Dimensionality Reduction

Data Intelligence and Learning (DIAL) Lab
Prof. Jongwuk Lee

Dimensionality Reduction Basics

High-dimensional Data

- **High-resolution images**
  - Thousands of pixels

High-dimensional Data

- **News articles**
  - Vocabulary of hundreds of thousands of words

BASEBALL'S BACK: A preview of the Nationals and the rest. Sports

The Washington Post

Tornadoes tear through the Lone Star State

CAMPAIGN 2012

Romney bolsters his case to GOP with triple win

CAPTURES WISCONSIN, MARYLAND, D.C.

Santorum digs in heels, looks ahead to Pa.

MARYLAND

Delaney tops Garagiola in key Democratic race

GPT-3는 한국어의 미래를 어떻게 바꿀까

〈AI 언어모델〉

뉴웨이브

윤석찬

AWS 수석 테크에반젤리스트

최근 뉴스 기사나 시(詩)를 척척 써내는 자연어처리(NLP) 인공지능(AI) 모델인 GPT-3(Generative Pre-Training 3)가 개발돼 세상을 깜짝 놀라게 했다. AI 연구에서 사람인지 AI인지를 판단하는 튜링 테스트(Turing Test)가 있다. 컴퓨터와의 대화에서 컴퓨터의 반응을 인간의 반응과 구별할 수 없다면 컴퓨터가 스스로 사고할 수 있는 것으로 간주한다는 것이다. 물론 GPT-3의 한계도 분명 존재하지만, 그 능력은 우리가 그동안 예상했던 것을 뛰어넘는 것이 사실이다.

GPT-3는 지난 6월 초 오픈AI라는 연구소에서 발표한 AI 언어 모델로 약 4990억개의 데이터 세트 중 가중치 샘플링을 한 3000억여개로 학습했다. 모델 훈련에 들어간 매개변수만 1750억개다. 수년간 인터넷에 올라온 5조개 단위의 문서 데이터 세트를 학습한 것인데, 간단한 키워드를 제시하면 이에 걸맞은 대답을 해준다. 한 번의 모델 학습에만 수십억 원 정도의 컴퓨팅 용량이 소모되는 것으로 예상되는 이 모델은 데이터 사이즈와 규모 면에서 세계 최강이다. 베타 버전을 체험한 AI 연구자들도 혀를 내두를 정도로 놀라운 예제들이 인터넷에 넘쳐난다. 학습 데이터에는 한국어도 포함돼 한국어 질문에 답하거나 자연스러운 한국어 기사도 만들 수 있다.

국내에서도 한국어 기반 NLP 모델에 대한 연구가 활발히 진행 중이다. 지난 6월 SK텔레콤의 연구진은 한국어 위키 프로젝트, 한국어 뉴스와 기타 소스들을 활용해서 문장 1억2500만개와 단어 16억개를 기반으로 KoGPT-2라는 모델을 오픈 소스로 공개해 시장에서 많은 주목을 받았다. KoGPT-2라는 이름에서 알 수 있듯 기존 GPT-2 모델을 한국어로 학습시켰다. SK텔레콤은 아마존의 머신러닝 연구진과 함께 아마존웹서비스(AWS) 클라우드의 64개 그래픽처리장치(GPU)를 1주일 동안 사용해 모델을 완성했다. GPT-3가 유료로 서비스될 예정임을 감안하면 이를 오픈 소스로 공개한 것이 더욱 눈에 띈다.

GPT-3는 지난해 초 발표된 GPT-2에서 더 발전된 모델로, 더 많은 데이터를 기반으로 더 많은 컴퓨팅 용량으로 학습한 점이 차별화된다. GPT-2 역시 당시 놀라운 성능을 보인 것을 감안하면 데이터 크기와 컴퓨팅 용량은 AI 발전에 중요한 역할을 하고 있음을 알 수 있다. 그러나 일부 국내 AI 연구자들은 GPT-3가 아무리 좋은 성능을 갖췄더라도 한국어 학습 데이터가 절대적으로 부족하다면 이를 활용하는 것이 어렵다고 말한다.

이제 AI가 한국어의 미래를 바꿀 날이 멀지 않았다. 정부에서도 디지털 뉴딜을 통한 AI 학습용 데이터 구축에 투자하는 등 미래를 위한 준비를 착착 진행하고 있다. 대기업들도 대규모 한국어 데이터 수집에 적극 나서고 있다. KoGPT-2와 같은 고급 언어 모델 개발을 위해서는 많은 양의 학습 데이터, 상당한 양의 컴퓨팅 자원, NLP에 대한 전문 지식이 필요하다.

하지만 클라우드는 누구나 쉽게 머신러닝을 활용하고, 이에 필요한 대용량 IT 자원을 제공해 학습할 수 있는 환경을 마련해준다. 개발자들은 더 적은 수의 GPU를 사용해 더 빠른 모델의 학습이 가능하다. 특히 이들 기술이 오픈 소스로 공개됨으로써 누구나 쉽게 접근해 활용할 수 있는 길이 열린 것은 큰 성과라 하겠다. 한국에서도 선순환 생태계가 만들어지고 있으며, 이를 기반으로 다양한 아이디어가 쏟아져 나올 것으로 기대된다.

Published By Denton Publications Inc.

THE SUN

April 25, 2020

Valley News · EDITION ·

suncommunitynews.com

GETTING READY FOR COLLEGE

CFES helps students negotiate difficult times

By Tim Rowland

STAFF WRITER

ESSEX | CFES Brilliant Pathways, an Essex-based nonprofit that helps usher disadvantaged kids into meaningful careers, is pivoting to guide students in a time when the short-term future of higher education is unknown, and the effects of the coronavirus pandemic may have far-reaching effects that permanently change the face of higher education.

To that end, the organization, which works with 12 North County schools as well as 200 schools across the nation, is working with sports stars and other inspirational speakers to provide online content, and encouraging students to stay disciplined in uncertain times.

In a heartbeat, K-12 schools across the country went from the traditional classroom setting that have been around since the days of the one-room schoolhouse, to virtual learning that depends on home-based lesson plans and online resources.

"The COVID shutdown will transform forever how we ready students for college and careers and how education is delivered," said CFES President Rick Dalton. "This is a challenging time, but one that is already producing innovative ways for helping students pursue postsecondary opportunities."

That increases both the challenges and opportunities for nonprofits like CFES, which is playing a key role in keeping students engaged and preparing them for college and careers.

The pandemic has created an unprecedented need for online student support of daily homework assignments, college and career readiness and mentoring, Dalton said. As such, CFES has shifted to an online platform to meet the needs of its schools in these areas by providing virtual mentors and tutors, links to virtual campus tours, online college fairs and financial aid webinars.

High school juniors, for example, need help writing college essays, completing applications and navigating the financial aid process. Seniors, meanwhile, are concerned about the transition to college and careers that will undoubtedly be altered by COVID, Dalton said.

CFES is addressing these concerns by hosting live webinars on the components of college and career readiness. The interactive series began in March with a webinar on leadership featuring Miami Dolphins Head Coach Brian Flores.

And this week, Don Outing, vice president for equity and community at Lehigh University, spoke about key strategies that start long before a student ever sets foot on a college campus. Outing is internationally known for helping underserved students pursue careers in STEM.

On April 30, financial expert Yanely Espinal, director of educational outreach at Next Gen Personal Finance, who will discuss how high school students can best prepare financially for college. The webinars are recorded and posted on the CFES website at brilliantpathways.org/webinars.

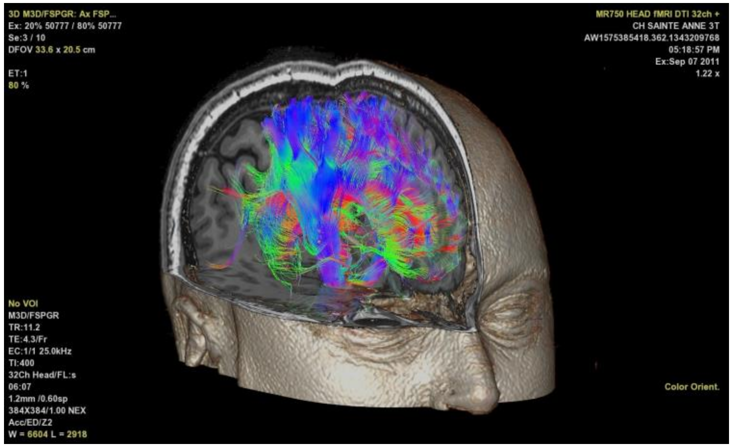
CFES schools that are encouraged to emphasize essential skills such as perseverance, agility and leadership are experiencing strong student engagement. Beekmantown Central School in Clinton County, for example, has near-perfect online attendance and strong student support from CFES-GEAR UP Fellows.

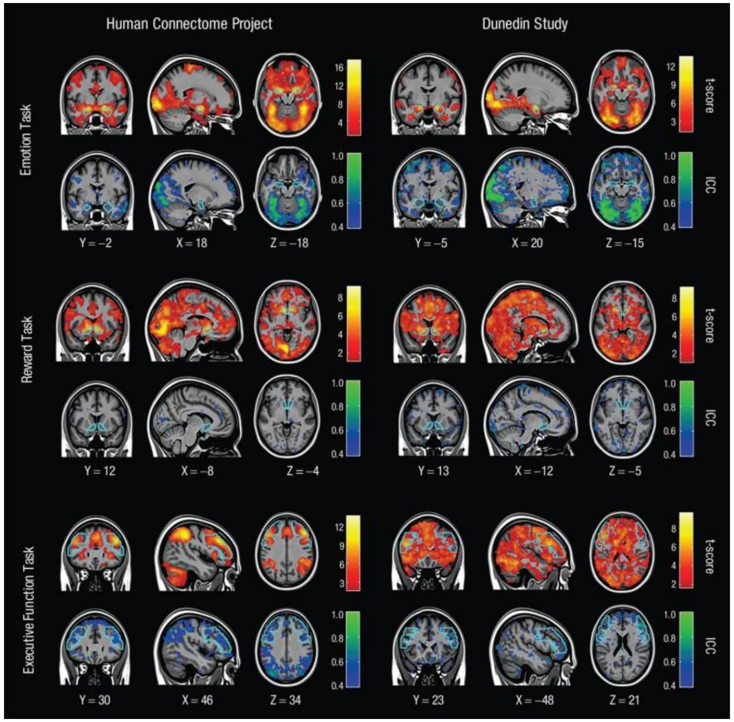
"Educators at Beekmantown deserve a ton of credit for being way out in front of this crisis," said Brett McClelland, a CFES-GEAR UP Fellow at Beekmantown who co-produced this Essential Skills video with Fellow Mallory Carpenter. "All year long we teach students Essential Skills, which they need now more than ever." ■

High-dimensional Data

➤ 3D brain imaging data

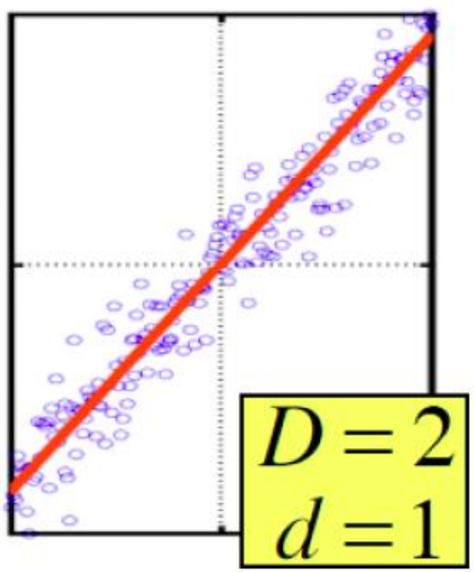
- > 100 MB per scan

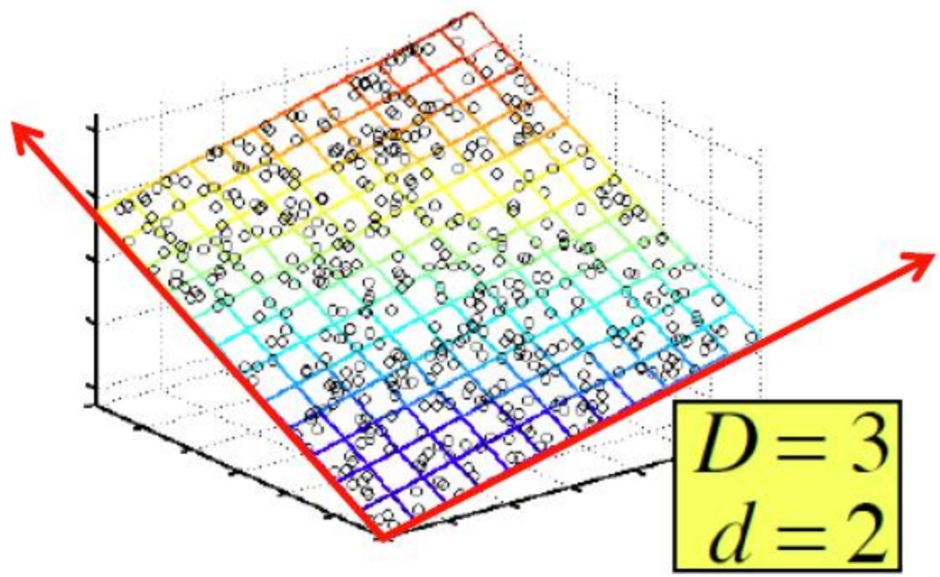




What is Dimensionality Reduction?

