Cross Validated

Relationship between SVD and PCA. How to use SVD to perform PCA?

Asked 9 years, 1 month ago Modified 23 days ago Viewed 486k times

599

Principal component analysis (PCA) is usually explained via an eigen-decomposition of the covariance matrix. However, it can also be performed via singular value decomposition (SVD) of the data matrix $\mathbf X$. How does it work? What is the connection between these two approaches? What is the relationship between SVD and PCA?

Or in other words, how to use SVD of the data matrix to perform dimensionality reduction?

pca dimensionality-reduction matrix svd faq

Share Cite Improve this question Follow

edited Apr 5, 2021 at 17:18 kjetil b halvorsen ♦ 76.8k 31 187 579

asked Jan 20, 2015 at 23:47 amoeba 104k 35 311 338

14 I wrote this FAQ-style question together with my own answer, because it is frequently being asked in various forms, but there is no canonical thread and so closing duplicates is difficult. Please provide meta comments in this accompanying meta thread. – amoeba Jan 22, 2015 at 11:25

2 stats.stackexchange.com/questions/177102/... – kjetil b halvorsen ♦ Feb 3, 2016 at 10:45

2 In addition to an excellent and detailed amoeba's answer with its further links I might recommend

to check this, where PCA is considered side by side some other SVD-based techniques. The discussion there presents algebra almost identical to amoeba's with just minor difference that the speech there, in describing PCA, goes about svd decomposition of $\mathbf{X}/\sqrt{n}$ [or $\mathbf{X}/\sqrt{n-1}$] instead of $\mathbf{X}$ - which is simply convenient as it relates to the PCA done via the eigendecomposition of the covariance matrix. – ttnphns Feb 3, 2016 at 12:18

PCA is a special case of SVD. PCA needs the data normalized, ideally same unit. The matrix is nxn in PCA. – Orvar Korvar Oct 17, 2017 at 9:12

1 @OrvarKorvar: What n x n matrix are you talking about ? – Cbhihe Mar 29, 2018 at 15:16

5 Answers

Sorted by: Highest score (default)

731

+200

Let the real values data matrix $\mathbf{X}$ be of $n \times p$ size, where n is the number of samples and p is the number of variables. Let us assume that it is *centered*, i.e. column means have been subtracted and are now equal to zero.

Then the $p \times p$ covariance matrix $\mathbf{C}$ is given by $\mathbf{C} = \mathbf{X}^\top\mathbf{X}/(n-1)$. It is a symmetric matrix and so it can be diagonalized:

$$\mathbf{C} = \mathbf{V}\mathbf{L}\mathbf{V}^\top,$$

where $\mathbf{V}$ is a matrix of eigenvectors (each column is an eigenvector) and $\mathbf{L}$ is a diagonal matrix with eigenvalues λ_i in the decreasing order on the diagonal. The eigenvectors are called *principal axes* or *principal directions* of the data. Projections of the data on the principal axes are called *principal components*, also known as *PC scores*; these can be seen as new, transformed, variables. The j-th principal component is given by j-th column of $\mathbf{XV}$. The coordinates of the i-th data point in the new PC space are given by the i-th row of $\mathbf{XV}$.

If we now perform singular value decomposition of $\mathbf{X}$, we obtain a decomposition

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^\top,$$

where $\mathbf{U}$ is a unitary matrix (with columns called left singular vectors), $\mathbf{S}$ is the diagonal matrix of singular values s_i and $\mathbf{V}$ columns are called right singular vectors. From here one can easily see that

$$\mathbf{C} = \mathbf{V}\mathbf{S}\mathbf{U}^\top\mathbf{U}\mathbf{S}\mathbf{V}^\top/(n-1) = \mathbf{V}\frac{\mathbf{S}^2}{n-1}\mathbf{V}^\top,$$

meaning that right singular vectors $\mathbf{V}$ are principal directions (eigenvectors) and that singular values are related to the eigenvalues of covariance matrix via $\lambda_i = s_i^2/(n-1)$. Principal components are given by $\mathbf{XV} = \mathbf{U}\mathbf{S}\mathbf{V}^\top\mathbf{V} = \mathbf{US}$.

