SWCON253 Machine Learning

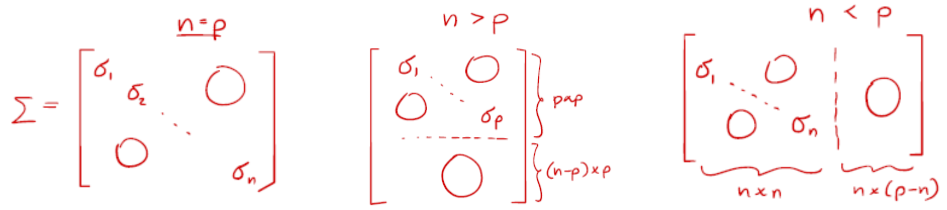
Lecture 8. Singular Value Decomposition in Machine Learning

$X \in \mathbb{R}^{nxp}$ has SVD, $U\Sigma V^T$, where

$U \in \mathbb{R}^{nxn}$ is orthogonal $U^TU = UU^T = I$ (columns of U = left singular vectors)

$V \in \mathbb{R}^{pxp}$ is orthogonal $V^TV = VV^T = I$ (columns of V = right singular vectors)

$\Sigma \in \mathbb{R}^{nxp}$ is diagonal with diagonal elements (singular values), $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(n,p)} \geq 0$



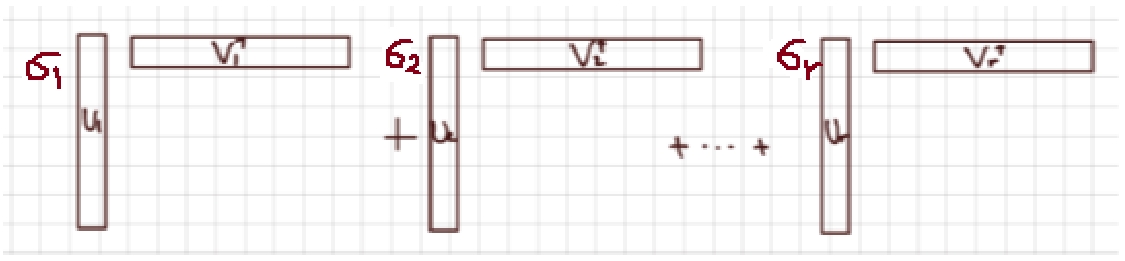
Note that the number of $\sigma_i > 0$ = rank(X) = the number of linearly independent columns

Ex.

$$X = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I \begin{bmatrix} 10 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix} I = U\Sigma V^T$$

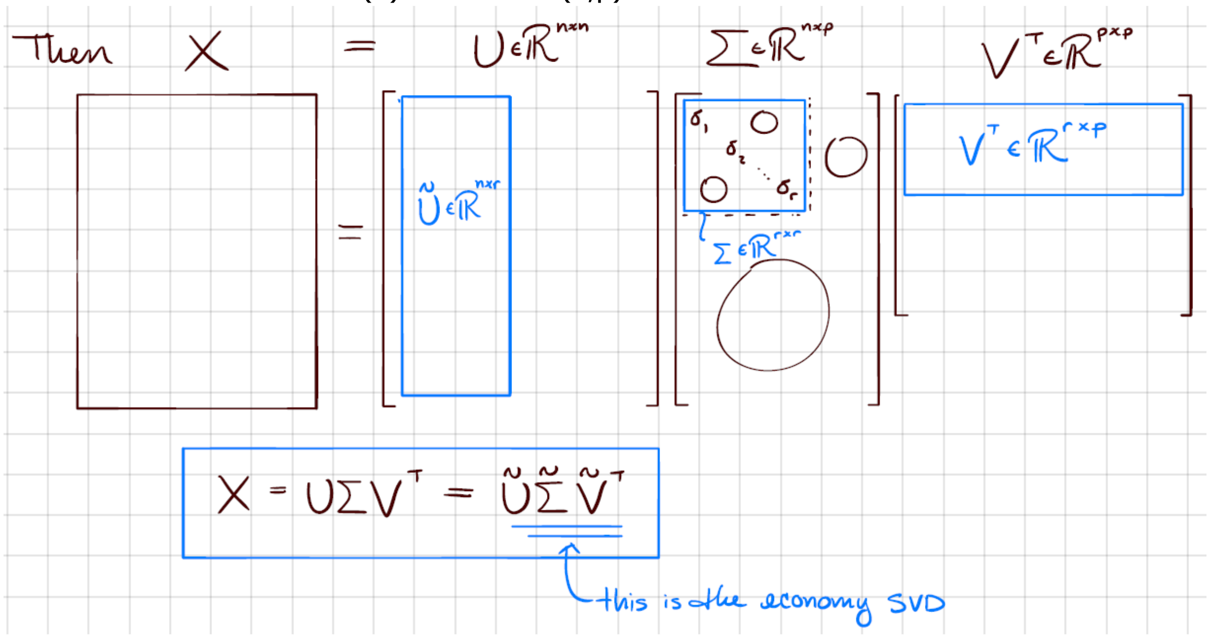
Equivalently, we can write X as sum of rank-1 matrices