- **Goal: Unsupervised learning methods for extracting hidden structures from high-dimensional datasets**
- **Assumption: Data objects lie on or near a low d-dimensional subspace.**





Example of Dimensionality Reduction

- How to reduce dimensionality without loss of information?
- Using different axis [2 3 0 0 0] and [0 0 2 4 2], data can be represented by two-dimensional space.

Five-dimensional space

	Mon	Tue	Wed	Thu	Fri
Alice	2	3	0	0	0
Bob	4	6	0	0	0
Carol	6	9	0	0	0
David	0	0	2	4	2
Eve	0	0	3	6	3
Frank	0	0	1	2	1

Two-dimensional space

	F1	F2
Alice	1	0
Bob	2	0
Carol	3	0
David	0	1
Eve	0	1.5
Frank	0	0.5

Low-Dimensional Projection

- Instead of picking a subset of the features, we obtain new features by combining existing features $\mathbf{x}_1, \dots, \mathbf{x}_d$.
- We can reduce dimensions, i.e., $k < d$.

$$\mathbf{z}_1 = \sum_{i=1}^{d} w_i^{(1)} \mathbf{x}_i$$

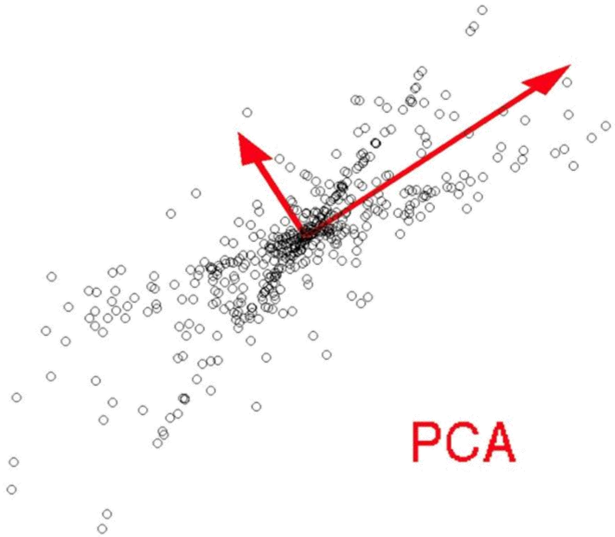
$$\vdots$$

$$\mathbf{z}_k = \sum_{i=1}^{d} w_i^{(k)} \mathbf{x}_i$$

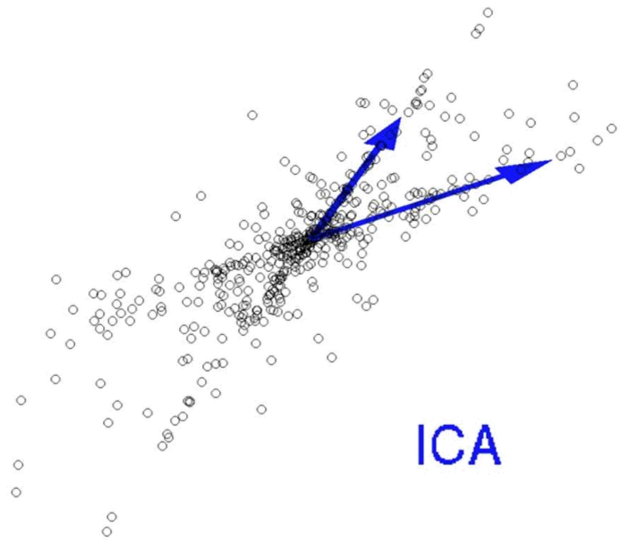
- New features are represented by linear combinations of old features.

Linear Dimensionality Reduction

- **Principle component analysis (PCA)**
  - Finding directions of maximal variance
- **Independent component analysis (ICA)**
  - Finding directions of maximal independence
- **Non-negative matrix factorization (NMF)**



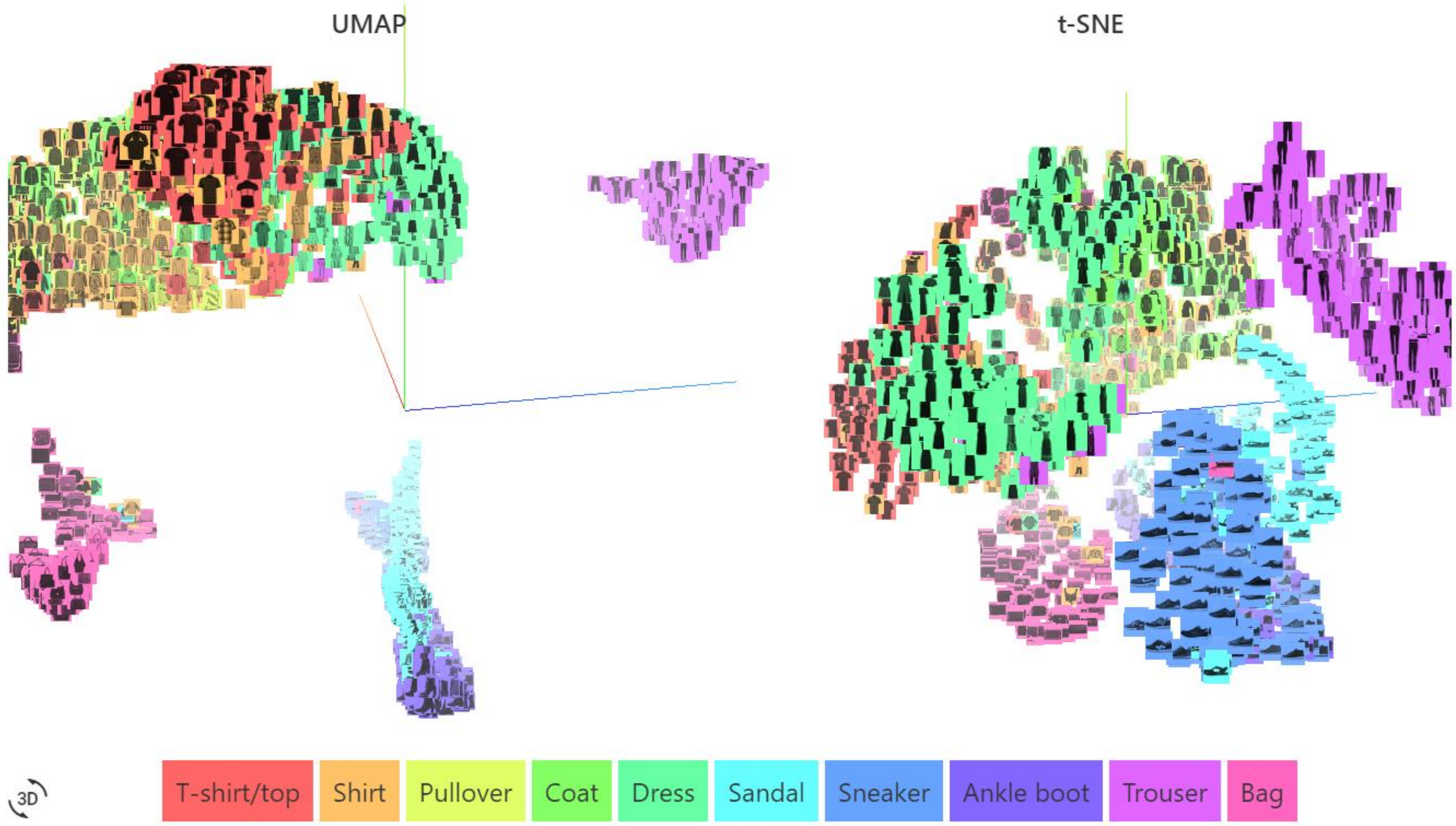
PCA



ICA

Non-linear Dimensionality Reduction

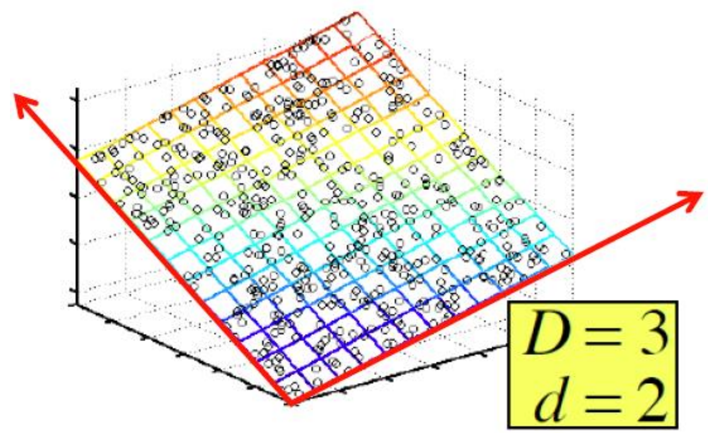
- ➢ **t-Distributed stochastic neighbor embedding (t-SNE)**
- ➢ **Uniform Manifold Approximation and Projection (UMAP)**



https://pair-code.github.io/understanding-umap/

Why Reducing Dimensions?

- Discovering hidden correlations of features
- Removing redundant and noisy features
- Easier storage and processing of the data
- Interpretation and visualization



Review: Linear Algebra

Rank of a Matrix

- **Q: What is the rank of a matrix $\mathbf{X}$?**
- **A: Dimension of the vector space spanned by its columns**
  - A: Number of **linearly independent** columns of $\mathbf{X}$.

- **Example**

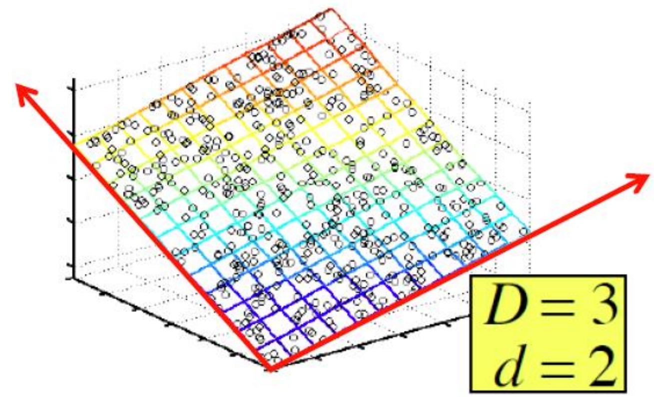
$$\mathbf{X} = \begin{bmatrix} 1 & -2 & 3 \\ 2 & -3 & 5 \\ 1 & 1 & 0 \end{bmatrix} \; has\ rank\ r = 2.$$

  - The first two columns are **linearly independent**, so the rank is 2.
  - The third column is equal to the sum of the first and second columns, so the rank must be less than 3.

Rank is Dimensionality

- **Cloud of points 3D space:**
  - Each column means a point.
  - Think of point positions as a matrix:

$$\mathbf{X} = \begin{bmatrix} 1 & -2 & 3 \\ 2 & -3 & 5 \\ 1 & 1 & 0 \end{bmatrix}$$



- **Old basis vectors: [1 0 0] [0 1 0] [0 0 1]**

- **We can rewrite coordinates more efficiently!**
- **New basis vectors: [1 2 1] [-2 -3 1]**
  - **We reduced the number of coordinates!**

Linear Independence

- **A set of vectors is said to be linearly dependent.**
  - if one of the vectors in the set can be defined as a linear combination of the others.

- **How to check the linear independence?**
  - n vectors are **linearly independent** if and only if the determinant of the matrix formed by taking the vectors as its columns is **non-zero**.

$$A = \begin{bmatrix} 1 & -3 \\ 1 & 2 \end{bmatrix} \Rightarrow \det(A) = 1 \cdot 2 - 1 \cdot (-3) = 5 \neq 0$$

  - Because the determinant is non-zero, two vectors $(1, 1)$ and $(-3, 2)$ are linearly independent.

Orthogonality

- **Two vectors x and y are orthogonal if their inner product $\langle x, y \rangle$ is zero.**
  - $(1, 3, 2)^{\mathrm{T}}$, $(3, -1, 0)^{\mathrm{T}}$, $(1, 3, -5)^{\mathrm{T}}$ are orthogonal to each other.

- **If two vectors are orthogonal, then they are linearly independent.**
  - However, the inverse relationship does not hold.

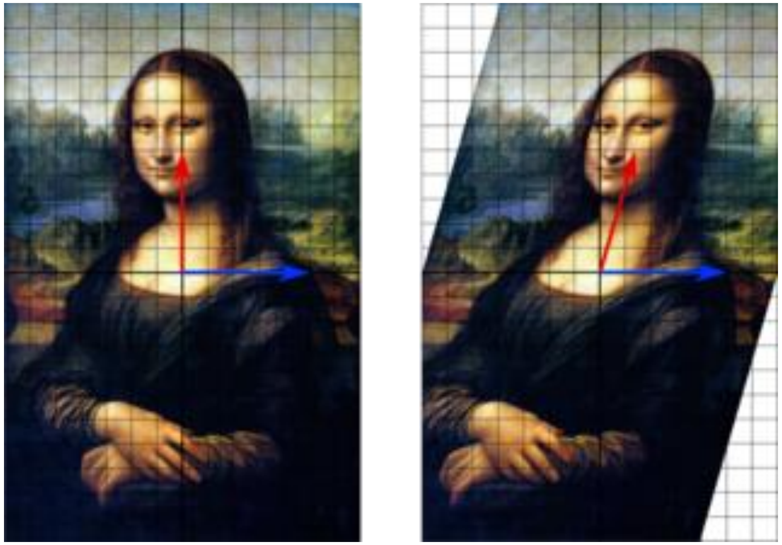
- **Two vectors are orthonormal if they are orthogonal and unit vectors.**

Eigenvalue and Eigenvector

What is an eigenvector?

