

Machine Learning

Textbook: Machine Learning Refined, by Watt, Borhani, and Katsaggelos, 2nd edition:
MLRefined.com

Previous Lecture

- ◆ PCA chooses projection directions from centered data rather than specifying them in advance.
- ◆ The first principal component maximizes projected variance $\mathbf{u}^T C \mathbf{u}$ over unit vectors.
- ◆ Lagrange multipliers show that principal directions are eigenvectors of the covariance matrix.
- ◆ The eigenvalue associated with a principal direction is the variance captured along that direction.
- ◆ The first d principal components are the eigenvectors associated with the d largest covariance eigenvalues.
- ◆ Mean squared reconstruction error after retaining d components equals the sum of the omitted eigenvalues.



§8.3 - PCA, SVD, and the Linear Autoencoder

Last time:

PCA directions were eigenvectors
of the covariance matrix

$$C = \frac{1}{P} X_c^T X_c.$$

Today:

derive the same directions
directly from the singular value decomposition of
 X_c .

Then reinterpret PCA as
an encoder–decoder model.

Centered Data Matrix

Let

$$X_c \in \mathbb{R}^{P \times N}$$

contain centered observations in its rows.

Thus

$$X_c = \begin{bmatrix} \text{---} \mathbf{q}_1^T \text{---} \\ \vdots \\ \text{---} \mathbf{q}_P^T \text{---} \end{bmatrix}.$$

Recall:

PCA covariance matrix:

$$C = \frac{1}{p} X_c^T X_c.$$

Principal directions:
eigenvectors of C .

There is another route
that begins with X_c itself.

Singular Value Decomposition

Let
 $r = \text{rank}(X_c).$

A compact singular value decomposition is
 $X_c = L \Sigma V^T,$

where
 $L \in \mathbb{R}^{P \times r},$
 $V \in \mathbb{R}^{N \times r},$

and
 $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_r).$

The singular values satisfy
 $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0.$

Columns of L are orthonormal.

Columns of V are orthonormal.

What the SVD Says Geometrically

The decomposition
 $X_c = L \Sigma V^T$
separates three ingredients.

V :
directions in feature space.

Σ :
strengths of those directions.

L :
observation-side coordinates after scaling.

The columns of
 V
are called
right singular vectors.

Those right singular vectors
will turn out to be exactly the PCA directions.

Insert the SVD Into the Covariance Matrix

Start with

$$C = \frac{1}{P} X_c^T X_c.$$

Using

$$\begin{aligned} X_c &= L \Sigma V^T: \\ X_c^T X_c &= (V \Sigma L^T)(L \Sigma V^T). \end{aligned}$$

Since

$$\begin{aligned} L^T L &= I: \\ X_c^T X_c &= V \Sigma^2 V^T. \end{aligned}$$

Therefore

$$C = V \left(\frac{\Sigma^2}{P} \right) V^T.$$

This is already an eigendecomposition of the covariance matrix.

PCA Eigenvalues From Singular Values

The columns

$$\mathbf{v}_j$$

of

$$V$$

satisfy

$$\begin{aligned} C \mathbf{v}_j &= \frac{\sigma_j^2}{P} \mathbf{v}_j. \end{aligned}$$

Thus

$$\lambda_j = \frac{\sigma_j^2}{P}.$$

And

PCA direction j = right singular vector $\mathbf{v}_j$.

Covariance eigenvectors
and right singular vectors
are two descriptions of the same directions.

Why the Ordering Matches

Singular values are ordered:

$$\sigma_1 \geq \sigma_2 \geq \dots.$$

Squaring preserves this ordering:

$$\sigma_1^2 \geq \sigma_2^2 \geq \dots.$$

Therefore

$$\lambda_1 = \sigma_1^2/P$$

is the largest covariance eigenvalue.

So:

largest singular value

$$\Leftrightarrow$$

first principal component.

Second largest singular value

$$\Leftrightarrow$$

second principal component.

And so on.

Principal Coordinates From the SVD

PCA coordinates are

$$Z = X_c V.$$

But

$$\begin{aligned} X_c V \\ = L \Sigma V^T V. \end{aligned}$$

Since

$$\begin{aligned} V^T V &= I; \\ Z &= L \Sigma. \end{aligned}$$

Thus:

the principal-component score matrix
is the left singular-vector matrix
scaled by the singular values.