$$X = \sum_{i=1}^{r=\min(n,p)} \sigma_i u_i v_i^T$$



The Economy SVD

Assume $X \in \mathbb{R}^{nxp}$ has rank(X) = r << min(n,p)



Then $X = U \in \mathbb{R}^{n\times n}$ $\Sigma \in \mathbb{R}^{n\times p}$ $V^T \in \mathbb{R}^{p\times p}$
$\tilde{U} \in \mathbb{R}^{n\times r}$
σ_1 σ_2 σ_r 0
$\tilde{\Sigma} \in \mathbb{R}^{r\times r}$
$\tilde{V}^T \in \mathbb{R}^{r\times p}$
$X = U\Sigma V^T = \tilde{U}\tilde{\Sigma}\tilde{V}^T$
this is the economy SVD

$\tilde{\Sigma} \in \mathbb{R}^{rxr}$ diagonal with diagonal elements called singular values, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$

$\tilde{U} \in \mathbb{R}^{nxr}$ $\tilde{U}^T\tilde{U} = \mathrm{I}$, but $\tilde{U}\tilde{U}^T \neq \mathrm{I}$
$\tilde{V} \in \mathbb{R}^{pxr}$ $\tilde{V}^T\tilde{V} = \mathrm{I}$, but $\tilde{V}\tilde{V}^T \neq \mathrm{I}$

The columns of $\tilde{U}(\tilde{V})$ are all orthogonal to each other, but the rows of $\tilde{U}(\tilde{V})$ are not orthogonal to each other.

Netflix example
n = number of movies ≈ 5k
p = number of customers ≈ 100 million
storage = 5k x 100 m x 4 bytes = 2 TB
okay to store, difficult to use in learning algorithms

If X is rank-10, then compute
$\tilde{U}$ = 5k (row) x 10 (cols) x 4 bytes

$\tilde{\Sigma}$ = 10 (singular values) x 4 bytes

$\tilde{V}$ = 100 m x 10 x 4 bytes
Storage = 4 GB

Dimensionality Reduction

We have n points $\underline{x}_i \in \mathbb{R}^p$ for i=1,...,n.
Dimensionality reduction means defining new points $\underline{z}_i \in \mathbb{R}^k$ for i=1,…n for k<p that preserve important properties of the $\underline{x}_i$'s.

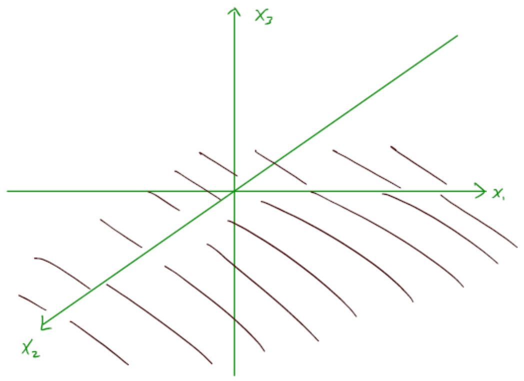
Observe $\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_n \in \mathbb{R}^p$

Goal:
Find $\underline{z}_1, \underline{z}_2, \ldots, \underline{z}_n \in \mathbb{R}^k$ for k < p
k implies some lower dimensional space than p. $\underline{z}_i$ is reduced dimensionality vectors

We want to find for each one of these points we want to map it to smaller or fewer dimensional vectors. So that we can represent essentially the same information but with fewer or smaller vectors.