- A non-zero vector $\mathbf{v}$ whose direction does not change when a linear transformation $\mathbf{A}$ is applied to it
- An eigenvalue λ is a scalar associated with the eigenvector $\mathbf{v}$.

$$\mathbf{Av} = \lambda \mathbf{v}$$



The red arrow changes direction, but the blue arrow does not.

The blue arrow is an eigenvector of this mapping because it doesn't change direction, and since its length is unchanged, its eigenvalue is 1.

How to Compute Eigenvalues?

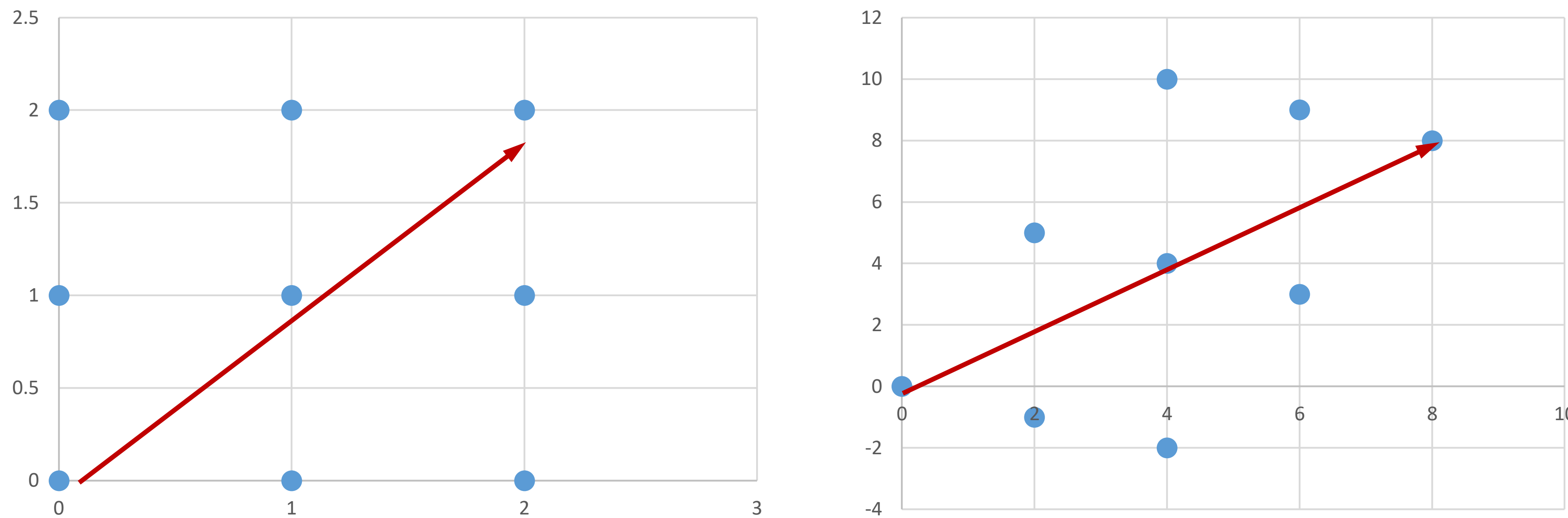
- Given $\mathbf{X} = \begin{bmatrix} 2 & 2 \\ 5 & -1 \end{bmatrix}$, compute eigenvalues and eigenvectors.
  - $\mathbf{det(X - \lambda I)} = \begin{vmatrix} 2-\lambda & 2 \\ 5 & -1-\lambda \end{vmatrix}$

    $= (2-\lambda)(-1-\lambda) - 10 = \lambda^2 - \lambda - 12 = (\lambda - 4)(\lambda + 3) = 0$

  - When $\lambda = 4$, $\begin{bmatrix} 2 & 2 \\ 5 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 4 \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \Rightarrow x_1 = x_2$
  - When $\lambda = -3$, $\begin{bmatrix} 2 & 2 \\ 5 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = -3 \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \Rightarrow x_1 = -\frac{2}{5} x_2$

How to Compute Eigenvalues?

- **Given $\mathbf{X} = \begin{bmatrix} 2 & 2 \\ 5 & -1 \end{bmatrix}$, compute eigenvalues and eigenvectors.**
  - When $\lambda = 4$, $\begin{bmatrix} 2 & 2 \\ 5 & -1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 4\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \Rightarrow x_1 = x_2$
  - When $\lambda = -3$, $\begin{bmatrix} 2 & 2 \\ 5 & -1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = -3\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \Rightarrow x_1 = -\frac{2}{5}x_2$

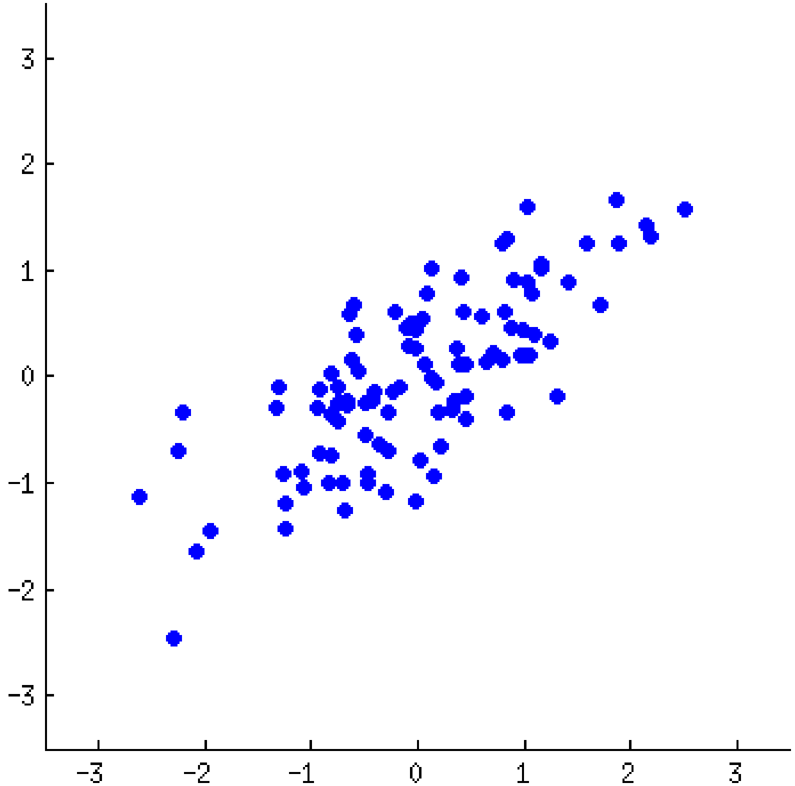


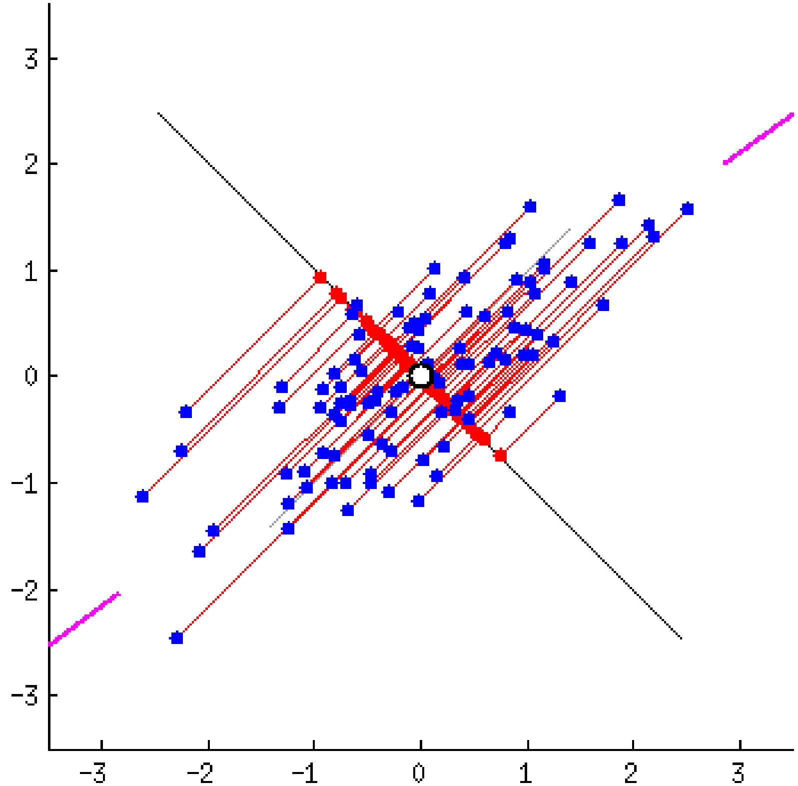
The direction of the red arrow is not changed.

Principle Component Analysis (PCA)

Motivation

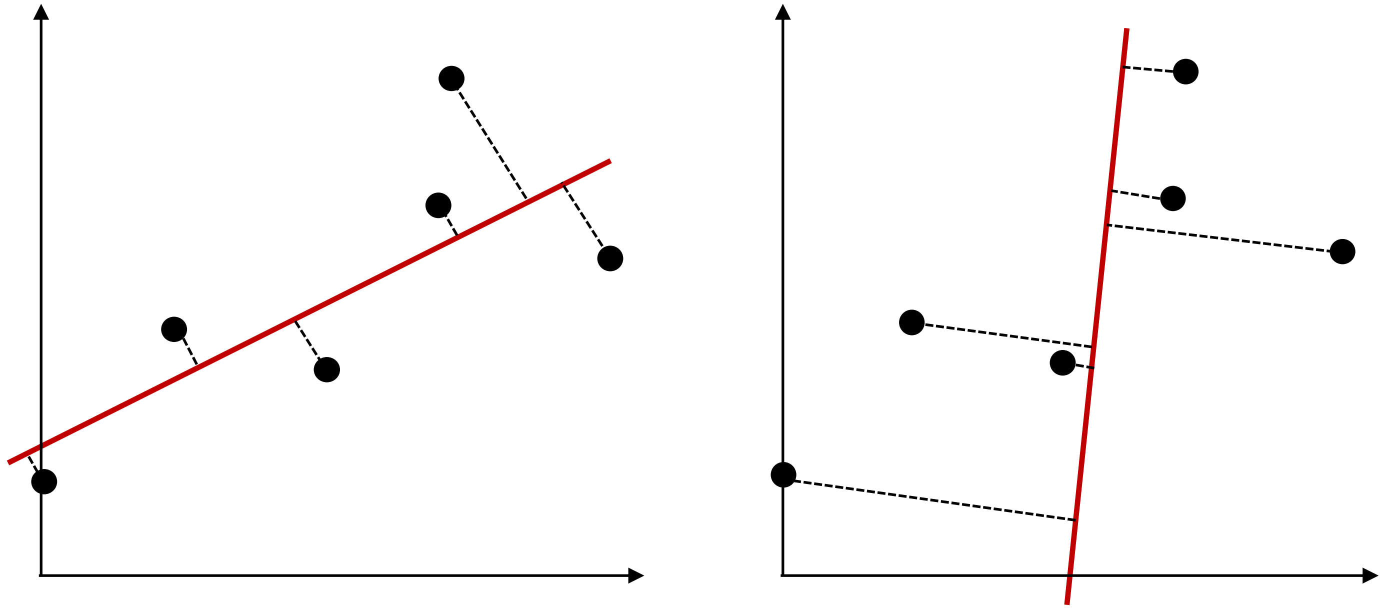
➢ An **orthogonal linear transformation** that transfers the data to a **new coordinate system**





Maximizing the Variance

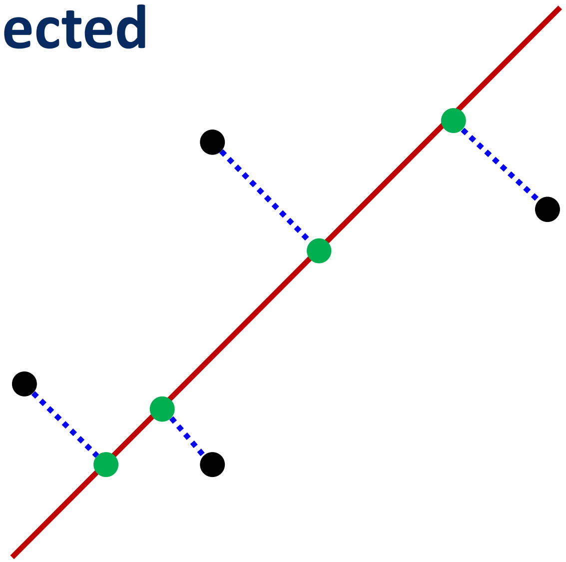
- Which maximizes the **variance**?
- Which minimizes **the sum of projected distances**?