To summarize:

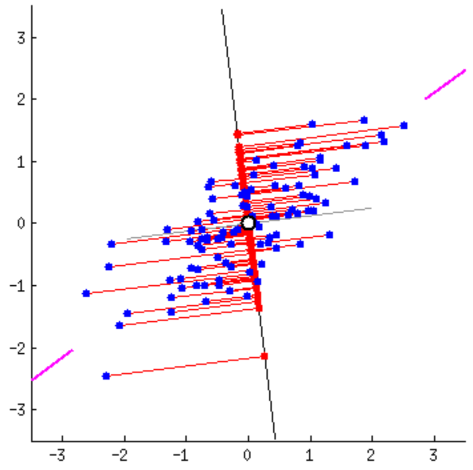
1. If $\mathbf X = \mathbf{USV}^\top$, then the columns of $\mathbf V$ are principal directions/axes (eigenvectors).
2. Columns of $\mathbf{US}$ are principal components ("scores").
3. Singular values are related to the eigenvalues of covariance matrix via $\lambda_i = s_i^2/(n-1)$. Eigenvalues λ_i show variances of the respective PCs.
4. Standardized scores are given by columns of $\sqrt{n-1}\mathbf U$ and loadings are given by columns of $\mathbf{VS}/\sqrt{n-1}$. See e.g. here and here for why "loadings" should not be confused with principal directions.
5. **The above is correct only if $\mathbf X$ is centered.** Only then is covariance matrix equal to $\mathbf X^\top \mathbf X/(n-1)$.
6. The above is correct only for $\mathbf X$ having samples in rows and variables in columns. If variables are in rows and samples in columns, then $\mathbf U$ and $\mathbf V$ exchange interpretations.
7. If one wants to perform PCA on a correlation matrix (instead of a covariance matrix), then columns of $\mathbf X$ should not only be centered, but standardized as well, i.e. divided by their standard deviations.
8. To reduce the dimensionality of the data from p to $k<p$, select k first columns of $\mathbf U$, and $k\times k$ upper-left part of $\mathbf S$. Their product $\mathbf U_k \mathbf S_k$ is the required $n \times k$ matrix containing first k PCs.
9. Further multiplying the first k PCs by the corresponding principal axes $\mathbf V_k^\top$ yields $\mathbf X_k = \mathbf U_k \mathbf S_k \mathbf V_k^\top$ matrix that has the original $n \times p$ size but is *of lower rank* (of rank k). This matrix $\mathbf X_k$ provides a *reconstruction* of the original data from the first k PCs. It has the lowest possible reconstruction error, see my answer here.
10. Strictly speaking, $\mathbf U$ is of $n\times n$ size and $\mathbf V$ is of $p \times p$ size. However, if $n>p$ then the last $n-p$ columns of $\mathbf U$ are arbitrary (and corresponding rows of $\mathbf S$ are constant zero); one should therefore use an *economy size* (or *thin*) SVD that returns $\mathbf U$ of $n\times p$ size, dropping the useless columns. For large $n\gg p$ the matrix $\mathbf U$ would otherwise be unnecessarily huge. The same applies for an opposite situation of $n\ll p$.

Further links

- What is the intuitive relationship between SVD and PCA -- a very popular and very similar thread on math.SE.
- Why PCA of data by means of SVD of the data? -- a discussion of what are the benefits of performing PCA via SVD [short answer: numerical stability].
- PCA and Correspondence analysis in their relation to Biplot -- PCA in the context of

some congeneric techniques, all based on SVD.