Column j of Z is

$$\sigma_j$$

times column j of L .

Retain Only the First d Components

Let

$$V_d = [\mathbf{v}_1, \dots, \mathbf{v}_d].$$

Low-dimensional coordinates:

$$Z_d = X_c V_d.$$

Reconstruct:

$$\hat{X}_c = Z_d V_d^T = X_c V_d V_d^T.$$

This is exactly the PCA projection formula

from Lesson 1.23.

Now rewrite the same reconstruction using the SVD.

Truncated SVD

Partition the SVD:

$$\begin{aligned}L_d &= [\mathbf{l}_1, \dots, \mathbf{l}_d], \\ \Sigma_d &= \text{diag}(\sigma_1, \dots, \sigma_d), \\ V_d &= [\mathbf{v}_1, \dots, \mathbf{v}_d].\end{aligned}$$

Then

$$\hat{X}_c = L_d \Sigma_d V_d^T.$$

This is the

rank- d truncated SVD

of

$$X_c.$$

So:

PCA reconstruction = truncated SVD reconstruction.

Why the Rank Is at Most d

The matrix

$$L_d \Sigma_d V_d^T$$

has at most

$$d$$

linearly independent columns and rows.

Therefore

$$\text{rank}(\hat{X}_c) \leq d.$$

Dimension reduction is therefore

a low-rank approximation problem.

Each observation may originally have

$$N$$

features,

but the reconstructed data matrix

lives in a d -dimensional feature subspace.

Best Rank- d Approximation

A central theorem of matrix approximation states:

among all matrices

$$A$$

with

$$\text{rank}(A) \leq d,$$

the matrix

$$X_{c,d} = L_d \Sigma_d V_d^T$$

minimizes

$$\|X_c - A\|_F.$$

This is the
Eckart–Young theorem.

Thus PCA is not merely a plausible compression.

It gives the optimal rank- d approximation
under squared Frobenius error.

Reconstruction Error From Singular Values

For the truncated SVD:

$$\|X_c - X_{c,d}\|_F^2 = \sum_{j=d+1}^r \sigma_j^2.$$

Divide by

$$P:$$
$$\frac{1}{P} \|X_c - X_{c,d}\|_F^2$$
$$= \sum_{j=d+1}^r \frac{\sigma_j^2}{P}$$
$$= \sum_{j=d+1}^r \lambda_j.$$

Exactly the PCA reconstruction-error formula
from Lesson 1.23.

Two derivations.

Same quantity.

Explained Variance From Singular Values

Since

$$\lambda_j = \sigma_j^2/P,$$

the explained-variance ratio can be written

$$\frac{\lambda_j}{\sum_r \lambda_r} = \frac{\sigma_j^2}{\sum_r \sigma_r^2}.$$

The factor
 $1/P$
cancels.

So:
PCA explained variance can be computed
directly from squared singular values.

No covariance matrix is required.

Return to the Four-Point Dataset

Our centered data matrix was

$$X_c = \begin{bmatrix} -1.5 & -1 \\ -0.5 & 0 \\ 0.5 & 0 \\ 1.5 & 1 \end{bmatrix}.$$

From Lesson 1.23:

$$\lambda_1 = \frac{7+3\sqrt{5}}{8},$$

and

$$\lambda_2 = \frac{7-3\sqrt{5}}{8}.$$

Since

$$P = 4:$$

$$\sigma_j = \sqrt{4\lambda_j}.$$

Exact Singular Values for the Example

Compute:

$$\begin{aligned} \sigma_1^2 &= 4\lambda_1 \\ &= \frac{7+3\sqrt{5}}{2}. \end{aligned}$$

But

$$\begin{aligned} \left(\frac{3+\sqrt{5}}{2}\right)^2 &= \frac{7+3\sqrt{5}}{2}. \end{aligned}$$

Therefore

$$\sigma_1 = \frac{3+\sqrt{5}}{2} \approx 2.6180.$$

Similarly:

$$\sigma_2 = \frac{3-\sqrt{5}}{2} \approx 0.3820.$$

The first singular value is much larger.

The data matrix is close to rank 1.