Maximizing the Variance

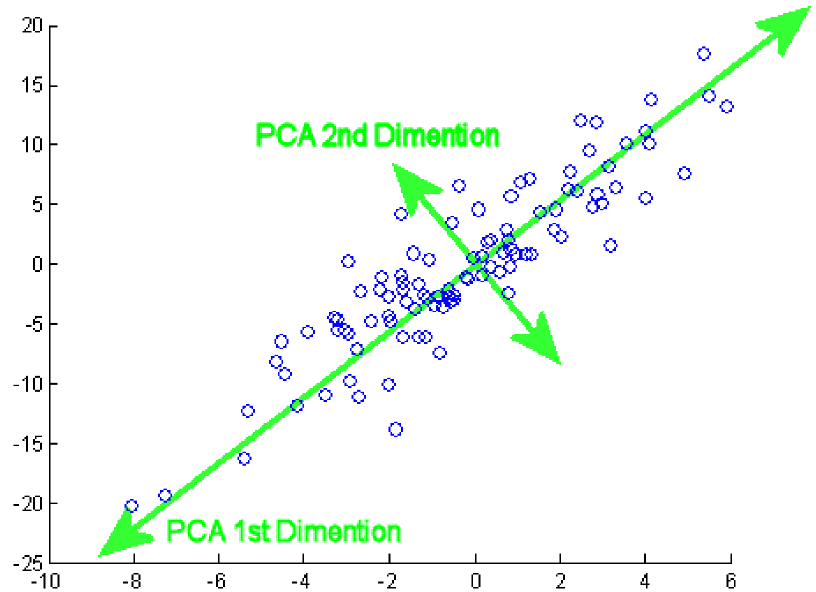
- **PCA is the orthogonal projection of the data onto a lower-dimension linear space that**
  - Maximizes the variance of projected data (**red line**).
  - Minimizes the mean squared distance between data point and projections (**sum of blue lines**).

- **Data in two-dimensional space is projected into a new one-dimensional space.**

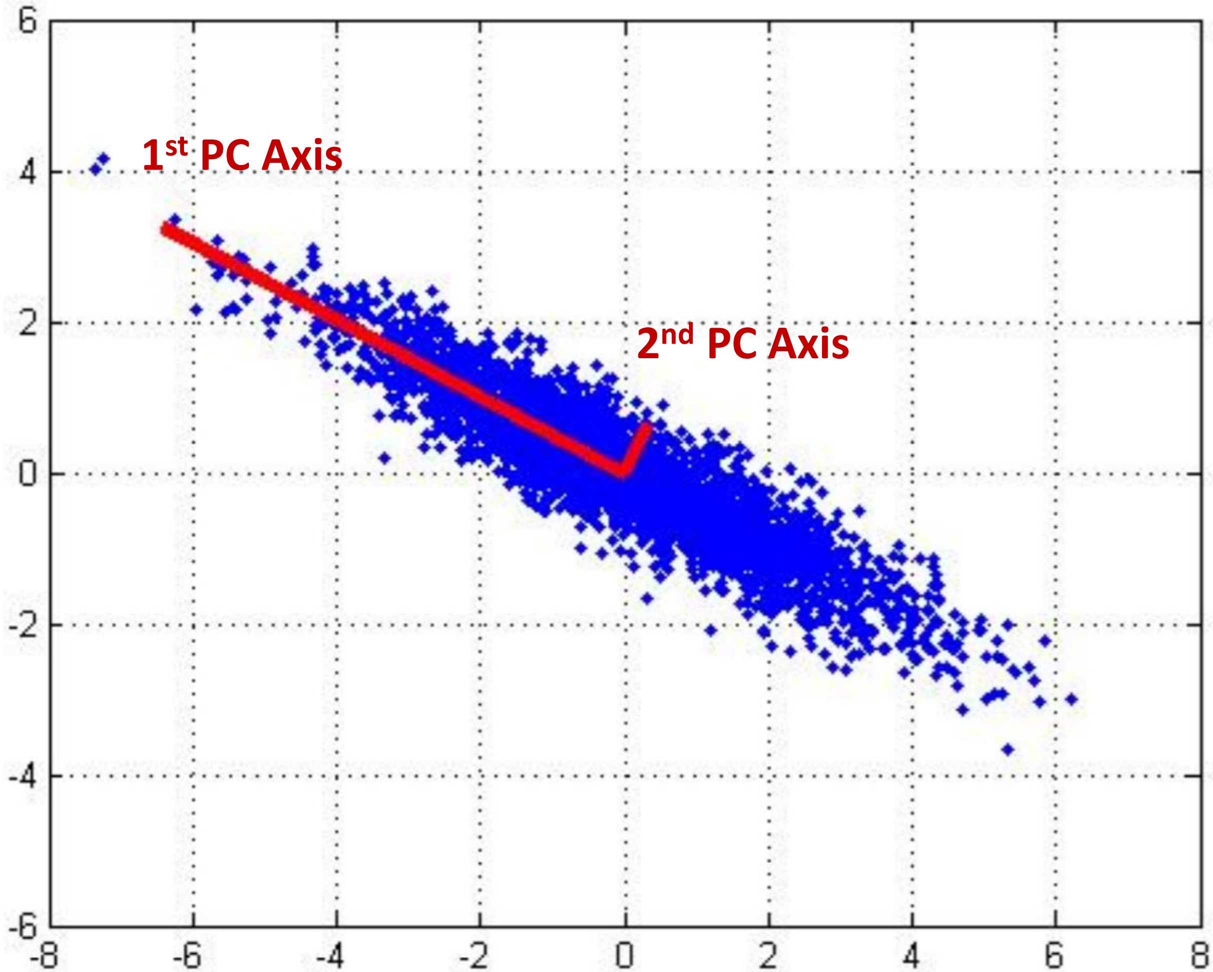


Principle Component Analysis (PCA)

- **PCA Vectors originate from the center of mass.**
- **The first principal component: points in the direction of the largest variance.**
- **Each subsequent principal component**
  - is **orthogonal to the previous ones**, and
  - points in the directions of the **largest variance of the residual subspace**.



Example



Multivariate Data

- **Let $\mathcal{D} = \{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(n)}\}$ be a training dataset.**
- **Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be a data matrix.**
  - n samples/instances/examples
  - d features/attributes

$$\mathbf{X} = \begin{bmatrix} \left[\mathbf{x}^{(1)}\right]^{\mathrm{T}} \\ \left[\mathbf{x}^{(2)}\right]^{\mathrm{T}} \\ \vdots \\ \left[\mathbf{x}^{(n)}\right]^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_d^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(n)} & x_2^{(n)} & \cdots & x_d^{(n)} \end{bmatrix}$$

# of samples

# of features

Zero-Centered Normalization

- We assume that data is **centered**.

$$\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}^{(i)} = 0$$

- Q: What if our data is not centered?
- A: Subtract the sample mean as **pre-processing**.

Sample Covariance Matrix

- **How to compute a covariance matrix?**

$$\boldsymbol{\Sigma}_{jk} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_j^{(i)} - \mu_j)(\mathbf{x}_k^{(i)} - \mu_k)$$

- **Since the data matrix is centered, we rewrite it as:**

$$\boldsymbol{\Sigma} = Cov(\mathbf{X}) = \frac{1}{n}(\mathbf{X} - \boldsymbol{\mu})^{\mathrm{T}}(\mathbf{X} - \boldsymbol{\mu}) \Rightarrow \boldsymbol{\Sigma} = \frac{1}{n}\mathbf{X}^{\mathrm{T}}\mathbf{X}$$

Note: $\boldsymbol{\mu}$ is a zero vector.

Formulating PCA

- The matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ is projected into a vector $\mathbf{v} \in \mathbb{R}^{d\times 1}$.

Assume that $\mathbf{v}$ is the unit vector, i.e., $\mathbf{v}^{\mathbf{T}}\mathbf{v} = 1$.

$$\mathbf{v}^* = \underset{\mathbf{v}:\, \|\mathbf{v}\|^2=1}{\operatorname{argmax}} Var(\mathbf{X}\mathbf{v}) = \underset{\mathbf{v}:\, \|\mathbf{v}\|^2=1}{\operatorname{argmax}} \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{x}^{(i)}\mathbf{v}\right)^{\mathrm{T}}\left(\mathbf{x}^{(i)}\mathbf{v}\right)$$

Data are zero-centered.

$$= \underset{\mathbf{v}:\, \|\mathbf{v}\|^2=1}{\operatorname{argmax}} \frac{1}{n}\sum_{i=1}^{n}\mathbf{v}^{\mathrm{T}}\left(\mathbf{x}^{(i)}\right)^{\mathrm{T}}\mathbf{x}^{(i)}\mathbf{v} = \underset{\mathbf{v}:\, \|\mathbf{v}\|^2=1}{\operatorname{argmax}} \mathbf{v}^{\mathrm{T}}\Sigma\mathbf{v}$$

Introduce a Lagrange multiplier for the equality constraint $\|\mathbf{v}\|^2 = 1$.

$$\boldsymbol{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}^{\mathrm{T}}\mathbf{x}$$

$$\mathcal{L}(v,\lambda) = \mathbf{v}^{\mathbf{T}}\boldsymbol{\Sigma}\mathbf{v} - \lambda\left(\mathbf{v}^{\mathbf{T}}\mathbf{v} - 1\right)$$

Formulating PCA

- Taking the derivative of the Lagrangian and setting it to zero,

$$\mathcal{L}(v,\lambda) = \mathbf{v}^{\mathrm{T}}\boldsymbol{\Sigma}\mathbf{v} - \lambda\left(\mathbf{v}^{\mathrm{T}}\mathbf{v} - 1\right)$$

$$\frac{d}{d\mathbf{v}}\left(\mathbf{v}^{\mathrm{T}}\boldsymbol{\Sigma}\mathbf{v} - \boldsymbol{\lambda}\left(\mathbf{v}^{\mathrm{T}}\mathbf{v} - 1\right)\right) = 0 \quad \Rightarrow \quad \boldsymbol{\Sigma}\mathbf{v} - \lambda\mathbf{v} = 0 \;\Leftrightarrow\; \boldsymbol{\Sigma}\mathbf{v} = \lambda\mathbf{v}$$

- **We can compute the PCs as the eigenvector of $\boldsymbol{\Sigma}$.**

- Recall: For a square matrix $\mathbf{A}$, the vector $\mathbf{v}$ is an eigenvector if and only if there exist eigenvalue λ such that $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$.

How to Choose PCs?

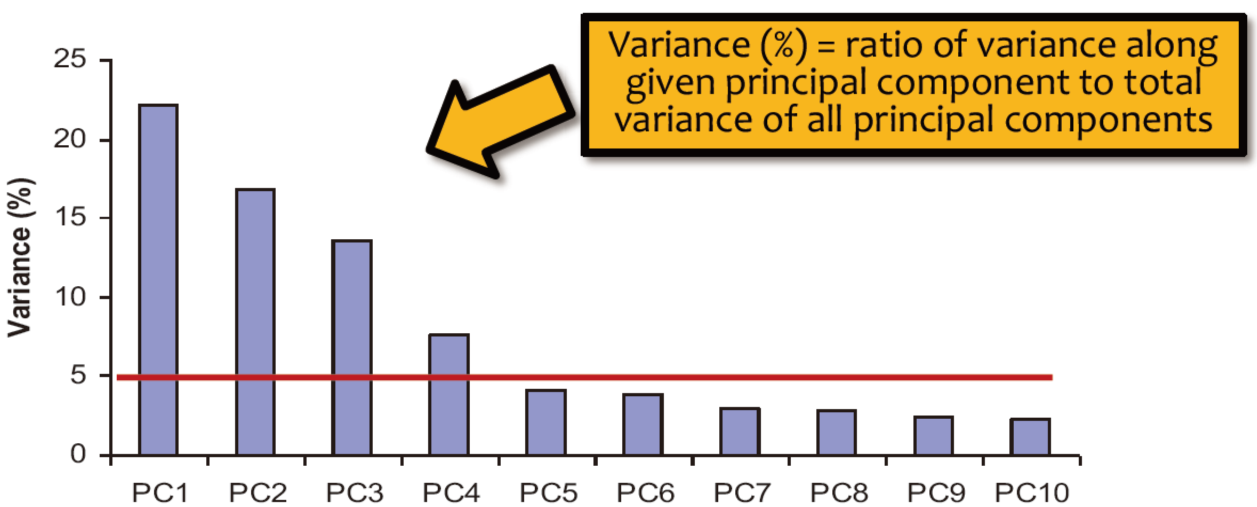
- **The eigenvalue λ denotes the amount of variability captured along that dimension.**

- **For eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots$,**
  - The 1st PC is the eigenvector $\mathbf{v}_1$ of the covariance matrix $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ associated with the largest eigenvalue.
  - The 2nd PC is the eigenvector $\mathbf{v}_2$ of the covariance matrix $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ associated with the second largest eigenvalue.
  - And so on ...

How to Choose PCs?

- For d $(d < n)$ original dimensions, sample covariance matrix is $d \times d$, and has up to d eigenvectors.

- **Q: How do we perform dimensionality reduction?**
- **A: Can ignore the components of lesser significance.**
  - May lose some information, but we lose insignificant information if the eigenvalues are small.



