14 Singular Value Decomposition

For any high-dimensional data analysis, one's first thought should often be: *can I use an SVD?* The singular value decomposition is an invaluable analysis tool for dealing with large high-dimensional data. In many cases, data in high dimensions, most of the dimensions do not contribute to the structure of the data. But filtering these takes some care since it may not be clear which ones are important, if the importance may come from a combination of dimensions. The singular value decomposition can be viewed as a way of finding these important dimensions, and thus the key relationships in the data.

On the other hand, the SVD is often viewed as a numerical linear algebra operation that is done on a matrix. It decomposes a matrix down into three component matrices. These matrices have structure, being orthogonal or diagonal.

The goal of this note is to bridge these views and in particular to provide geometric intuition for the SVD.

The SVD is related to several other tools which will also consider:

- **PCA (Principal Component Analysis)**: a geometric interpretation, after *centering* the data
- **Eigen-decomposition**: shares the same components after data has been made "square."
- **MDS (Multidimensional Scaling)**: Given just pairwise distances, convert to eigen-decomposition

Data. We will focus on a dataset $A \subset \mathbb{R}^d$ where A is a set of n "points." At the same time, we will think of A as a $n \times d$ matrix. Each row corresponds to a point, and each column corresponds to a dimension. *(Some interpretations reverse these so a column is a point, a row is a dimension – but they are really the same.)*

Then the goal will be to find a projection $\mu : \mathbb{R}^d \to \mathbb{R}^k$, where $k \ll d$ and in particular $\mathbb{R}^k \subset \mathbb{R}^d$, so that we minimize

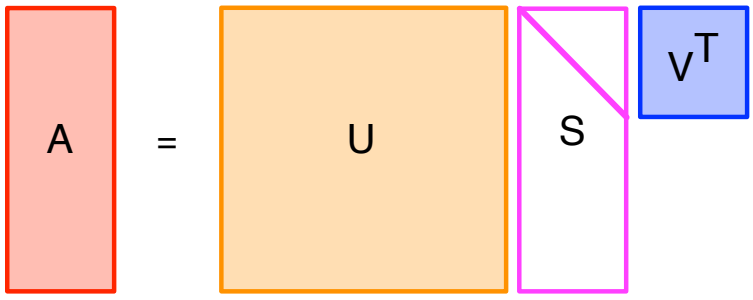
$$\sum_{p \in P} (p - \mu(p))^2.$$

The SVD will precisely provide us this answer!

14.1 The SVD Operator

First we document what the following operation in matlab does:

$$[U, S, V] = \texttt{svd}(A)$$



The backend of this (in almost any language) calls some very carefully optimized Fortran code as part of the LAPACK library. First of all, no information is lost since we can simply recover the original data as $A = USV^T$, up to numerical precision, and the Fortran library is optimized to provide very high numerical precision.

But there is more structure lurking in these matrices. S is $n \times d$ and almost all 0s, except for along the diagonal: $S = \mathsf{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$ where $r \leq d$ and where r is the *rank* of P. That is, S is a *diagonal matrix* with only entries on the diagonal. These values are (generally) output in non-increasing order so $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \geq 0$. They are known as the *singular values* of A.

Both U (size $n \times n$) and V (size $d \times d$) are *orthogonal matrices*. An orthogonal matrix is also a *rotation* matrix (more on this later), that can also allow mirror flips. They have the following properties:

- each column v_i has $\|v_i\| = 1$
- each pair of columns v_i, v_j have $\langle v_i, v_j \rangle = 0$
- Its transpose is its inverse $V^T = V^{-1}$, so $V^T V = I$.