- Is there any advantage of SVD over PCA? -- a question asking if there any benefits in using SVD *instead* of PCA [short answer: ill-posed question].
- Making sense of principal component analysis, eigenvectors & eigenvalues -- my answer giving a non-technical explanation of PCA. To draw attention, I reproduce one figure here:



Share Cite Improve this answer Follow

edited Feb 22, 2023 at 16:25
VMMF
115 5

answered Jan 20, 2015 at 23:47
amoeba
104k 35 311 338

4 +1 for both Q&A. Thanks for sharing. I have one question: why do you have to assume that the data matrix is centered initially? – Antoine Aug 6, 2015 at 8:39

20 @Antoine, covariance matrix is by definition equal to $\langle(\mathbf{x}_i-\bar{\mathbf{x}})(\mathbf{x}_i-\bar{\mathbf{x}})^\top\rangle$, where angle brackets denote average value. If all $\mathbf{x}_i$ are stacked as rows in one matrix $\mathbf{X}$, then this expression is equal to $(\mathbf{X}-\bar{\mathbf{X}})(\mathbf{X}-\bar{\mathbf{X}})^\top/(n-1)$. If $\mathbf{X}$ is centered then it simplifies to $\mathbf{X}\mathbf{X}^\top/(n-1)$. Think of variance; it's equal to $\langle(x_i-\bar{x})^2\rangle$. But if $\bar{x}=0$ (i.e. data are centered), then it's simply the average value of x_i^2. – amoeba Aug 6, 2015 at 9:43

2 @amoeba for those less familiar with linear algebra and matrix operations, it might be nice to mention that $(A.B.C)^T = C^T.B^T.A^T$ and that $U^T.U = Id$ because U is orthogonal – Antoine Feb 3, 2016 at 15:20

2 A code sample for PCA by SVD: stackoverflow.com/questions/3181593/... – optimist Mar 23, 2016 at 11:51

2 @amoeba yes, but why use it? Also, is it possible to use the same denominator for S? The problem is that I see formulas where $\lambda_i = s_i^2$ and try to understand, how to use them? – Dims Sep 5, 2017 at 22:49

I wrote a Python & Numpy snippet that accompanies @amoeba's answer and I leave it here

41

I wrote a Python & Numpy snippet that accompanies @amoeba's answer and I leave it here in case it is useful for someone. The comments are mostly taken from @amoeba's answer.

```python
import numpy as np
from numpy import linalg as la
np.random.seed(42)


def flip_signs(A, B):
    """
    utility function for resolving the sign ambiguity in SVD
    http://stats.stackexchange.com/q/34396/115202
    """
    signs = np.sign(A) * np.sign(B)
    return A, B * signs


# Let the data matrix X be of n x p size,
# where n is the number of samples and p is the number of variables
n, p = 5, 3
X = np.random.rand(n, p)
# Let us assume that it is centered
X -= np.mean(X, axis=0)

# the p x p covariance matrix
C = np.cov(X, rowvar=False)
print("C = \n", C)
# C is a symmetric matrix and so it can be diagonalized:
l, principal_axes = la.eig(C)
# sort results wrt. eigenvalues
idx = l.argsort()[::-1]
l, principal_axes = l[idx], principal_axes[:, idx]
# the eigenvalues in decreasing order
print("l = \n", l)
# a matrix of eigenvectors (each column is an eigenvector)
print("V = \n", principal_axes)
# projections of X on the principal axes are called principal components
principal_components = X.dot(principal_axes)
print("Y = \n", principal_components)

# we now perform singular value decomposition of X
# "economy size" (or "thin") SVD
U, s, Vt = la.svd(X, full_matrices=False)
V = Vt.T
S = np.diag(s)

# 1) then columns of V are principal directions/axes.
assert np.allclose(*flip_signs(V, principal_axes))

# 2) columns of US are principal components
assert np.allclose(*flip_signs(U.dot(S), principal_components))

# 3) singular values are related to the eigenvalues of covariance matrix
assert np.allclose((s ** 2) / (n - 1), l)

# 8) dimensionality reduction
k = 2
PC_k = principal_components[:, 0:k]
US_k = U[:, 0:k].dot(S[0:k, 0:k])
assert np.allclose(*flip_signs(PC_k, US_k))
```

```
# 10) we used "economy size" (or "thin") SVD
assert U.shape == (n, p)
assert S.shape == (p, p)
assert V.shape == (p, p)
```

Share Cite Improve this answer Follow

edited Oct 9, 2022 at 16:45

community wiki
3 revs, 3 users 67%
turdus-merula

FWIW, here's a Jupyter version: davidvandebunte.gitlab.io/executable-notes/notes/se/... – davidvandebunte Jun 20, 2022 at 18:57