Example of PCA

Zero-Centered Normalization

- **Step 1: Given a data, we subtract the average.**

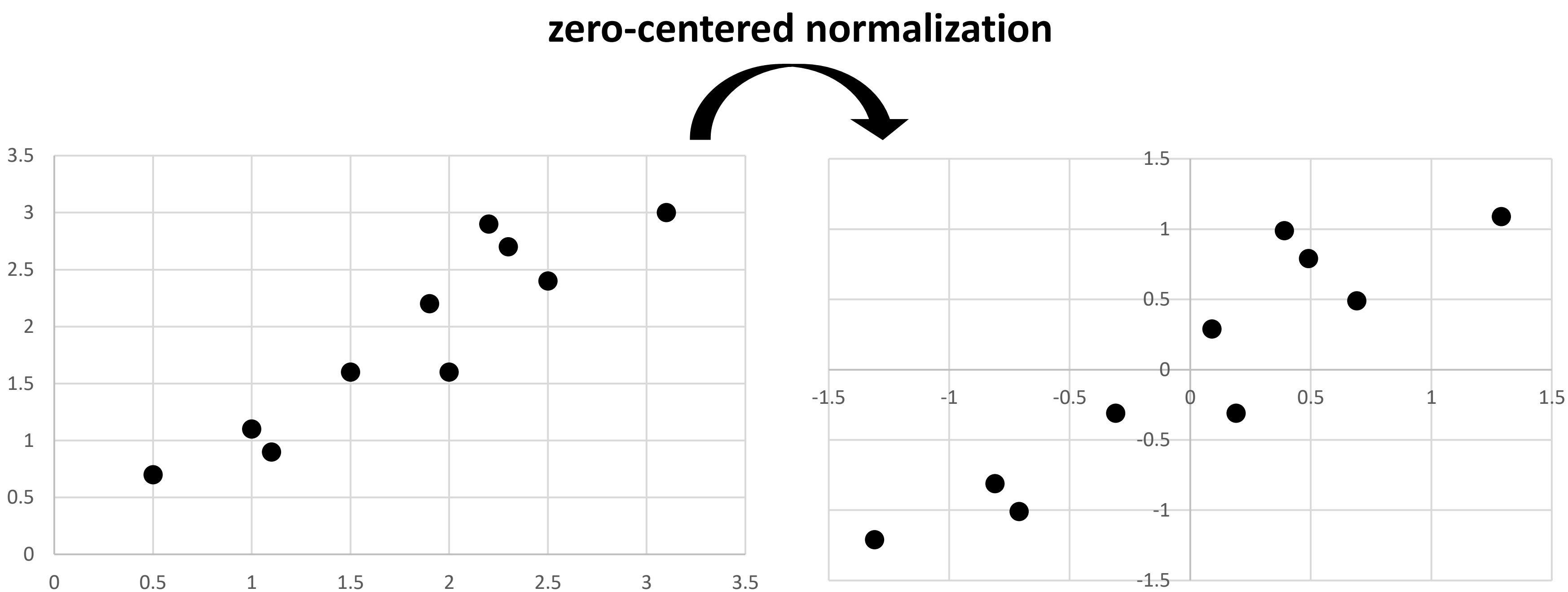
A1	A2
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

A1	A2
0.69	0.49
-1.31	-1.21
0.39	0.99
0.09	0.29
1.29	1.09
0.49	0.79
0.19	-0.31
-0.81	-0.81
-0.31	-0.31
-0.71	-1.01

http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf

Zero-Centered Normalization

- **Step 1: Given a data, we subtract the average.**

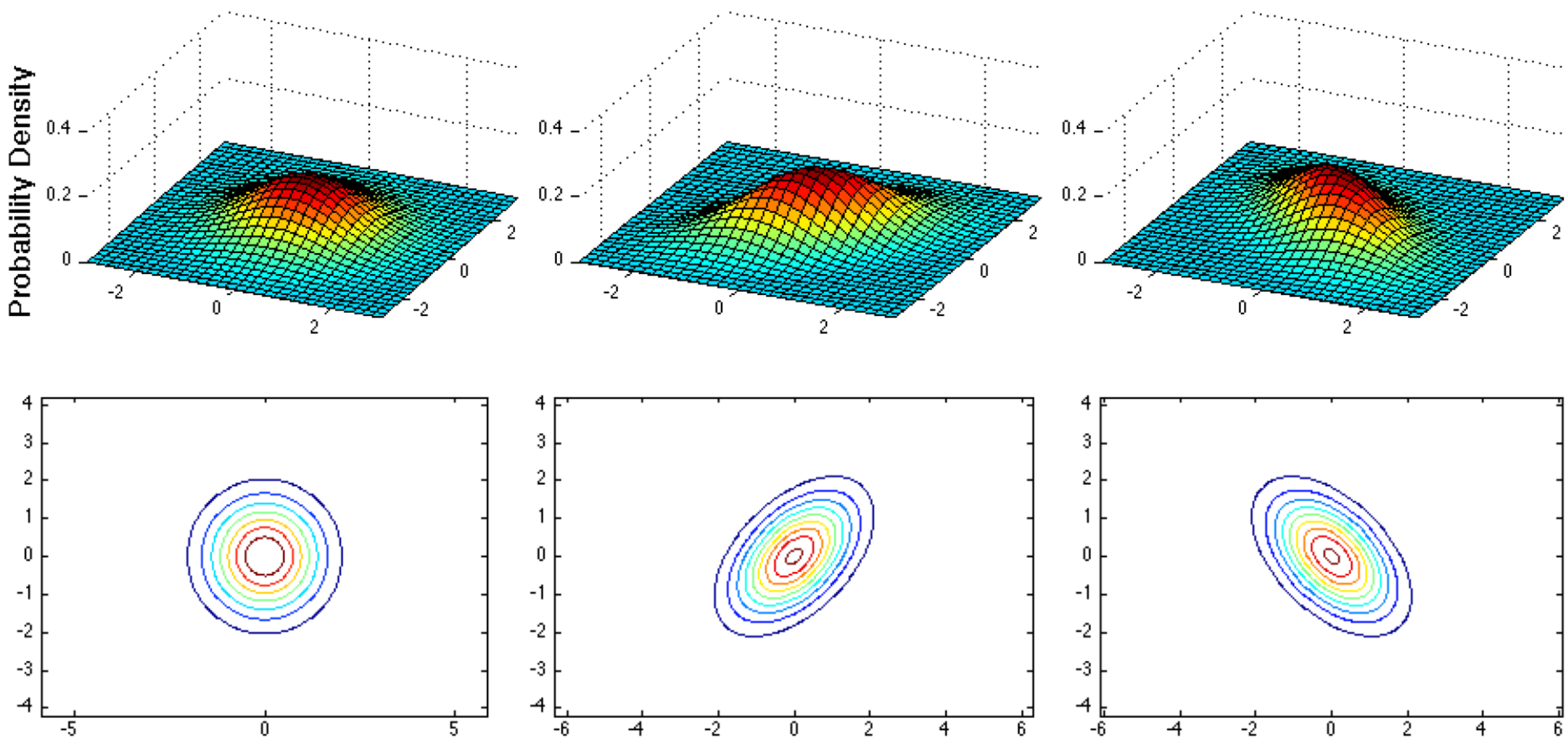


Computing a Covariance Matrix

➢ **Step 2: Calculate the covariance matrix.**

- $cov = \begin{pmatrix} .6166 & .6154 \\ .6154 & .7166 \end{pmatrix}$

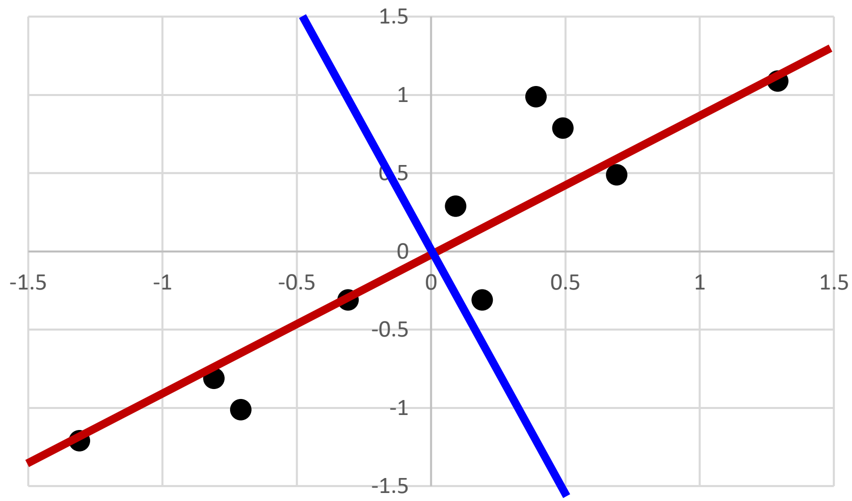
$$\boldsymbol{\Sigma} = \frac{1}{n}\mathbf{X}^{\mathrm{T}}\mathbf{X}$$



http://www.visiondummy.com/2014/04/geometric-interpretation-covariance-matrix/

Computing Eigenvectors

- **Step 3: Calculate the eigenvectors and eigenvalues for the covariance matrix.**
  - $eigenvalues = \begin{pmatrix} 1.2840 \\ 0.0491 \end{pmatrix}$
  - $eigenvectors = \begin{pmatrix} -0.6779 & -0.7352 \\ -0.7352 & 0.6779 \end{pmatrix}$



Transforming into New Features

- Construct a new feature vector $\boldsymbol{f} = (\mathbf{v}_1, \ldots, \mathbf{v}_d)$.
  - $new\ feature\ vector = \begin{pmatrix} -0.6779 & -0.7352 \\ -0.7352 & 0.6779 \end{pmatrix}$

- For approximation, it is possible to reduce the feature vector on one-dimensional space.
  - $new\ feature\ vector = \begin{pmatrix} -0.6779 \\ -0.7352 \end{pmatrix}$

Transforming into New Features

- **Deriving a new data set**
  - **Final data = row data adjust * new feature vector**

A1	A2
0.69	0.49
-1.31	-1.21
0.39	0.99
0.09	0.29
1.29	1.09
0.49	0.79
0.19	-0.31
-0.81	-0.81
-0.31	-0.31
-0.71	-1.01

$$\cdot \begin{pmatrix} -0.6779 & -0.7352 \\ -0.7352 & 0.6779 \end{pmatrix} =$$

A1’	A2’
-0.82797	-0.17512
1.77758	0.142857
-0.9922	0.384375
-0.27421	0.130417
-1.6758	-0.2095
-0.91295	0.175282
0.099109	-0.34982
1.144572	0.046417
0.438046	0.017765
1.223821	-0.16268

Represent data into new transformed space.

Transforming into New Features

- Deriving a new data set
  - Final data = row data adjust * new feature vector

A1	A2
0.69	0.49
-1.31	-1.21
0.39	0.99
0.09	0.29
1.29	1.09
0.49	0.79
0.19	-0.31
-0.81	-0.81
-0.31	-0.31
-0.71	-1.01

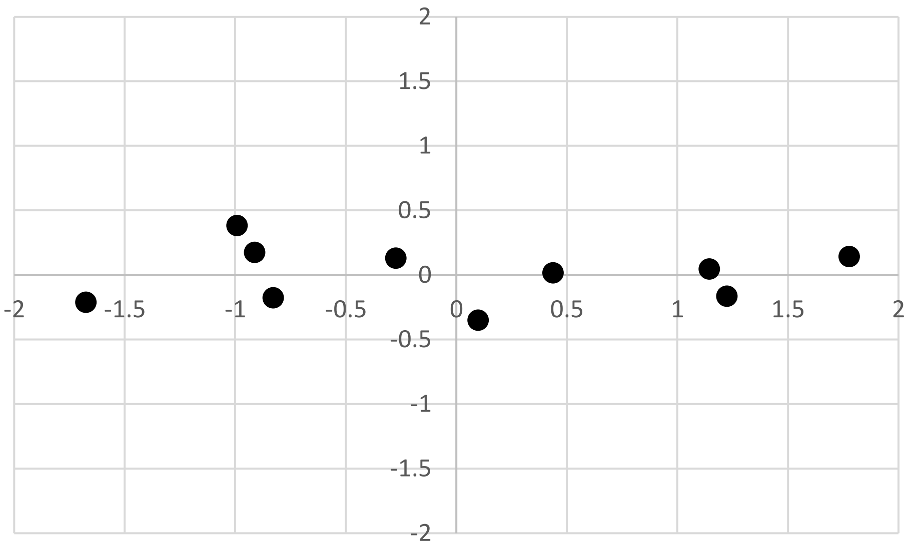
$$\cdot \begin{pmatrix} -0.6779 \\ -0.7352 \end{pmatrix} =$$

A1’
-0.82797
1.77758
-0.9922
-0.27421
-1.6758
-0.91295
0.099109
1.144572
0.438046
1.223821

Transforming into New Features

- **Transform original data into new data space using chosen components.**
  - **Final data = row data adjust * new feature vector**

A1'	A2'
-0.82797	-0.17512
1.77758	0.142857
-0.9922	0.384375
-0.27421	0.130417
-1.6758	-0.2095
-0.91295	0.175282
0.099109	-0.34982
1.144572	0.046417
0.438046	0.017765
1.223821	-0.16268



Applications of PCA

Face Recognition

- **We want to identify a specific person based on facial images.**
  - It should be robust to glasses, lighting, etc.
  - However, 256 x 256 images are high-dimensional data.
- **Solution: Build one PCA database for the whole dataset and then classify based on the weights.**

https://machinelearningmastery.com/face-recognition-using-principal-component-analysis/

Eigenfaces: Applying PCA

- **Example data set: Images of faces**
  - Eigenface approach
  - Each face $\mathbf{x}$ is 256 x 256 pixels
  - $\mathbf{x}$ is a 64K dimensional vector.

- **Form $\mathbf{X} \in \mathbb{R}^{n\times 64K}$ centered data**
  - Let n be the number of samples.