The Right Singular Vectors

One valid right singular-vector matrix is

$$V \approx \begin{bmatrix} 0.8507 & -0.5257 \\ 0.5257 & 0.8507 \end{bmatrix}.$$

The first column is
the first PCA direction:

$$\mathbf{v}_1 \approx (0.8507, 0.5257)^T.$$

The second column is perpendicular.

Signs may differ
depending on the SVD implementation.

Changing

$$\mathbf{v}_j$$

to

$$-\mathbf{v}_j$$

does not change the principal axis.

Rank-One Reconstruction Error

Keep only

$$\sigma_1.$$

Then

$$\begin{aligned} & \|X_c - X_{c,1}\|_F^2 \\ &= \sigma_2^2 \\ &= \frac{7-3\sqrt{5}}{2} \\ &\approx 0.1459. \end{aligned}$$

Mean squared reconstruction distance:

$$\begin{aligned} & \frac{\sigma_2^2}{4} \\ &= \lambda_2 \approx 0.03647. \end{aligned}$$

This is exactly the result obtained
from the covariance eigenvalues.

Why Use the SVD Instead of Forming the Covariance Matrix?

Covariance approach:

form

$$X_c^T X_c,$$

then compute eigenvectors.

SVD approach:

decompose

$$X_c$$

directly.

A numerical advantage:

forming

$$X_c^T X_c$$

squares the singular values.

Thus its condition number is squared:

$$\kappa(X_c^T X_c) = \kappa(X_c)^2$$

when the relevant singular values are nonzero.

This is the same numerical warning
we encountered with least-squares normal equations.

SVD Also Adapts to Matrix Shape

If N
is very large,
forming an $N \times N$
covariance matrix may be expensive.

Direct or truncated SVD methods
can work with the data matrix more naturally.

If only the first few components are needed,
specialized algorithms can estimate only
the leading singular vectors.

The mathematical PCA problem is unchanged.

Only the computational route differs.

Centering Reduces Rank

Because each centered feature column has mean zero:
 $\mathbf{1}^T X_c = \mathbf{0}^T$.

Therefore the rows of X_c
are linearly dependent whenever
 $P > 0$.

Hence
 $\text{rank}(X_c) \leq \min(P - 1, N)$.

So at most $\min(P - 1, N)$
principal components
can have nonzero variance.

From PCA to an Encoder

For a centered observation
 $\mathbf{q} \in \mathbb{R}^N$,

PCA encoding is

$$\mathbf{z} = V_d^T \mathbf{q} \in \mathbb{R}^d.$$

This maps

$$N$$

numbers

to

$$d$$

numbers.

The vector

$$\mathbf{z}$$

is a compressed representation.

In machine-learning language:

this map can be viewed as an

encoder.

The PCA Decoder

Given the compressed coordinates

$$\mathbf{z},$$

reconstruct the centered observation:

$$\hat{\mathbf{q}} = V_d \mathbf{z}.$$

This is the

decoder.

Combining encoder and decoder:

$$\hat{\mathbf{q}}$$

$$= V_d (V_d^T \mathbf{q})$$

$$= V_d V_d^T \mathbf{q}.$$

Exactly the PCA orthogonal projection.

What Is an Autoencoder?

An **autoencoder**

tries to reconstruct its input.

General structure:

$$\mathbf{x} \rightarrow \text{encoder} \rightarrow \mathbf{z} \rightarrow \text{decoder} \rightarrow \hat{\mathbf{x}}.$$

The middle vector

$$\mathbf{z}$$

is called the

latent representation

or

bottleneck.

If

$$d < N,$$

the bottleneck forces compression.

Training minimizes reconstruction error.

A Linear Autoencoder

For centered data,

let the encoder be linear:

$$\mathbf{z} = E^T \mathbf{q},$$

where

$$E \in \mathbb{R}^{N \times d}.$$

Let the decoder also be linear:

$$\hat{\mathbf{q}} = D \mathbf{z},$$

where

$$D \in \mathbb{R}^{N \times d}.$$

Thus

$$\hat{\mathbf{q}} = DE^T \mathbf{q}.$$

For the whole dataset:

$$\hat{X}_c = X_c E D^T.$$

Check the Matrix Dimensions

For row-oriented data:

X_c :

$$P \times N.$$

E :

$$N \times d.$$

Encoded data:

$$Z = X_c E:$$

$$P \times d.$$

D^T :

$$d \times N.$$

Decoded data:

$$Z D^T:$$

$$P \times N.$$

Therefore

$$\hat{X}_c = X_c E D^T$$

has the same shape as the input data matrix.