This errors when $p > n$. – Physmatik Dec 3, 2023 at 4:03

33

Let me start with PCA. Suppose that you have n data points comprised of d numbers (or dimensions) each. If you center this data (subtract the mean data point μ from each data vector x_i) you can stack the data to make a matrix

$$X = \left(\begin{array}{c} x_1^T - \mu^T \\ \hline x_2^T - \mu^T \\ \hline \vdots \\ \hline x_n^T - \mu^T \end{array} \right).$$

The covariance matrix

$$S = \frac{1}{n-1} \sum_{i=1}^n (x_i - \mu)(x_i - \mu)^T = \frac{1}{n-1} X^T X$$

measures to which degree the different coordinates in which your data is given vary together. So, it's maybe not surprising that PCA -- which is designed to capture the variation of your data -- can be given in terms of the covariance matrix. In particular, the eigenvalue decomposition of S turns out to be

$$S = V \Lambda V^T = \sum_{i=1}^r \lambda_i v_i v_i^T \,,$$

where v_i is the i-th *Principal Component*, or PC, and λ_i is the i-th eigenvalue of S and is also equal to the variance of the data along the i-th PC. This decomposition comes from a general theorem in linear algebra, and some work *does* have to be done to motivate the relation to PCA.

SVD is a general way to understand a matrix in terms of its column-space and row-space. (It's a way to rewrite any matrix in terms of other matrices with an intuitive relation to the row and column space.) For example, for the matrix $A = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$ we can find directions u_i and v_i in the domain and range so that

You can find these by considering how A as a linear transformation morphs a unit sphere $\mathbb{S}$ in its domain to an ellipse: the principal semi-axes of the ellipse align with the u_i and the v_i are their preimages.

In any case, for the data matrix X above (really, just set $A = X$), SVD lets us write

$$X = \sum_{i=1}^{r} \sigma_i u_i v_i^T \,,$$

where $\{u_i\}$ and $\{v_i\}$ are orthonormal sets of vectors. A comparison with the eigenvalue decomposition of S reveals that the "right singular vectors" v_i are equal to the PCs, the "right singular vectors" are

$$u_i = \frac{1}{\sqrt{(n-1)\lambda_i}} X v_i \,,$$

and the "singular values" σ_i are related to the data matrix via

$$\sigma_i^2 = (n-1)\lambda_i \,.$$

It's a general fact that the left singular vectors u_i span the column space of X. In this specific case, u_i gives us a scaled projection of the data X onto the direction of the i-th principal component. The right singular vectors v_i in general span the row space of X, which gives us a set of orthonormal vectors that spans the data much like PCs.

I go into some more details and benefits of the relationship between PCA and SVD in this longer article.

Share Cite Improve this answer Follow

edited Mar 11, 2023 at 11:53
Community Bot
1

answered Aug 23, 2017 at 13:07
Andre P
511 5 3

Thanks for your anser Andre. Just two small typos correction: 1. In the last paragraph you`re confusing left and right. 2. In the (capital) formula for X, you're using v_j instead of v_i. – Alon Sep

3, 2019 at 13:09

2

Let's try to understand using a data matrix X of dimension $n \times d$, where $d \gg n$ and $rank(X) = n$.

Then $\underset{n\times d}{X} = \underset{n\times n}{U}\underset{n\times n}{\Sigma}\underset{n\times d}{V^T}$ (reduced SVD)

with $X^TX = V\Sigma^TU^TU\Sigma V^T = V\Sigma^2V^T$ (since unitary / orthonormal U, V and diagonal Σ)

and the covariance matrix (assuming X is already mean centered, i.e., the columns of X have 0 means)

$C = E[X^TX] - E[X]^TE[X] = \frac{X^TX}{n-1} - 0 = V\frac{\Sigma^2}{n-1}V^T = \tilde{V}\Lambda\tilde{V}^T$ (PCA, by spectral decomposition)

