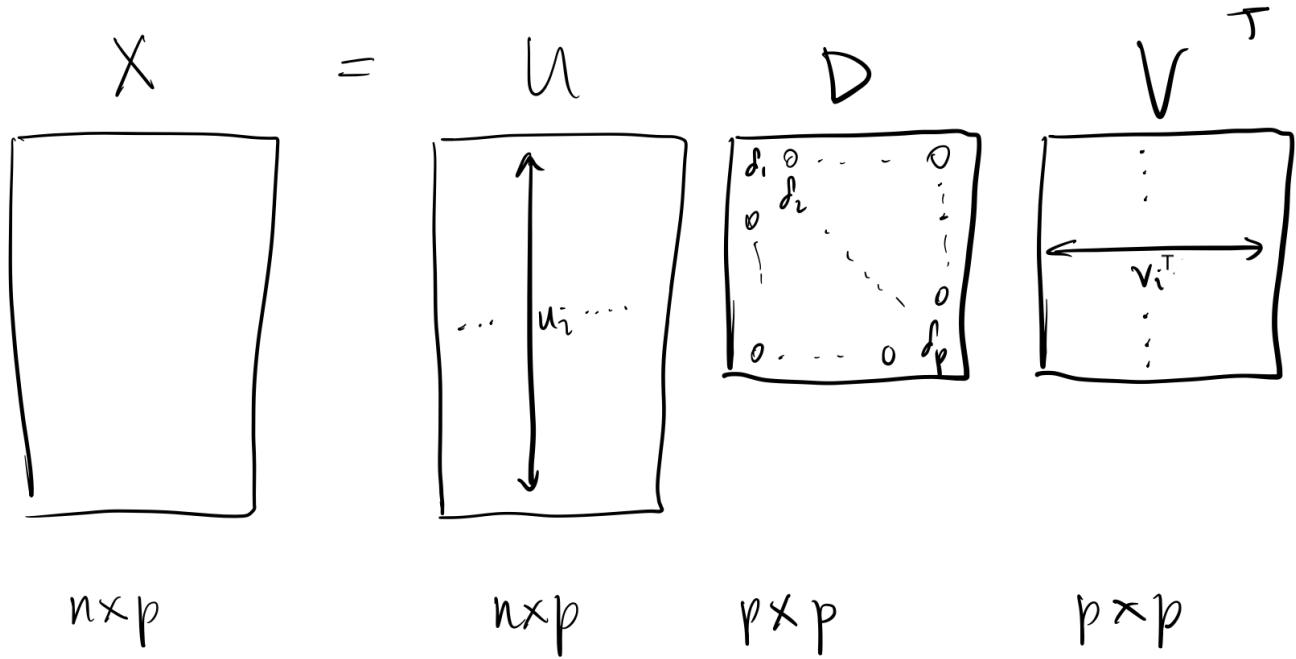


Singular Value Decomposition

Suppose we have a matrix X with n rows & p columns (without loss of generality, assume $n \geq p$). It turns out that every matrix X has a *singular value decomposition*:



where the following statements are true:

1. The columns of U : u_1, \dots, u_p are orthonormal vectors and are called the *left singular vectors* of X .
2. $U^T U = I_p$
3. The rows of V : v_1, \dots, v_p are orthonormal vectors and are called the *right singular vectors* of X .
4. $V^T V = I_p$ (i.e. the rows of V : v_1, \dots, v_p are orthonormal vectors)
5. The diagonal entries of D , $\{\delta_1, \dots, \delta_p\}$, are nonnegative & called the *singular values* of X . Without loss of generality we can order them such that they are always nonincreasing:

$$\delta_1 \geq \delta_2 \geq \dots \geq \delta_p \geq 0.$$

Note: If $n < p$, an SVD still exists, but now U is square and V is not.