So that $\underline{z}_i$'s share useful properties with $\underline{x}_i$'s
(e.g., $\|\underline{x}_i - \underline{x}_j\| \approx \|\underline{z}_i - \underline{z}_j\|$)
How? Let's use the SVD.

Ex. n=3, S=$\{\underline{x} \in \mathbb{R}^3\colon\ x_3 = 0\}$
Horizontal plane



Instead of representing each x_i using 3D coordinates, we only need to represent when it is in the x_1-x_2 plane.

Ex

$$X=\begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 0 & 0 \end{bmatrix} \quad \text{e.g., } x_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} \in \mathbb{R}^4$$

We want to find some z_i, reduced dimensional representation of x_i.

$$X=\begin{bmatrix} 1 & 1 \\ -1 & 1 \\ 1 & 1 \\ -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

This is not SVD. Why? These matrices are not orthogonal.
What is rank of X? rank(X)=2

We will compute the SVD using an example:

$$X=\underbrace{\begin{bmatrix} 1/\sqrt{5} & 1/2 \\ -1/\sqrt{5} & 1/2 \\ 1/\sqrt{5} & 1/2 \\ -1/\sqrt{5} & 1/2 \\ 1/\sqrt{5} & 0 \end{bmatrix}}_{\tilde{U}} \underbrace{\begin{bmatrix} \sqrt{5}\sqrt{2} & 0 \\ 0 & 2\sqrt{2} \end{bmatrix}}_{\tilde{\Sigma}} \underbrace{\begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 & 0 \\ 0 & 0 & -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}}_{\tilde{V}^T}$$

As the rank of X is 2, all of the columns of X lie in a 2D subspace and also all of the rows of X lie in a 2D subspace. So, what they suggest is that we should be able to do some dimensionality reduction because everything lies in some sort of a 2D plane and we don't necessarily need to all 4 dimensions to represent where these points are. So we try to accomplish dimensionality reduction using SVD.

Let's consider X^T. We want to make new versions that we want to reduce their dimension.

$$X^T = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ \underline{x_1} & \underline{x_2} & \underline{x_3} & \underline{x_4} & \underline{x_5} \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} = \tilde{V}\tilde{\Sigma}\tilde{U}^T$$

$$= \underbrace{{}^{1}\!/\!_{\sqrt{2}}\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix}}_{\tilde{V}^T} \underbrace{\sqrt{2}\begin{bmatrix} 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}}_{\tilde{\Sigma}\tilde{U}^T}$$

Each of the x_i's that we want to reduce the dimensionality is the rank-4 vectors. Each of them can be written as a weighted sum of the columns of $\tilde{V}$. So they all are in a 2D subspace.

$\tilde{\Sigma}\tilde{U}^T$ tells us where they lie in that subspace.

So we can say that

$\underline{x_i}$ = weighted sum of columns of $\tilde{V}$.
$\underline{z_i}$ = reduced dimensional vectors = weights
For example,
$\underline{z_1} = \sqrt{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$,
$\underline{z_2} = \sqrt{2}\begin{bmatrix} -1 \\ 1 \end{bmatrix}$
…

We started with 4D points x_i and then we used SVD to compute 2D representations that still preserve these distances, e.g., $\|\underline{x_i} - \underline{x_j}\| \approx \|\underline{z_i} - \underline{z_j}\|$.