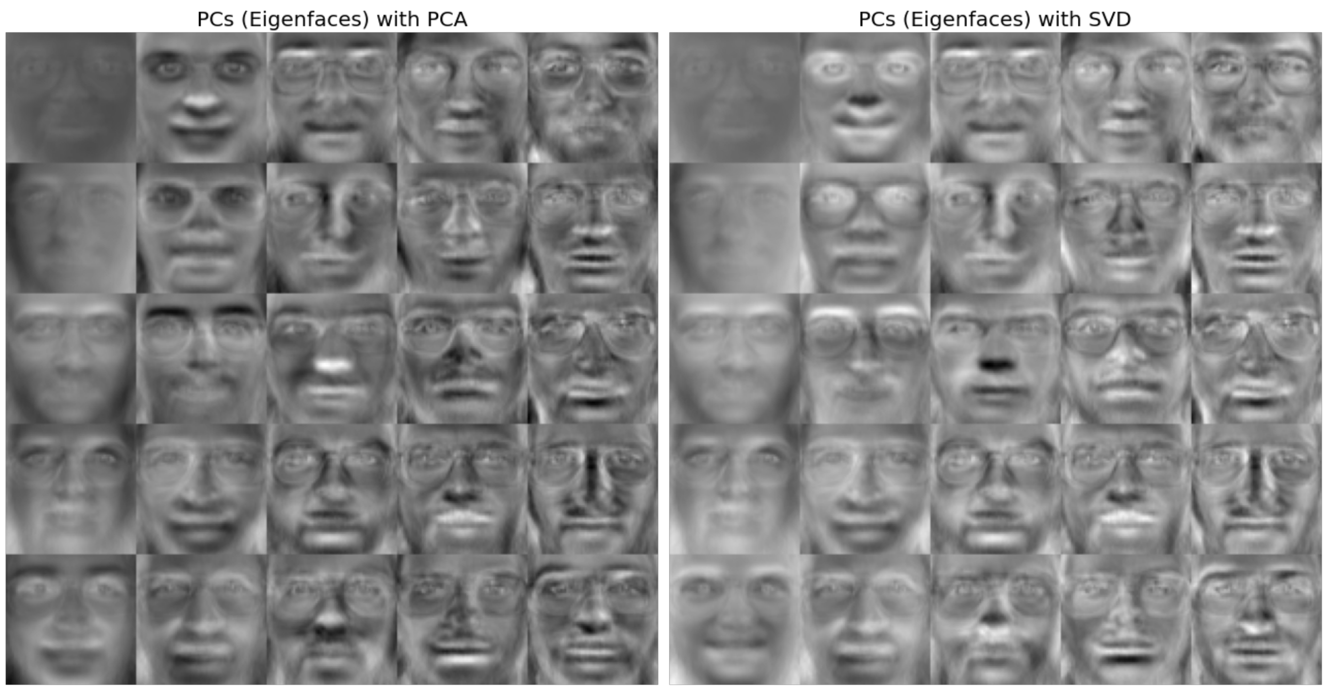
$\implies \Lambda = \frac{\Sigma^2}{n-1}$ and $V = \tilde{V}$ upto sign flip.

Let's validate the above with `eigenfaces` (i.e., the principal components / eigenvectors of the covariance matrix for such a face dataset) using the following face dataset:

```
import numpy as np
from sklearn.decomposition import PCA
from sklearn.datasets import fetch_olivetti_faces

X = fetch_olivetti_faces().data
X.shape # 400 face images of size 64×64 flattened
# (400,4096)
n = len(X)

# z-score normalize
X = X - np.mean(X, axis=0) # mean-centering
# X = X / np.std(X, axis=0)  # scaling to have sd=1

# choose first k eigenvalues / eigenvectors for dimensionality reduction
k = 25

# SVD
U, Σ, Vt = np.linalg.svd(X, full_matrices=False)

# PCA
pca = PCA(k).fit(X)
PC = pca.components_.T
#Vt.shape, PC.shape
```