Linear Autoencoder Loss

Use mean squared reconstruction error:

$$g(E,D) = \frac{1}{P} \|X_c - X_cED^T\|_F^2.$$

The product

$$ED^T$$

has rank at most

$$d.$$

Therefore

$$X_cED^T$$

also has rank at most

$$d.$$

So a linear bottleneck autoencoder
searches among low-rank reconstructions.

That brings us directly back to truncated SVD.

PCA Is One Optimal Linear Autoencoder

Choose

$$E = D = V_d.$$

Then

$$\begin{aligned} \hat{X}_c \\ = X_cV_dV_d^T. \end{aligned}$$

But

$$X_cV_dV_d^T$$

is exactly the PCA / truncated-SVD reconstruction.

Since truncated SVD gives the best rank- d
Frobenius approximation:

$$E = D = V_d$$

produces a global minimum
of the linear reconstruction problem.

Thus PCA can be realized
as a linear autoencoder with tied orthonormal weights.

But the Autoencoder Weights Are Not Unique

Suppose

$$A \in \mathbb{R}^{d \times d}$$

is invertible.

Let

$$E' = V_dA$$

and

$$D' = V_d(A^{-1})^T.$$

Then

$$\begin{aligned} &E'D'^T \\ &= V_dA[V_d(A^{-1})^T]^T \\ &= V_dAA^{-1}V_d^T \\ &= V_dV_d^T. \end{aligned}$$

Same reconstruction.

Different internal coordinates.

What the Linear-Autoencoder Equivalence Really Means

An unconstrained linear autoencoder
with a d -dimensional bottleneck
and squared reconstruction loss
can attain the PCA-optimal reconstruction subspace.

But:
its learned encoder columns
need not equal the principal-component vectors.

The latent coordinates may be
an invertible linear transformation
of PCA coordinates.

So the precise statement is:
optimal reconstruction subspace = PCA principal subspace.

Not:
every optimal autoencoder parameterization
literally equals the PCA eigenvectors.

Why Centering Still Matters

PCA operates on
 $X_c = X - \mu.$

Our linear autoencoder derivation
also assumes centered inputs.

Without centering:
a purely linear map through the origin
must spend capacity representing the data mean.

With biases:
an autoencoder can learn an affine offset.

For the clean PCA equivalence:

center the data first
or include appropriate bias terms.

What Changes With Nonlinear Autoencoders?

Suppose the encoder becomes

$$\mathbf{z} = \phi(E^T \mathbf{q})$$

and the decoder is also nonlinear.

Then reconstruction is no longer restricted
to an orthogonal linear subspace.

A nonlinear autoencoder may represent
curved low-dimensional structure.

- But:
- optimization becomes nonconvex;
 - the simple eigenvector solution disappears;
 - latent coordinates become less uniquely interpretable.

We will return to nonlinear networks later.

For now:
PCA is the canonical linear case.

PCA Versus a General Linear Autoencoder

	PCA	Linear Autoencoder
Encoder	V_d^T	learned linear map
Decoder	V_d	learned linear map
Latent dimension	d	d
Loss	squared reconstruction	squared reconstruction
Optimal subspace	principal subspace	principal subspace
Coordinates unique?	up to signs / degenerate eigenspaces	no

PCA gives a canonical orthonormal coordinate system
ordered by variance.

A general linear autoencoder
need not choose that particular basis
inside the same optimal subspace.

SVD in Python

Use the running dataset:

```
import numpy as np

X = np.array([[0., 1.],
```

```

        [1., 2.],
        [2., 2.],
        [3., 3.]])

mu = np.mean(X, axis=0)
Xc = X - mu

L, s, Vt = np.linalg.svd(
    Xc,
    full_matrices=False
)

```

New Python idea:

```

np
returns
    L,
singular values,
and
 $V^T$ .

```

full_matrices=False
requests the economy-size decomposition.