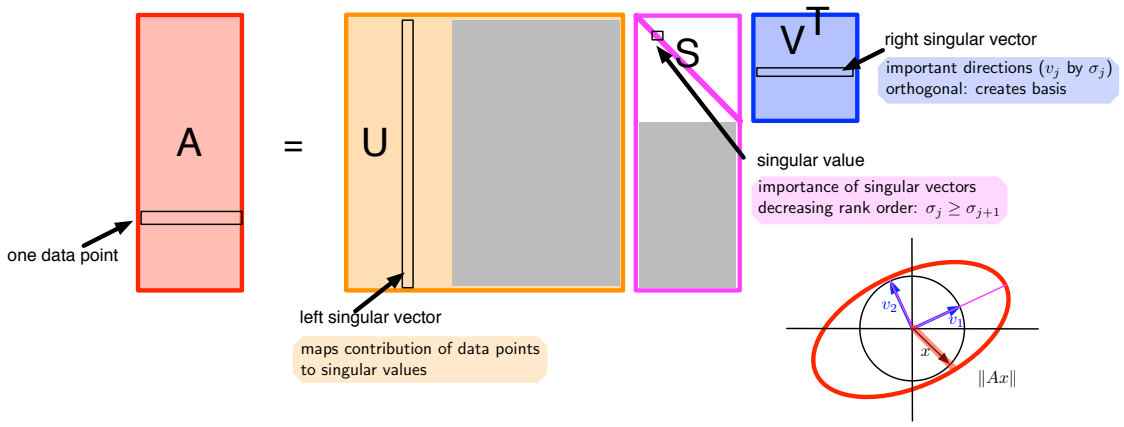
Moreover the columns (and rows) of V form a d-dimensional orthogonal basis (usually not the original basis). That is, for any $x \in \mathbb{R}^d$ we can write

$$x = \sum_{i=1}^{t} \alpha_i v_i$$

for some scalar $\alpha_i = \langle x, v_i \rangle$. This is the ith coordinate in the new basis.

This implies that for any $x \in \mathbb{R}^d$ that $\|Vx\| = \|x\|$ (it can only rotate or flip).

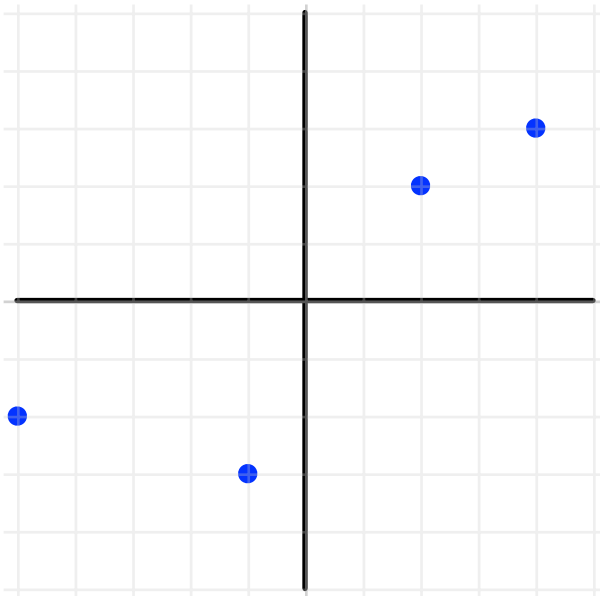
Moreover the columns of $V = [v_1\ v_2\ \ldots\ v_d]$ are known as the *right singular vectors* and the columns of $U = [u_1\ u_2\ \ldots\ u_n]$ are known as the *left singular vectors*.



14.1.1 Example

Consider the set of $n = 4$ points in $\mathbb{R}^2$ $\{a_1 = (4, 3), a_2 = (1, 2), a_3 = (-1, -3), a_4 = (-4, 2)\}$. Note, these are chosen so that average x- and average y-coordinate is 0; it is *centered*. We can equivalently write this as a the matrix

$$A = \begin{pmatrix} 4 & 3 \\ 2 & 2 \\ -1 & -3 \\ -5 & -2 \end{pmatrix}.$$



Then $[U, S, V] = \texttt{svd}(A)$ where

$$U = \begin{pmatrix} -0.6122 & 0.0523 & 0.0642 & 0.7864 \\ -0.3415 & 0.2026 & 0.8489 & -0.3487 \\ 0.3130 & -0.8070 & 0.4264 & 0.2625 \\ 0.6408 & 0.5522 & 0.3057 & 0.4371 \end{pmatrix},$$

$$S = \begin{pmatrix} 8.1655 & 0 \\ 0 & 2.3074 \\ 0 & 0 \\ 0 & 0 \end{pmatrix},$$