Now let's compare the eigenvalues and eigenvectors computed:

```
# first k eigenvalues of Λ = Σ^2/(n-1) # from SVD
print(Σ[:k]**2/(n-1))              # from SVD
# [18.840178   11.071763    6.304614    3.9545844   2.8560426   2.49771
# 1.9200633   1.611159    1.5492224   1.3229507   1.2621089   1.1369102
# 0.98639774  0.90758985  0.84092826  0.77355367  0.7271429   0.64526594
# 0.59645116  0.5910001   0.55270135  0.48628208  0.4619924   0.45075357
# 0.4321357 ]
print(pca.explained_variance_[:k]) # from PCA
# [18.840164   11.07176     6.3046117   3.9545813   2.8560433   2.4977121
# 1.9200654   1.6111585   1.549223    1.3229507   1.2621082   1.1369106
# 0.98639697  0.9075892   0.84092826  0.773553    0.72714305  0.64526534
# 0.59645087  0.5909973   0.55269724  0.4862703   0.461944    0.45075053
0.43211046]

# plot PC the k dominant principal components / eigenvectors as images
# 1. using PC obtained with PCA
# 2. using Vt[:k,:].T obtained from SVD
```



Here the differences in the eigenvectors are due to sign ambiguity (refer to https://www.osti.gov/servlets/purl/920802)

Share Cite Improve this answer Follow

edited Apr 13, 2023 at 4:18

answered Apr 12, 2023 at 12:53

Sandipan Dey
739 3 12

0

[This is just to mention how matrix U in the SVD of a data matrix X (also determined by eigenvectors of XX^T) gives the principal directions in PCA.]

Consider data $x_1, \ldots, x_n \in \mathbb{R}^d$.

Goal is to find, for every k, an affine subspace $\mathscr{A}_k$ of dimension k minimising the mean squared loss/error

$$\frac{1}{n}\sum_{i=1}^{n} \operatorname{dist}(x_i, \mathscr{A}_k)^2.$$

Best 0 dimensional fit:

We want a $p \in \mathbb{R}^d$ such that the loss $\frac{1}{n}\sum_{i=1}^{n}\|x_i - p\|^2$ is minimised.
Note loss

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}\|x_i - p\|^2 &= \frac{1}{n}\sum_{i=1}^{n}\|(x_i - q) - (p - q)\|^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}\|x_i - q\|^2 - \frac{1}{n}\sum_{i=1}^{n} 2(p-q)^T(x_i - q) + \|p - q\|^2.\end{aligned}$$

Setting $q = \overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ gives the inequality

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}\|x_i - p\|^2 &= \frac{1}{n}\sum_{i=1}^{n}\|x_i - \overline{x}\|^2 + \|p - \overline{x}\|^2 \\ &\geq \frac{1}{n}\sum_{i=1}^{n}\|x_i - \overline{x}\|^2,\end{aligned}$$

with equality holding when $p = \overline{x}$.
Hence the loss $\frac{1}{n}\sum_{i=1}^{n}\|x_i - p\|^2$ is minimised when $p = \overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$.

Best k dimensional fit:

We want orthonormal vectors $[\hat{w}_1, \ldots, \hat{w}_k]$ in $\mathbb{R}^d$ so that $\overline{x} + \operatorname{span}(\hat{w}_1, \ldots, \hat{w}_k)$ is minimiser of the loss

$$\frac{1}{n}\sum_{i=1}^{n} \operatorname{dist}(x_i, \mathscr{A}_k)^2$$

taken over all k dimensional affine subspaces $\mathscr{A}_k = \overline{x} + \operatorname{span}(w_1, \ldots, w_k)$.

Firstly, for any orthonormal vectors $W = [w_1, \ldots, w_k]$ in $\mathbb{R}^d$ we have

$$\begin{aligned}
&\operatorname{dist}(x_i, \overline{x} + \operatorname{span}(w_1, \dots, w_k))^2 \\
&= \operatorname{dist}(x_i - \overline{x}, \operatorname{span}(w_1, \dots, w_k))^2 \\
&= \|x_i - \overline{x}\|^2 - \sum_{j=1}^{k} (w_j^T (x_i - \overline{x}))^2 \\
&= \|x_i - \overline{x}\|^2 - \sum_{j=1}^{k} w_j^T \left((x_i - \overline{x})(x_i - \overline{x})^T\right) w_j
\end{aligned}$$

and hence the loss

$$\begin{aligned}
&\frac{1}{n} \sum_{i=1}^{n} \operatorname{dist}(x_i, \overline{x} + \operatorname{span}(w_1, \dots, w_k))^2 \\
&= \frac{1}{n} \sum_{i=1}^{n} \|x_i - \overline{x}\|^2 - \sum_{j=1}^{k} w_j^T \left(\frac{1}{n} \sum_{i=1}^{n} (x_i - \overline{x})(x_i - \overline{x})^T \right) w_j .
\end{aligned}$$