- **Compute $\boldsymbol{\Sigma} = \mathbf{X}^{\mathrm{T}}\mathbf{X}$.**

- **Problem: The $64K \times 64K$ matrix is too HUGE!**

Computational Complexity

- **Suppose that we have n samples.**
  - Eigenfaces: $n = 500$ faces, each of size $d = 64K$

- **Given $d \times d$ covariance matrix Σ, we have to compute**
  - All d eigenvectors/eigenvalues in O(d^3)
  - First k eigenvectors/eigenvalues in O(kd^2)

- **However, if $d = 64K$, it is too EXPENSIVE!**

A Clever Idea

- **Note that $n \ll 64K$.**
  - Use $\mathbf{L} = \mathbf{X}\mathbf{X}^{\mathrm{T}}$ instead of $\Sigma = \mathbf{X}^{\mathrm{T}}\mathbf{X}$
- **If $\mathbf{v}$ is eigenvector of $\mathbf{L}$, then $\mathbf{X}^{\mathrm{T}}\mathbf{v}$ is eigenvector of $\boldsymbol{\Sigma}$.**

- **Proof:**

$$\mathbf{L}\mathbf{v} = \lambda\mathbf{v}$$

$$\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{v} = \lambda\mathbf{v}$$

$$\mathbf{X}^{\mathrm{T}}(\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{v}) = \mathbf{X}^{\mathrm{T}}(\lambda\mathbf{v})$$

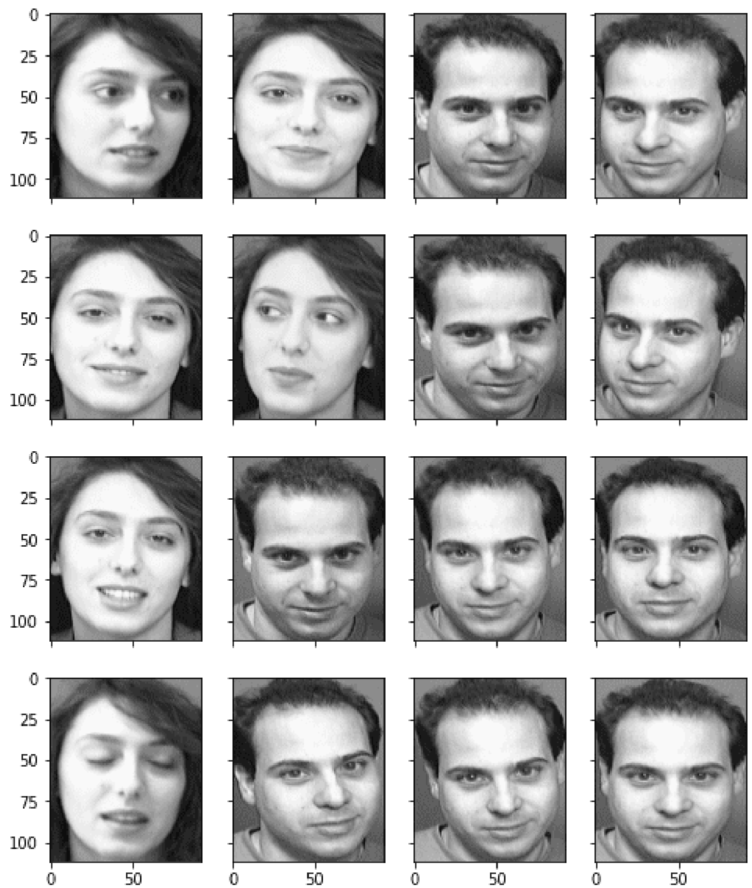
$$(\mathbf{X}^{\mathrm{T}}\mathbf{X})\mathbf{X}^{\mathrm{T}}\mathbf{v} = \lambda(\mathbf{X}^{\mathrm{T}}\mathbf{v})$$

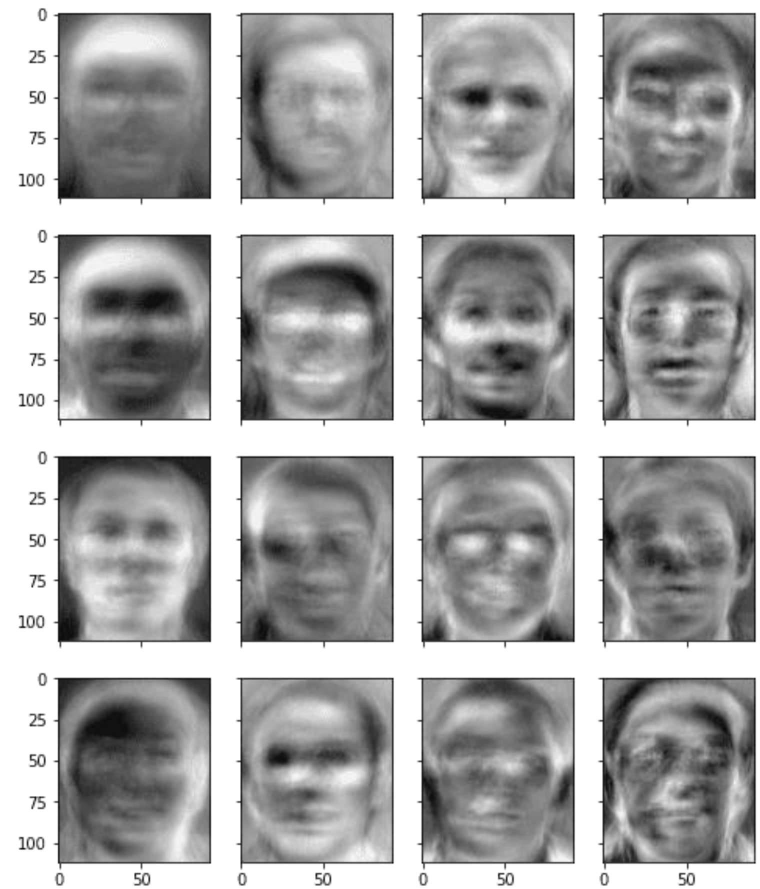
$$\boldsymbol{\Sigma}\mathbf{X}^{\mathrm{T}}\mathbf{v} = \lambda(\mathbf{X}^{\mathrm{T}}\mathbf{v})$$

Instead of computing the eigenvector of $\boldsymbol{\Sigma}$ directly, compute the eigenvector of $\boldsymbol{\Sigma}$ using the eigenvector of $\mathbf{L}$.

Face Recognition with EigenFaces

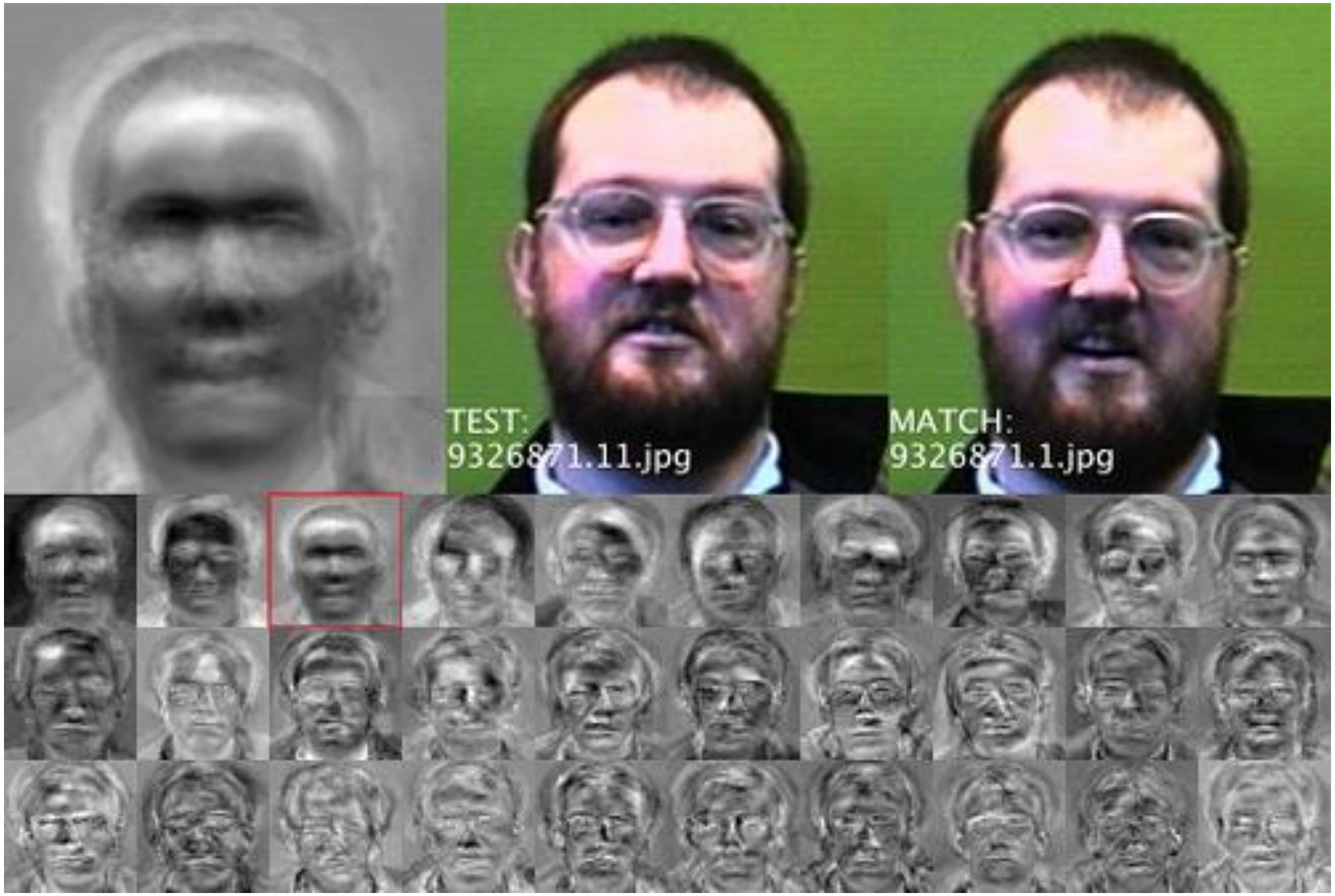
- Using PCA, we represent eigenfaces from original images.





Face Recognition with EigenFaces

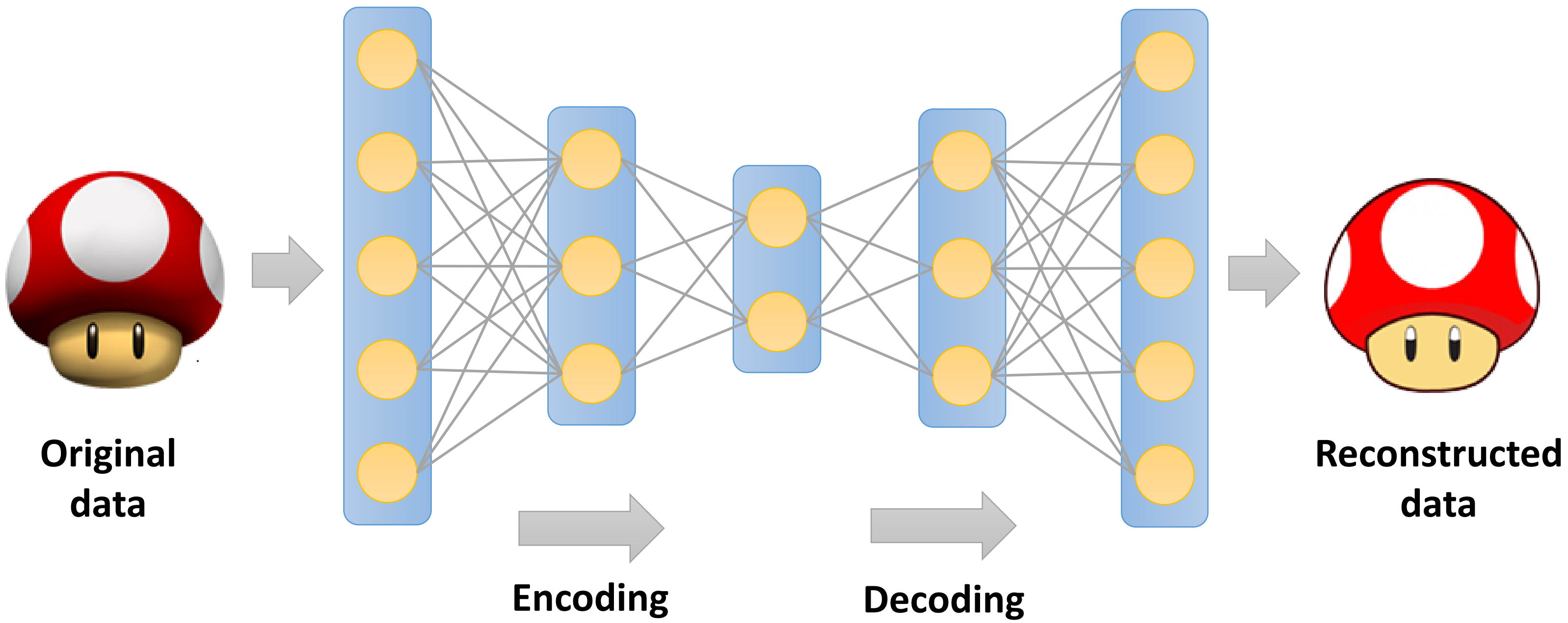
- Finding the most similar face from databases



Autoencoders

What are Autoencoders (AE)?

