = A / np.sqrt(A.shape[0] - 1)
Unew, snew, Vhnew = np.linalg.svd(Y, full_matrices=True)
# Compare
print('Is SVD U same from A vs. Y = 1/\sqrt{n-1} A?', np.allclose(U, Unew))
print('Is SVD s same from A vs. Y = 1/\sqrt{n-1} A?', np.allclose(s, snew))
print('Is SVD Vh same from A vs. Y = 1/\sqrt{n-1} A?', np.allclose(Vh, Vhnew))
```

```
# Check that s is scaled:
print('Is SVD s from Y = 1/\sqrt{n-1} A, same as s_original/\sqrt{n-1} \
where s_original is the SVD s from A?', \
np.allclose(s/np.sqrt(Xused.shape[0] - 1), snew))
```