Writing $X = [x_1 - \overline{x}, \dots, x_n - \overline{x}] \in \mathbb{R}^{d \times n}$ we have the matrix

$$\frac{1}{n} \sum_{i=1}^{n} (x_i - \overline{x})(x_i - \overline{x})^T = \frac{1}{n} X X^T .$$

The problem of minimising the loss is reduced to the problem of finding orthonormal $[w_1, \dots, w_k] \in \mathbb{R}^{d \times k}$ which maximise

$$\sum_{j=1}^{k} w_j^T \left(\frac{1}{n} X X^T \right) w_j .$$

Writing the SVD of X as $X = U \Sigma V^T$, we have

$$\begin{aligned}
\frac{1}{n} X X^T &= \frac{1}{n} U \Sigma \Sigma^T U^T \\
&= \frac{1}{n} U \underbrace{\begin{pmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_r^2 \\ & & \end{pmatrix}}_{d \times d} U^T \\
&= \frac{1}{n} U \Sigma_d^2 U^T .
\end{aligned}$$

Now for any orthonormal $[w_1, \dots, w_k]$, we have

$$\begin{aligned}
&\sum_{j=1}^{k} w_j^T \left(\frac{1}{n} X X^T\right) w_j \\
&= \frac{1}{n} \sum_{j=1}^{k} w_j^T U \Sigma_d^2 U^T w_j \\
&= \frac{1}{n} \sum_{j=1}^{k} \tilde{w}_j^T \Sigma_d^2 \tilde{w}_j
\end{aligned}$$

where orthonormal set $[\tilde{w}_1, \dots, \tilde{w}_k] = U^T [w_1, \dots, w_k]$.

For $k \le r$ we have

$$\frac{1}{n} \sum_{j=1}^{k} \tilde{w}_j^T \Sigma_d^2 \tilde{w}_j \le \frac{1}{n}(\sigma_1^2 + \dots + \sigma_k^2),$$

with equality attained for eg when

$$[\tilde{w}_1, \dots, \tilde{w}_k] = [e_1, \dots, e_k]$$

that is $U^T [w_1, \dots, w_k] = [e_1, \dots, e_k]$. So the required orthonormal set is

$$[w_1, \dots, w_k] = [U_1, \dots, U_k] \text{ for } k \le r.$$

Finally, for $k \le r$, the set $\overline{x} + \operatorname{span}(U_1, \dots, U_k)$ is the best fit k dimensional affine subspace for the data.

> The corresponding minimum loss is
>
> $$\begin{aligned}
> &\frac{1}{n} \sum_{i=1}^{n} \operatorname{dist}(x_i, \overline{x} + \operatorname{span}(U_1, \dots, U_k))^2 \\
> &= \frac{1}{n} \sum_{i=1}^{n} \|x_i - \overline{x}\|^2 - \frac{1}{n}(\sigma_1^2 + \dots + \sigma_k^2) \\
> &= \frac{1}{n} \operatorname{tr}(X^T X) - \frac{1}{n}(\sigma_1^2 + \dots + \sigma_k^2) \\
> &= \frac{1}{n}(\sigma_{k+1}^2 + \dots + \sigma_r^2).
> \end{aligned}$$

One can also consider the best rank k approximation of

$X = [x_1 - \overline{x}, \dots, x_n - \overline{x}] \in \mathbb{R}^{d \times n}$ to get the same result.

Share Cite Improve this answer Follow

edited Feb 5 at 11:12

answered Dec 28, 2023 at 5:08

Venkata Karthik Bandaru

121 4