Inspect the SVD

```

print(s)
print(Vt.T)

```

One possible output:

```

[2.61803399  0.38196601]

[[-0.85065081 -0.52573111]
 [-0.52573111  0.85065081]]

```

The signs may differ.

Compare with Lesson 1.23:

PCA eigenvalues satisfy

```

eigenvalues = s**2 / len(X)

```

giving

```

[1.71352549  0.03647451]

```

Exactly the covariance eigenvalues.

PCA Coordinates Directly From the SVD

Right singular vectors:

```

V = Vt.T

```

First principal direction:

```

v1 = V[:, 0]

```

PCA coordinates:

$$Z = Xc @ V$$

But SVD also gives

$$Z_{\text{from_svd}} = L * s$$

because multiplying each column of

$$L$$

by the corresponding singular value

gives

$$L\Sigma.$$

Check:

```
np.allclose(Z, Z_from_svd)
```

Result:

True.

Rank- d Reconstruction in Python

Keep

$$d$$

components:

$$d = 1$$

```
Ld = L[:, :d]
sd = s[:d]
Vtd = Vt[:d, :]
```

```
Xc_hat = (
    Ld
    @ np.diag(sd)
    @ Vtd
)
```

Equivalent PCA form:

$$Vd = V[:, :d]$$

$$Xc_hat_2 = Xc @ Vd @ Vd.T$$

Check:

```
np.allclose(Xc_hat, Xc_hat_2)
```

Result:

True.

Same reconstruction.

Reconstruction Error From the Singular Values in Python

For

$$d = 1:$$

```
total_squared_error = np.sum(
    s[d:]**2
)
```

```
mean_squared_error = (
    total_squared_error / len(X)
)
```

Results:

$\text{total_squared_error} \approx 0.14590$,

and

$\text{mean_squared_error} \approx 0.03647$.

No reconstructed points are needed
to compute the optimal PCA error.

The singular values already contain it.

A Linear Autoencoder in Python

Use the PCA weights:

```
d = 1

E = V[:, :d]
D = V[:, :d]

Z = Xc @ E

Xc_hat = Z @ D.T
```

Here:

E

is the encoder matrix;

D

is the decoder matrix;

Z

is the bottleneck representation.

Because

$$E = D = V_d:$$

the reconstruction is exactly PCA.

A Nonunique Autoencoder Parameterization in Python

For

$$d = 1,$$

choose any nonzero scalar

$$a.$$

For example:

$$a = 3.0$$

```
E2 = E * a
D2 = D / a
```

```
Xc_hat_2 = Xc @ E2 @ D2.T
```

Then

```
np.allclose(Xc_hat, Xc_hat_2)
```

returns

True.

The latent coordinate has been rescaled,
but reconstruction is unchanged.

This is the simplest example
of autoencoder nonuniqueness.

Three Equivalent Views of the Same Linear Structure

View	Main object
Covariance PCA	eigenvectors of $(1/P)X_c^T X_c$
SVD	right singular vectors of X_c
Linear autoencoder	optimal rank- d reconstruction subspace

Covariance language emphasizes
variance.

SVD language emphasizes
matrix approximation.

Autoencoder language emphasizes
representation and reconstruction.

The underlying linear subspace is the same.

What did we learn?

- ◆ If $X_c = L\Sigma V^T$, then the PCA directions are the right singular vectors in V .
- ◆ Covariance eigenvalues and singular values satisfy $\lambda_j = \sigma_j^2/P$.
- ◆ The rank- d PCA reconstruction equals the truncated SVD $L_d \Sigma_d V_d^T$.
- ◆ The squared optimal reconstruction error is the sum of the squares of the omitted singular values.
- ◆ PCA can be written as a linear encoder $\mathbf{z} = V_d^T \mathbf{q}$ and decoder $\hat{\mathbf{q}} = V_d \mathbf{z}$.
- ◆ A general optimal linear autoencoder recovers the PCA principal subspace, although its internal coordinates need

Next

- PCA describes continuous low-dimensional directions in a data cloud.
- Another unsupervised task seeks discrete groups of similar observations.
- Next we replace subspace approximation with prototype-based clustering.

- **1.25 - Clustering and k-Means**

Prepared by Dr. Jodin Morey.

Materials for Other Courses Found at MathTalker.org