Alternative form of the SVD:

If we carry out the matrix multiplication, we can formulate the decomposition as:

$$X = \sum_{i=1}^d \delta_i u_i v_i^T$$

Things to notice:

- We have expressed the design matrix X as a weighted sum of rank 1 matrices: $u_1 v_1^T, \dots, u_p v_p^T$ with weights equal to the singular values, $\{\delta_1, \dots, \delta_p\}$.
- Some of the singular values could be zero! If there are r nonzero singular values, then $\text{rank}(X) = r$.
- Recall that when X is a design matrix, then its rank is the dimension of the space in which variation of its data actually exists! If all the sample data points x_1, \dots, x_n (defined by the rows of the design matrix X) lie on a line, there is only one nonzero singular value. If all the sample data points lie on a plane, there are only two nonzero singular values. In general, if all of the points span a subspace of dimension r , there are r nonzero singular values.

Practical Question:

How do I find out what U, D , & V are?

Consider $X^T X$:

$$X^T X = (UDV^T)^T (UDV^T) = V(D^T D)V^T$$

$$= V \begin{pmatrix} \delta_1^2 & 0 & \cdots & 0 \\ 0 & \delta_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \delta_p^2 \end{pmatrix} V^T$$

Does this structure/factorization look familiar?

Claim: V is the matrix of eigenvectors of $X^T X$ and $\delta_1^2, \dots, \delta_p^2$ are the corresponding eigenvalues.

Proof:

$$\begin{aligned} X^T X v_1 &= V(D^T D)V^T v_1 \\ &= V D^2 \begin{pmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_p^T \end{pmatrix} v_1 \\ &= V D^2 \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} | & | & \cdots & | \\ v_1 & v_2 & \cdots & v_p \\ | & | & \cdots & | \end{pmatrix} \begin{pmatrix} \delta_1^2 & 0 & \cdots & 0 \\ 0 & \delta_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \delta_p^2 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \delta_1^2 v_1 \end{aligned}$$

Same argument holds for v_2, \dots, v_p .

Similar claim:

XX^T has an eigendecomposition UD^2U^T .

Why do we care about the SVD? A few reasons:

1. Analysis using the SVD can lend insights and different perspectives into other ML methods (e.g. regression)
2. SVD can help with dimension reduction
 \Rightarrow has a close tie to PCA (more on this later, but a sneak preview: row i of UD describes the coordinates of the centered sample point x_i^C in principal component space)

Geometric Interpretation of the SVD

Suppose we know that for an arbitrary vector w , we know $Xw = z$; i.e. multiplying w by X maps it to another vector z . Recall that orthonormal matrices represent rotation operations & diagonal matrices correspond to scaling operations. So if we break X down into its SVD components, we can consider the linear transformation induced by applying the matrix X on an arbitrary vector w as three consecutive transformations:

$$Xw = UDV^Tw$$

1. First multiplication: multiplication by V^T rotates w onto a new vector; call this $V^Tw = \tilde{w}$.

$$Xw = U\tilde{w}$$

2. Second multiplication: multiplication by the diagonal matrix D rescales axes by the singular values, but does not change the direction of the vector. Call the transformed vector after this multiplication $D\tilde{w} = \tilde{\tilde{w}}$