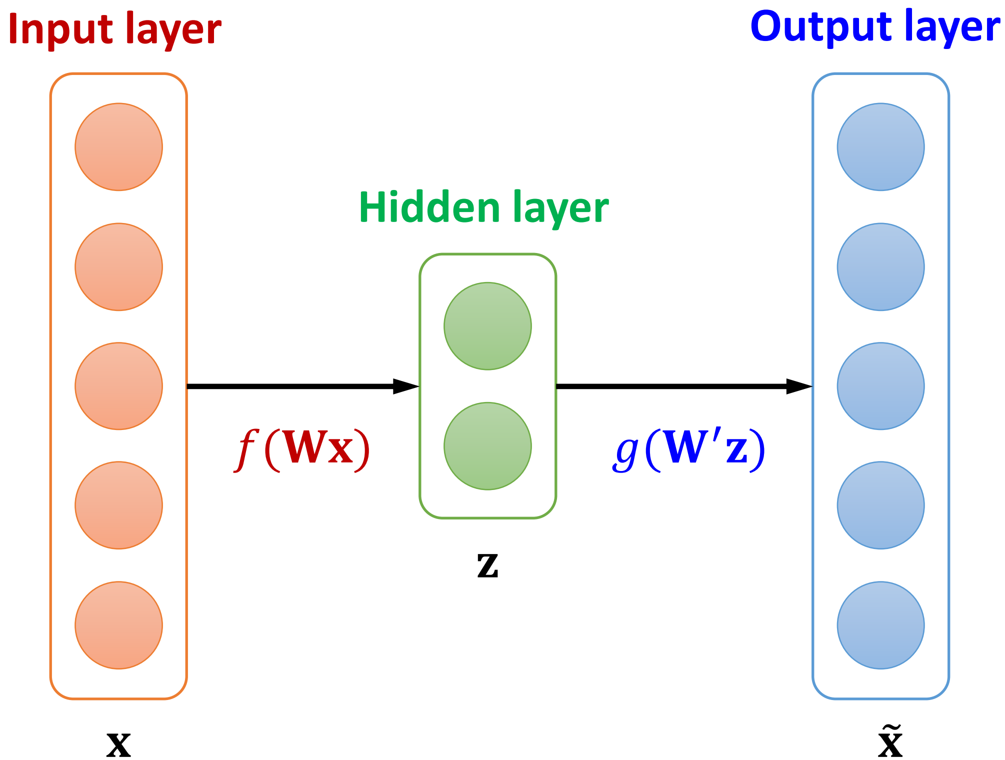
- **It is the feed-forward neural network trained to reconstruct its input at the output layer**
  - It has a **bottleneck layer**.
  - The key purpose of AE is **dimensionality reduction**.
  - This idea is also used for **learning generative data models**.



What are Autoencoders (AE)?

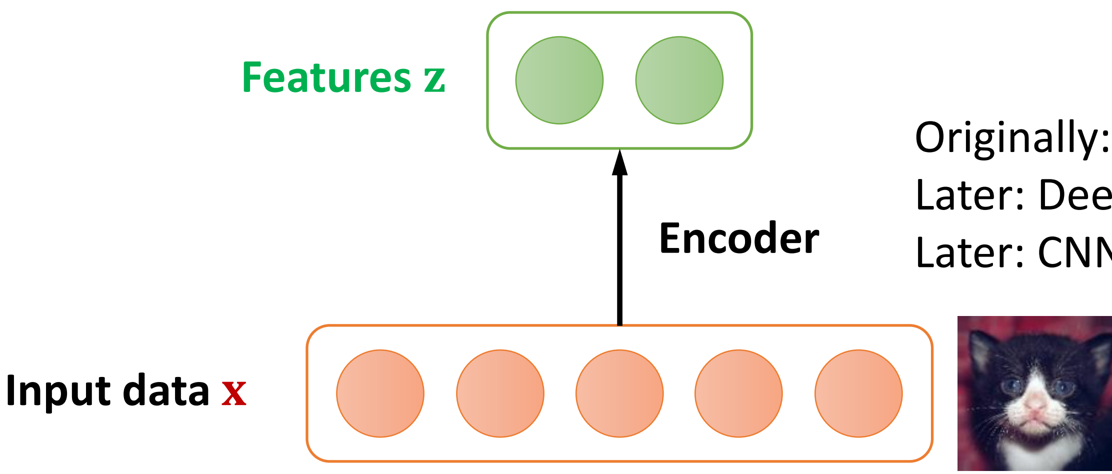
- **Encoder-decoder architecture**
  - **Encoder**: $\mathbf{z} = f(\mathbf{W}\mathbf{x}), \mathbf{W} \in \mathbb{R}^{k \times d}$
  - **Decoder**: $\hat{\mathbf{x}} = g(\mathbf{W}'\mathbf{z}), \mathbf{W}' \in \mathbb{R}^{d \times k}$
  - Often, use the tied weight, i.e., $\mathbf{W}^{\mathrm{T}} = \mathbf{W}'$.

$$\hat{\mathbf{x}} = f\big(g(\mathbf{x})\big)$$



Encoders in AE

- **Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data**

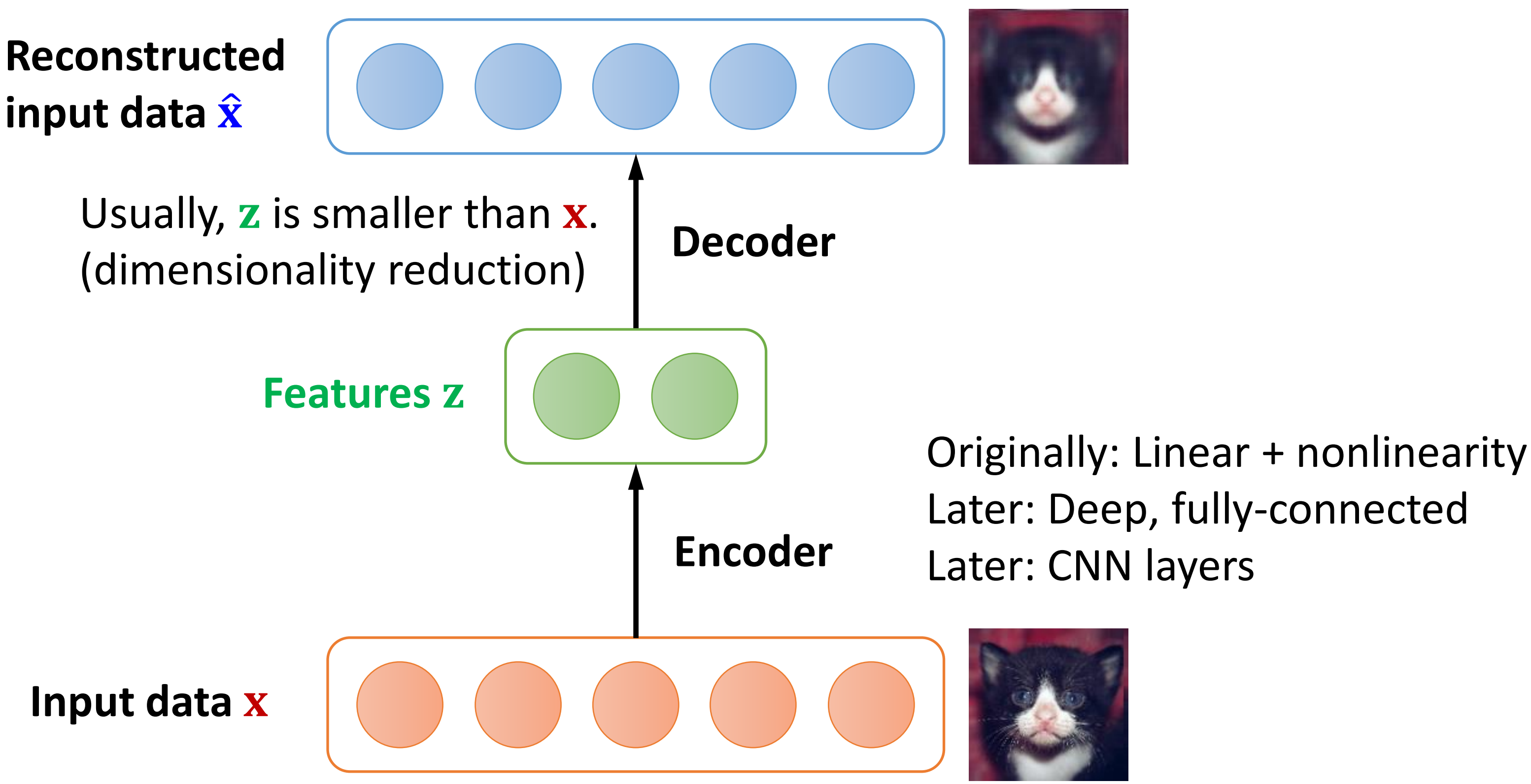


Originally: Linear + nonlinearity
Later: Deep, fully-connected
Later: CNN layers

Decoders in AE

➢ How to learn feature representation?

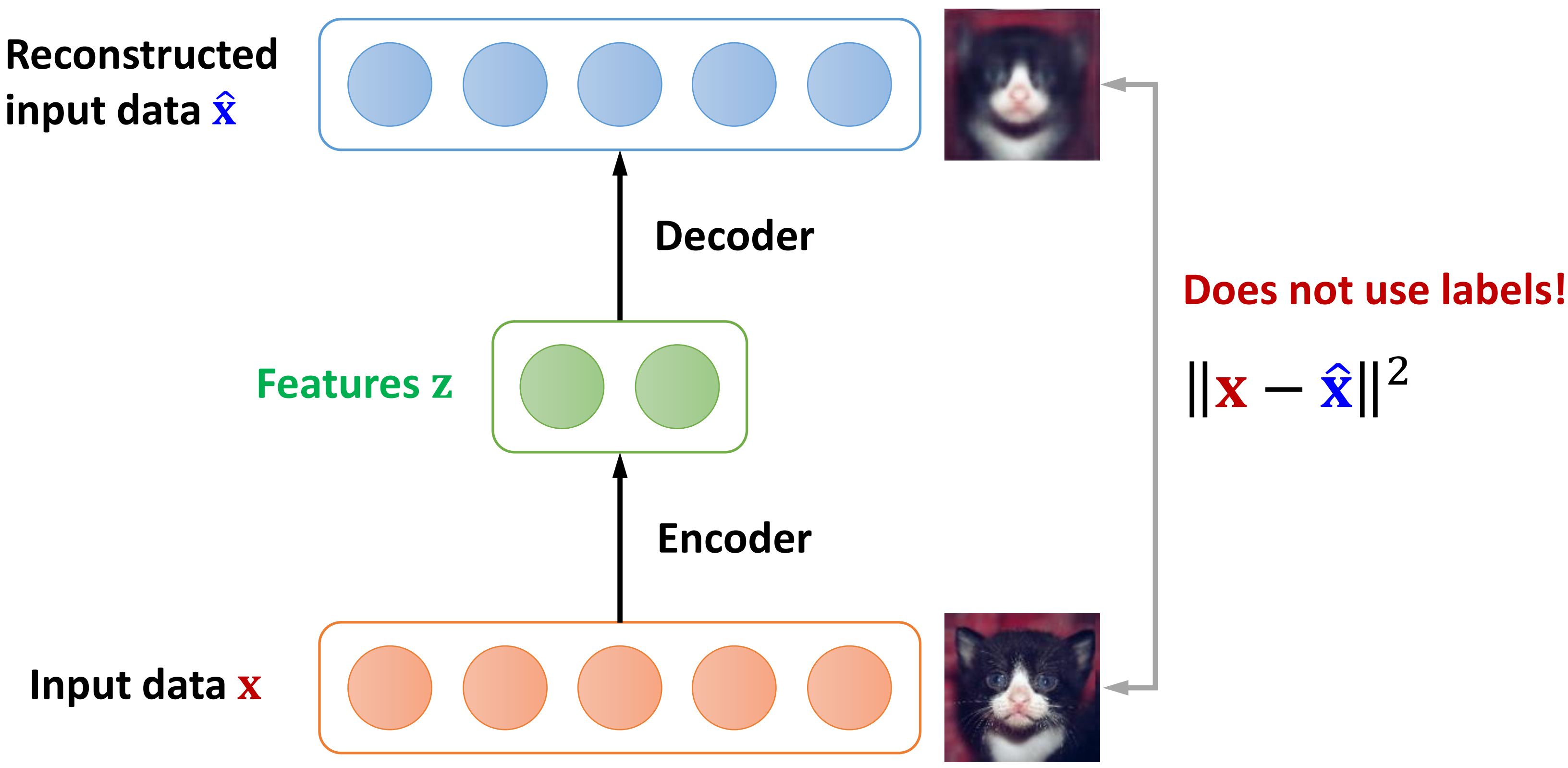
- Train such that features can be used to **reconstruct** original data.
- Encoding itself, i.e., "**autoencoding**"



Feature Representation

➢ **How to learn feature representation?**

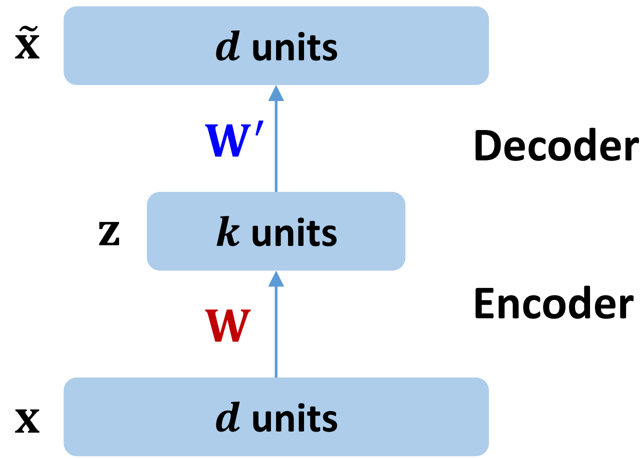
- Train such that features can be used to **reconstruct** original data.
- Encoding itself, i.e., "**autoencoding**"