For example,
$x_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix}$ -> $\underline{z_1} = \sqrt{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$,
We are only using half of many features to represent $\underline{z_i}$

Given $(\underline{x}_i, y_i)$ for i=1, ...,n

Labels $\underline{y} \in \mathbb{R}^p$ for n training samples

Features $X \in \mathbb{R}^{nxp}$ (p features)

$y_i \approx \hat{y}_i = \underline{x}^T \underline{\widehat{w}}$

$y_i \approx \hat{y}_i = X\underline{\widehat{w}}$

If n ≥ p, rank(X) = p (X has p linearly independent columns), then

$\widehat{w} = (X^T X)^{-1} X^T \underline{y}$

X = UΣV^T

$(X^T X)^{-1} X^T$

$= (V\Sigma^T U^T U\Sigma V^T)^{-1} V\Sigma^T U^T$

$U^T U = I$

If A, B are square and invertible, then (AB)$^{-1}$ = B^{-1}A^{-1}

$= (V\Sigma^T \Sigma V^T)^{-1} V\Sigma^T U^T$

$= (V\Sigma^T \Sigma V^T)^{-1} V\Sigma^T U^T$

$= (V^T)^{-1} (V\Sigma^T \Sigma)^{-1} V\Sigma^T U^T$

$= (V^T)^{-1} (V\Sigma^T \Sigma)^{-1} V\Sigma^T U^T$

$= (V^{-1})^{-1} (V\Sigma^T \Sigma)^{-1} V\Sigma^T U^T$

$= (V^{-1})^{-1} (\Sigma^T \Sigma)^{-1} V^{-1} V\Sigma^T U^T$

(V^{T}V = VV^T = I = V^{-1}V -> V^T = V^{-1})

$= V(\Sigma^T \Sigma)^{-1} V^T V\Sigma^T U^T$

$= V(\Sigma^T \Sigma)^{-1} \Sigma^T U^T$

Let's think about $(\Sigma^T \Sigma)^{-1} \Sigma^T$

$n \geq p$

Σ (n x p)

$$\left[\begin{array}{ccc} \sigma_1 & \ddots & 0 \\ 0 & & \sigma_p \\ \hline & 0 & \end{array}\right]$$

$\Sigma^T \Sigma$ (p x p)

$$\begin{bmatrix} \sigma_1^2 & & & 0 \\ & \sigma_2^2 & & \\ & & \ddots & \\ 0 & & & \sigma_p^2 \end{bmatrix}$$

$(\Sigma^T \Sigma)^{-1}$

$$\begin{bmatrix} 1/\sigma_1^2 & & & 0 \\ & 1/\sigma_2^2 & & \\ & & \ddots & \\ 0 & & & 1/\sigma_p^2 \end{bmatrix}$$

$(\Sigma^T \Sigma)^{-1} \Sigma^T$ (p x n)

$$\left[\begin{array}{cccc:c} 1/\sigma_1 & & & 0 & \\ & 1/\sigma_2 & & & 0 \\ & & \ddots & & \\ 0 & & & 1/\sigma_p & \end{array}\right]$$

$= \Sigma^+$ **(called pseudo-inverse)**

The pseudoinverse is the generalization of the matrix inverse for square matrices to rectangular matrices where the number of rows and columns are not equal.

We are interested in $(X^T X)^{-1} X^T = V\Sigma^+ U^T$

We said our model

$\hat{y} = U\Sigma V^T \underline{\hat{w}}$

Our estimator

$\hat{w} = V\Sigma^+ U^T \underline{y}$

$y \approx \hat{y} = U\Sigma V^T \underline{\hat{w}}$

Multiplying both sides by U^T

$U^T y \approx U^T U \Sigma V^T \underline{\hat{w}}$

Multiplying both sides by Σ^+

$\Sigma^+ U^T y \approx \Sigma^+ \Sigma V^T \underline{\hat{w}}$

Multiplying both sides by V

$V\Sigma^+ U^T y \approx V V^T \underline{\hat{w}}$

$\underline{\hat{w}} \approx V\Sigma^+ U^T y$

- Basically what we've shown here is that this **basic least squares model** $(X^T X)^{-1} X^T$ that we've been studying for a while that involve these big matrices and inverses.
- We start thinking about using SVD as an elegant, simple and interpretable form.
- So, ideally what we'd like to do is to just invert this matrix $X=U\Sigma V^T$ (rectangular and generally non-invertible) and so
- what we do instead is we use this SVD where U's and V's are invertible and instead of inverting sigma which is not exactly convertible.
- We use this notion of **pseudo-inverse** and the notion of pseudo-inverse gives us exactly the least squares estimator and
- We like to do a matrix inverse (only the matrix is not a convertible), suddenly becomes very clean and elegant using SVD.