$$V = \begin{pmatrix} -0.8142 & -.5805 \\ -0.5805 & 0.8142 \end{pmatrix}.$$

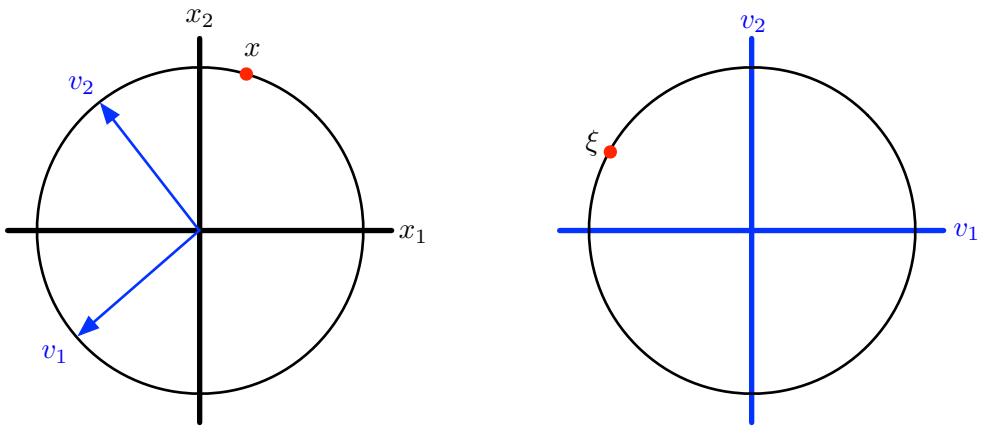
We will continue to use this example as we explain the geometry.

14.1.2 Geometry of the SVD

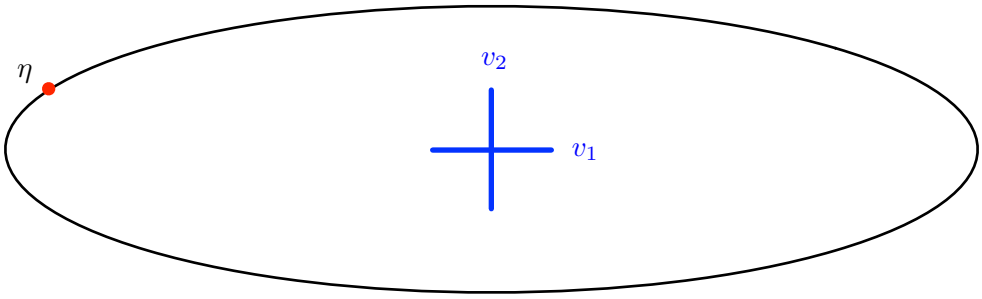
We will see how the matrix A transforms a circle in $\mathbb{R}^2$ to a two-dimensional ellipse the lives in $\mathbb{R}^4$. This ellipse will represent the size and magnitude of the principal components.

We will start with an arbitrary point x such that $\|x\| = 1$ (so it is on the unit circle), and see what happens in $b = Ax$. Specifically we will break down the process $b = USV^T x$ by examining $\xi = V^T x$, then $\eta = S\xi = SV^t x$ and then $b = U\eta = USV^T x$.

Step 1 ($\xi = V^T x$): Since V^T is orthogonal (and $x \in \mathbb{R}^2$) then it acts as a rotation. It puts x in the basis of V^T as a point ξ. Note that the orthogonal vectors v_1 and v_2 (of $V = [v_1, v_2]$) become the axis to describe ξ.



Step 2: ($\eta = S\xi$): Note that $\xi = (\xi_1, \xi_2)$ is still on a circle since it still has $\|\xi\| = 1$. The S matrix is a diagonal matrix, so it just scales the axis. Each ith axis is scaled according to σ_i. Specifically, $\eta_1 = \sigma_1 \xi_1$ and $\eta_2 = \sigma_2 \xi_2$, where ξ_1 and ξ_2 are coordinates in the basis along v_1 and v_2, respectively.



Step 3: ($b = U\eta$): We now apply another rotation. Notice that S had two rows that were all 0. This effectively scales η along two axes it did not know it had. But it sets these values to 0. So $\eta = (\eta_1, \eta_2, 0, 0) \in \mathbb{R}^4$.

Now we again use that U is a rotation (with possible mirror flips), but for points in $\mathbb{R}^4$. Each axis now represents the component along the direction of a point. Note that now $\|\eta\| = \|b\| = \|S\| = \|A\|$.