Linear Autoencoders

- **The simplest autoencoder has one hidden layer, linear activations, and squared error loss.**
  - This network computes $\tilde{\mathbf{x}} = \mathbf{W}'\mathbf{W}\mathbf{x}$, which is a linear function.
  - If $k \geq d$, we can choose $\mathbf{W}$ and $\mathbf{W}'$ such that $\mathbf{W}'\mathbf{W}$ is the identity matrix. This is not useful.

$$\mathcal{L}(\mathbf{x}, \tilde{\mathbf{x}}) = \|\mathbf{x} - \tilde{\mathbf{x}}\|^2$$

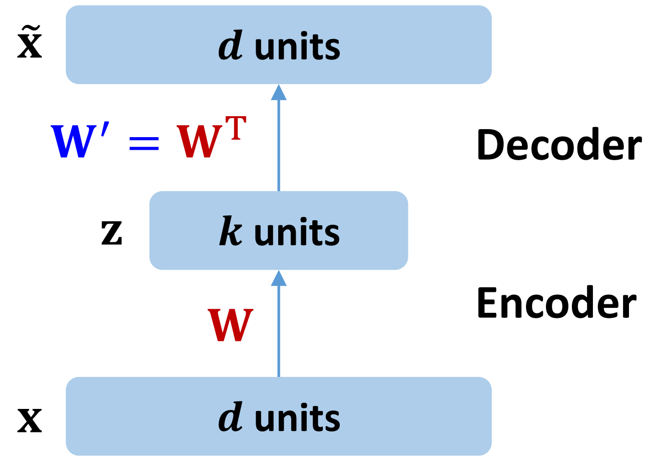


- **When $k < d$, $\mathbf{W}$ maps $\mathbf{x}$ to a k-dimensional space, it makes dimensionality reduction.**

Linear Autoencoders

- **The output must lie in a k-dimensional subspace spanned by the columns of $\mathbf{W}'$.**
  - The best possible k-dimensional subspace is the PCA subspace regarding the reconstruction error.
- **The autoencoder can achieve this by setting $\mathbf{W}' = \mathbf{W}^{\mathrm{T}}$.**
- **The optimal weights for a linear autoencoder are principal components!**

$$\min_{\mathbf{W}} \left\| \mathbf{X} - \mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{X} \right\|^2$$



Example of Linear Autoencoders

➢ **Original image**

https://www.kaggle.com/code/xvivancos/image-compression-using-pca

Example of Linear Autoencoders

Image reconstruction using 2 principal components

Image reconstruction using 30 principal components

Image reconstruction using 200 principal components

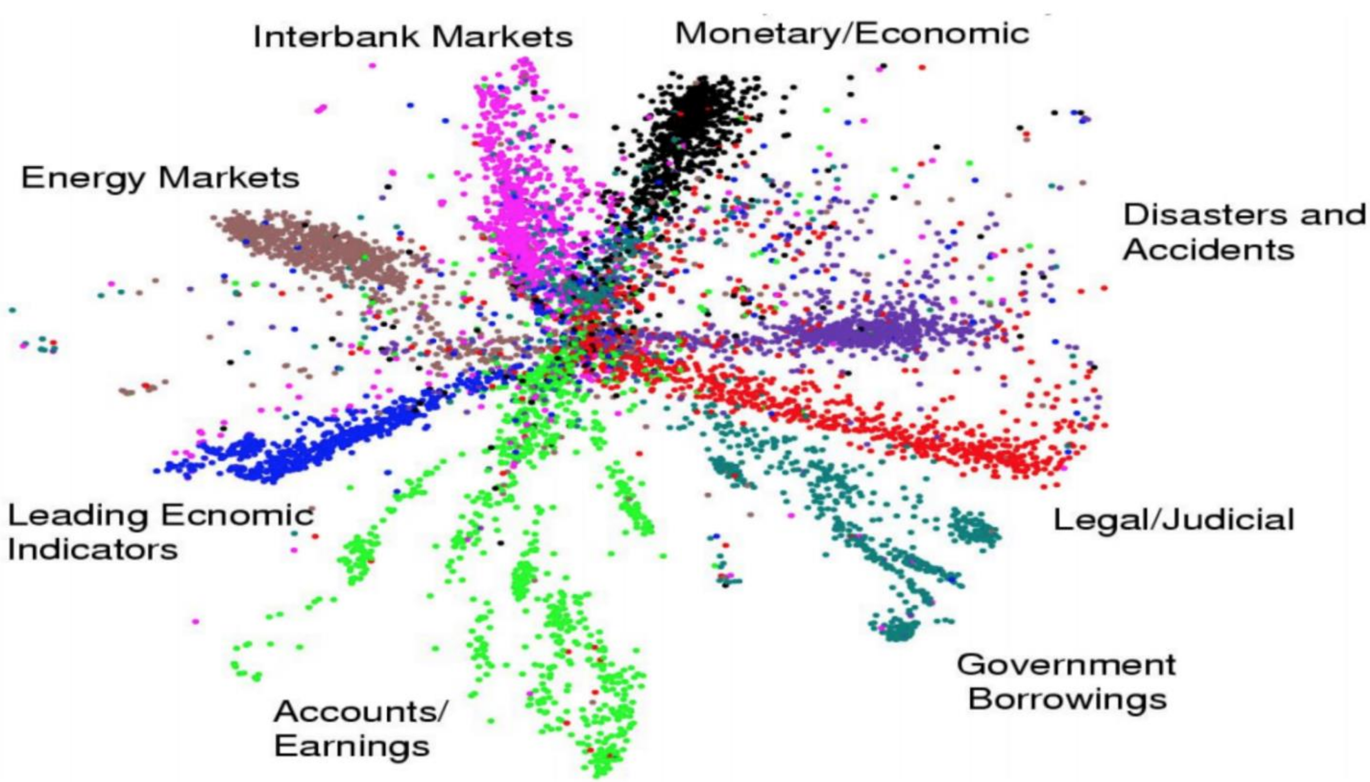
Image reconstruction using 300 principal components

Nonlinear Autoencoders

- **Deep nonlinear autoencoders learn to project the data not onto a subspace but a nonlinear manifold.**
  - This manifold is the image of the decoder.
- **This is called nonlinear dimensionality reduction.**
- **Nonlinear autoencoders can learn more powerful hidden vectors compared with linear autoencoders (PCA).**

Example of Nonlinear Autoencoders

- We use a two-dimensional autoencoder representation of newsgroup articles.
  - Color-coded by topic without giving the labels

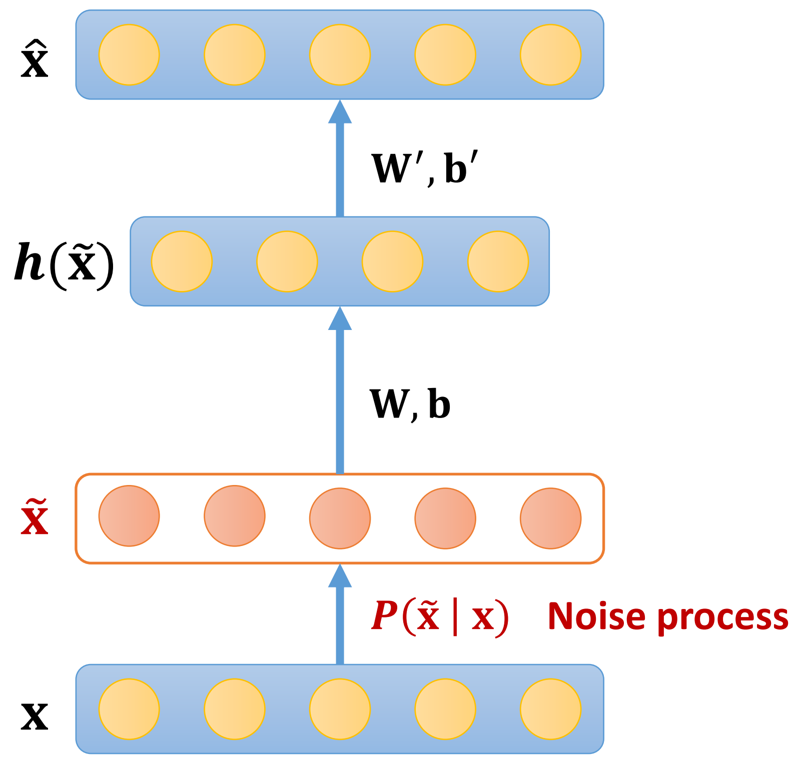


Denoising Autoencoders (DAE)

- **Idea: Representation should be robust to noises.**
  - Random assignment of subset of inputs to 0 with a probability
  - Similar to dropouts on the input layer

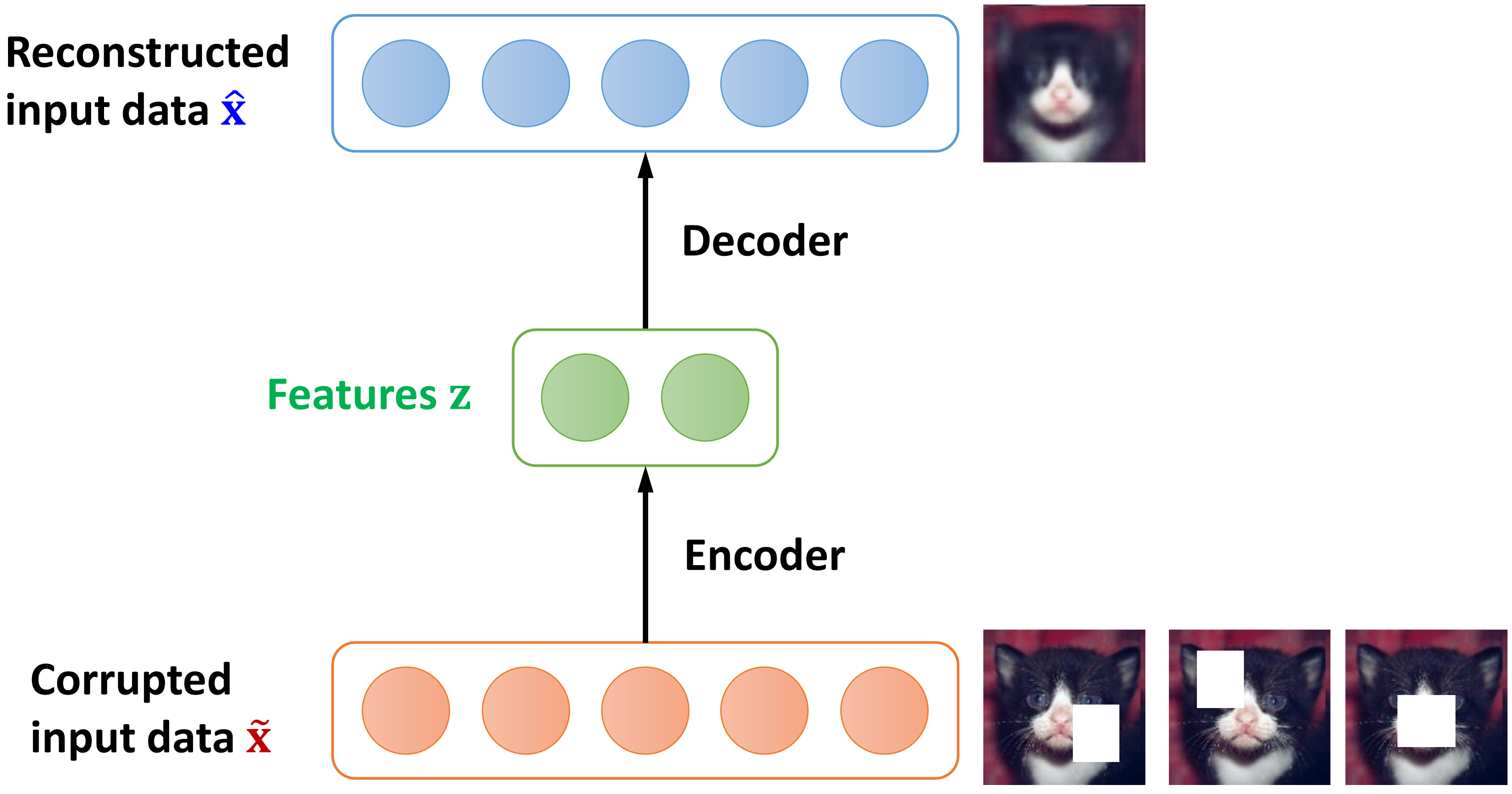
- **Reconstruct $\hat{\mathbf{x}}$ from corrupted input $\tilde{\mathbf{x}}$**

- **Loss function compares reconstructed input $\hat{\mathbf{x}}$ with the noiseless input $\mathbf{x}$**



Denoising Autoencoders (DAE)

- **Add some noise to the input $\tilde{\mathbf{x}} = \mathbf{x} + \epsilon$.**
  - Ask the autoencoder to reconstruct the original input.
  - Learn more generalized representation.



Properties of Autoencoders

- **Autoencoders are used for visualization**

- **Autoencoders are used for data compression.**
  - It is similar to PCA.
  - In fact, linear autoencoders learn the same subspace as PCA.

- **Autoencoders can be used for pre-training**
  - No need for labeled data

Q&A