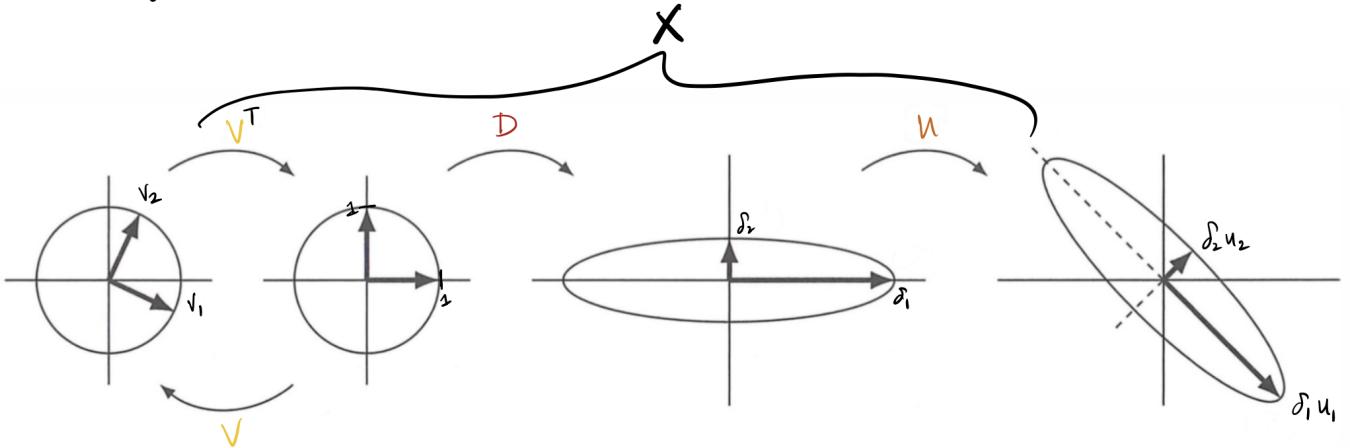
$$Xw = U\tilde{\tilde{w}}$$

- (3) Third multiplication: multiplication by U rotates $\tilde{\tilde{w}}$ again; call this final vector z .

$$Xw = U\tilde{\tilde{w}} = z$$

A nice way of visualizing this is to consider transforming the basis vectors of V , i.e. choose $w = v_i$ for any $i = 1, \dots, p$.

EX:



Intuition Building via SVD

Ex. OLS through SVD:

Consider the OLS setup

$$y = X\beta + \epsilon, \quad \text{epsilon} \sim N(0, \sigma^2 I_n).$$

Derive the OLS estimator $\hat{\beta}$ in terms of the SVD of X . You can assume X has full rank.

Sol:

$$\text{Normal Eq: } (X^T X) \hat{\beta} = X^T y$$

$$\implies \hat{\beta} = (X^T X)^{-1} X^T y$$

$$= (V D^2 V^T)^{-1} (U D V^T)^T y$$

$$= (V D^2 V^T)^{-1} (V D U^T) y$$

$$= V D^{-2} V^T V D U^T y$$

$$= V D^{-2} D U^T y$$

$$= V D^{-1} U^T y$$

How does this relate to multicollinearity & instability in $\hat{\beta}$?

$$\text{Notice: } \hat{\beta} = V D^{-1} U^T y = \sum_{j=1}^p \delta_j^{-1} (u_j^T y) v_j$$

so if δ_j is very close to zero, small changes in δ_j explode and result in very large changes in $\hat{\beta}$.

Question: In regression, how could regularization (e.g. ridge/lasso) stabilize the OLS estimate?

Exercise: For a given penalty parameter λ , calculate the \mathcal{L}^2 -regularized (i.e. ridge) estimate $\hat{\beta}_\lambda$ in terms of the SVD of X .

Dimension Reduction via SVD

Idea: If at some point k , the remaining singular values, $\{\delta_j\}_{j>k}$, are very small in magnitude, their relative contribution to the weighted sum is very small:

$$X = \delta_1 u_1 v_1^T + \cdots + \delta_k u_k v_k^T + \delta_{k+1} u_{k+1} v_{k+1}^T + \cdots + \delta_p u_p v_p^T$$

If they aren't adding much, **who needs them?**

Let's zero them out.

Doing so results in a *low rank/rank-k approximation* of the data matrix X :

$$X \approx \delta_1 u_1 v_1^T + \cdots + \delta_k u_k v_k^T \quad (+0 + \cdots + 0)$$

$$= \underbrace{[u_1 \cdots u_k]}_{n \times k} \underbrace{\begin{pmatrix} \delta_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \delta_k \end{pmatrix}}_{k \times k} \underbrace{\begin{pmatrix} v_1^T \\ \vdots \\ v_k^T \end{pmatrix}}_{k \times p}$$

Call these truncated matrices U_k , D_k , V_k .

Instead of storing $n \times p$ values, now I only need to store $(n \times k) + (k \times k) + (k \times p) = (n + k + p)k$. This can matter a lot for high dimensional data, i.e. when p is very large!!

Example: Preview of Application to Recommender Systems

■ $A = U \Sigma V^T$ - example: Users to Movies

Matrix	Alien	Serenity	Casablanca	Amelie	
SciFi	1	1	1	0	0
	3	3	3	0	0
	4	4	4	0	0
	5	5	5	0	0
	0	2	0	4	4
	0	0	0	5	5
	0	1	0	2	2

$$= \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, <http://www.mmds.org>

$U =$
User-to-concept matrix

$V =$
Movie-to-concept matrix

Reading Assignment:

Chapter 11.3 of *Mining of Massive Datasets*.