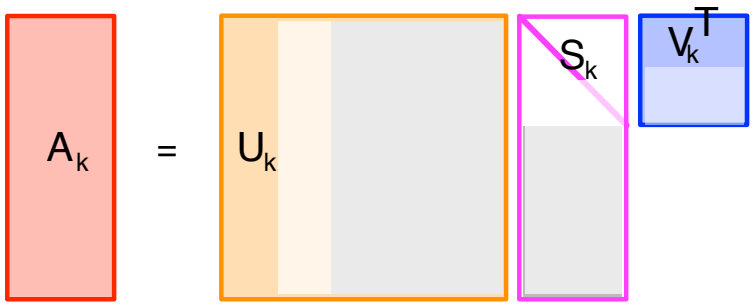
Unfortunately, this is harder to draw, but it looks like step 1, but in higher dimensions.

14.1.3 Best Rank-k Approximation

So how do we get this subspace that we project onto?

The vectors $V = [v_1, v_2, \ldots, v_d]$ are such that they describe the most *important* axis of the data in the following sense. The first right singular vector v_1 describes which direction has the most variance. The variance is precisely described by σ_1. Then since $v_2, \ldots, v_d$ are each orthogonal to v_1, this implies that v_2 is the direction (after v_1 has been factored out) that has the most variance.

So the k-dimensional subspace of $\mathbb{R}^d$ is defined by basis $[v_1, v_2, \ldots, v_k]$, the first k-right singular vectors.



So how large should k be? The amount of squared "mass" captured by v_k is σ_k^2. So if σ_{k+1} is small, we do not need to keep v_{k+1}. It means there is little variation along that direction, and each other directions not yet captured. Or use "elbow" technique where the difference between $\sigma_k - \sigma_{k+1}$ is large.

- In many statistical and numerical datasets, often σ_k decay quickly. Usually for k not too large σ_{k+1} is very close to 0.
- In many internet scale datasets (think Facebook graph), then typically σ_k decay slowly (they have a "heavy tail"). Often $\sum_{i=k+1}^{\infty} \sigma_i^2 \geq \frac{1}{10} \sum_{i=1}^{\infty} \sigma_i^2 = \frac{1}{10}\|A\|_F^2$, even at the appropriate cut-off k. (This may (or may not) indicate that SVD is the wrong approach to finding core structure.)

So what do we know:

- V does "bookkeeping" of moving original basis to new one,
- S stretches it accordingly (along new basis), and
- U describes results with respect to actual data points.

So to get the projection of the points A in the new subspace as $\mathbb{R}^k$ we create $A_k \in \mathbb{R}^k$ as

$$A_k = U_k S_k V_k^T$$

where $U_k = [u_1, u_2, \ldots, u_k]$, $S = \mathsf{diag}(\sigma_1, \sigma_2, \ldots, \sigma_k)$ and $V = [v_1, v_2, \ldots, v_k]$.

So the subspace is defined using just V_k. Then S_k describes the importance along each direction, and U_k relates it to the actual points.

The following two important optimality facts are known about A_k. Let $A_{i,j}$ be the entry in A at the intersection of the ith row and jth column. Let $[A_k]_{i,j}$ represent the same element for A_k.

- A_k is the rank k matrix that minimizes $\|A - A_k\|_F^2 = \sum_i \sum_j (A_{i,j} - [A_k]_{i,j})^2$, the *Frobenius* norm of the difference of A and A_k. In fact $\|A - A_k\|_F^2 = \sum_{i=k+1}^{d} \sigma_i^2$.
- A_k is the rank k matrix that minimizes $\|A - A_k\|_2 = \max_x \|(A - A_k)x\|/\|x\|$, the *spectral* norm of the difference of A and A_k. In fact, $\|A - A_k\|_2 = \sigma_{k+1}$.

An alternative interpretation, is that A_k captures the directions V_k in which A has the most variance.

14.2 Relationship to PCA, Eigendecomposition, and MDS

We next will relate the SVD to other common matrix analysis forms: PCA, Eigendecomposition, and MDS. One may find literature that uses slightly different forms of these terms (they are often intermingled), but I believe this is the cleanest, most consistent, mapping.

14.2.1 Relationship to PCA

Principle Component Analysis (PCA) is the result of the best rank-k SVD after centering the data.

Centering is adjusting the original input data A so that each column (each dimension) has an average value of 0. This is easier than it seems. Define $\bar{a}_j = \frac{1}{n}\sum_{i=1}^{n} A_{i,j}$. Then set each $\tilde{A}_{i,j} = A_{i,j} - \bar{a}_j$ to represent the entry in the ith row and jth column of centered matrix $\tilde{A}$.

There is a *centering matrix* $C_n = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^T$ where I_n is the $n \times n$ identity matrix, $\mathbf{1}$ is the all-ones column vector (of length n) and thus $\mathbf{1}\mathbf{1}^T$ is the all-ones $n \times n$ matrix. Then we can also just write $\tilde{A} = C_n A$.

Now to perform PCA on a data set A, we compute $[U, S, V] = \texttt{svd}(C_n A) = \texttt{svd}(\tilde{A})$.

Then the resulting singular values $\mathsf{diag}(S) = \{\sigma_1, \sigma_2, \ldots, \sigma_r\}$ are known as the *principle values*, and the top k right singular vectors $V_k = [v_1\ v_2\ \ldots\ v_k]$ are known as the top-k *principle directions*.

This often gives a better fitting to the data than just SVD. The SVD finds the best rank-k approximation of A, which is the best k-dimensional subspace (up to Frobenius and spectral norms) **which passes through the origin**. If all of the data is far from the origin, this can essentially "waste" a dimension to pass through the origin.

Note, this implies that PCA finds the best k-dimensional subspace V_k so $\sum_{a \in A} \|a - \pi_{V_k}(a)\|^2$ is minimized, where *projection* $\pi_{V_k}(a) = \arg\min_{x \in V_k} \|a - x\|$ is the point on subspace V_k (or in matrix terms, in the span of V_k) closest to a.

14.2.2 Relationship to Eigen-Decomposition

Recall that an eigenvalue λ and eigenvector v of a matrix M are such that $Mv = \lambda v$, and we usually normalize so that $\|v\| = 1$.

We can write

$$A^T A V = (VSU^T)(USV^T)V = VS^2$$

so the right singular vectors v_i are the eigenvectors of $A^T A$. Similarly

$$AA^T U = (USV^T)(VSU^T)U = US^2$$

so the left singular vectors u_i are the eigenvectors of AA^T. Thus also the squared singular values σ_i^2 are eigenvalues of $A^T A$ and of AA^T.

14.2.3 Relationship to MDS

A *distance matrix* D is an $n \times n$ matrix where entry $D_{i,j}$ is the distance between the ith and the jth point.

Multi-Dimensional Scaling (MDS) has the goal of taking a distance matrix D for n points and giving low-dimensional (typically) Euclidean coordinates to these points so that the embedded points have similar spatial relations to that described in D. If we had the original data set A which resulted in D, we could just apply PCA to find the embedding. It is important to note, in the setting of MDS we are typically just given D, and *not* the original data A. However, as we will show next, we can derive AA^T using only D.

A *similarity matrix* M is an $n \times n$ matrix where entry $M_{i,j}$ is the similarity between the ith and the jth data point. The similarity often associated with Euclidean distance $\|a_i - a_j\|$ is the standard inner (or dot product) $\langle a_i, a_j \rangle$. We can write

$$\|a_i - a_j\|^2 = \|a_i\|^2 + \|a_j\|^2 - 2\langle a_i, a_j \rangle,$$

and hence

$$\langle a_i, a_j \rangle = \frac{1}{2}\left(\|a_i\|^2 + \|a_j\|^2 - \|a_i - a_j\|^2\right). \tag{14.1}$$

Next we observe that for the $n \times n$ matrix AA^T the entry $[AA^T]_{i,j} = \langle a_i, a_j \rangle$. So it seems hopeful we can derive AA^T from D using equation (14.1). However, we need $\|a_i\|^2 = \langle a_i, a_i \rangle$ and $\|a_j\|^2 = \langle a_j, a_j \rangle$.

Since the embedding has an arbitrary shift to it (if we add a shift vector s to *all* embedding points, then no distances change), then we can arbitrarily choose a_1 to be at the origin. Then $\|a_1\|^2 = 0$ and $\|a_j\|^2 = \|a_1 - a_j\|^2 = D_{1,j}^2$. Using this assumption and equation (14.1), we can then derive the similarity matrix AA^T. Then we can run the eigen-decomposition on AA^T are use the coordinates of each point along the first k eigenvectors to get an embedding. This is known as *classical MDS*.

It is often used for k as 2 or 3 so the data can be easily visualized.

There are several other forms that try to preserve the distance more directly, where as this approach is essentially just minimizing the squared residuals of the projection from some unknown original (high-dimensional embedding). One can see that we recover the distances with no error if we use all n eigenvectors. This implies the cool fact that any $n \times n$ distance matrix can be embedded with 0 error using n dimensions.