



# 10-301/10-601 Introduction to Machine Learning

Machine Learning Department  
School of Computer Science  
Carnegie Mellon University

## Principal Component Analysis (PCA)

Matt Gormley

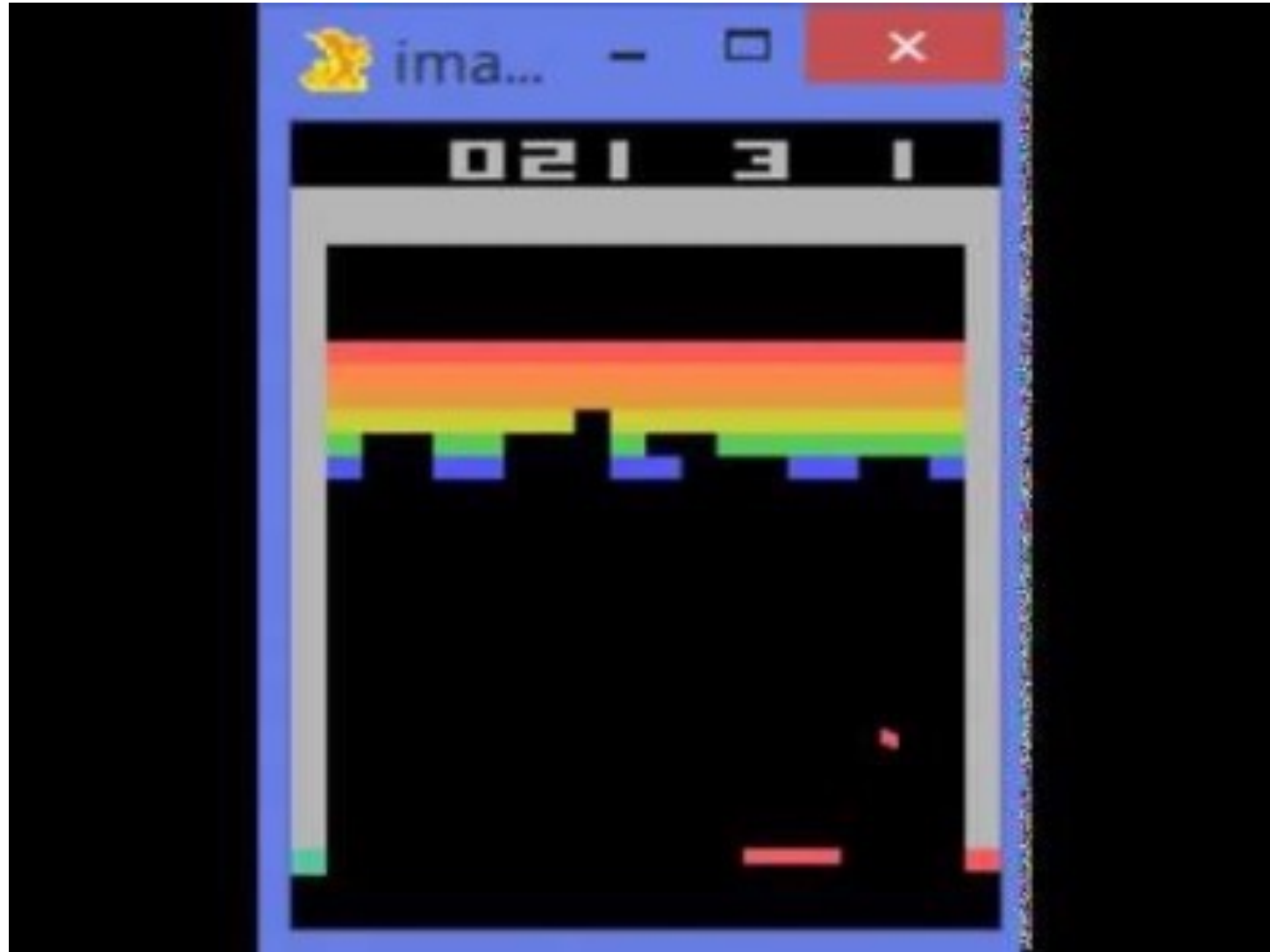
Lecture 23

Nov. 20, 2023

# Reminders

- **Homework 7: Transformer in PyTorch**
  - **Out: Sat, Nov. 11**
  - **Due: Mon, Nov. 20 at 11:59pm**
- **Homework 8: Reinforcement Learning**
  - **Out: Mon, Nov. 20**
  - **Due: Fri, Dec. 1 at 11:59pm**

# Playing Atari games with Deep RL



Source: [https://www.youtube.com/watch?v=V1eYniJ0Rnk&t=2s&ab\\_channel=TwoMinutePapers](https://www.youtube.com/watch?v=V1eYniJ0Rnk&t=2s&ab_channel=TwoMinutePapers)

# **DIMENSIONALITY REDUCTION**

# High Dimension Data

Examples of high dimensional data:

- High resolution images (millions of pixels)



# High Dimension Data

Examples of high dimensional data:

- Multilingual News Stories  
(vocabulary of hundreds of thousands of words)



# High Dimension Data

Examples of high dimensional data:

- Brain Imaging Data (100s of MBs per scan)

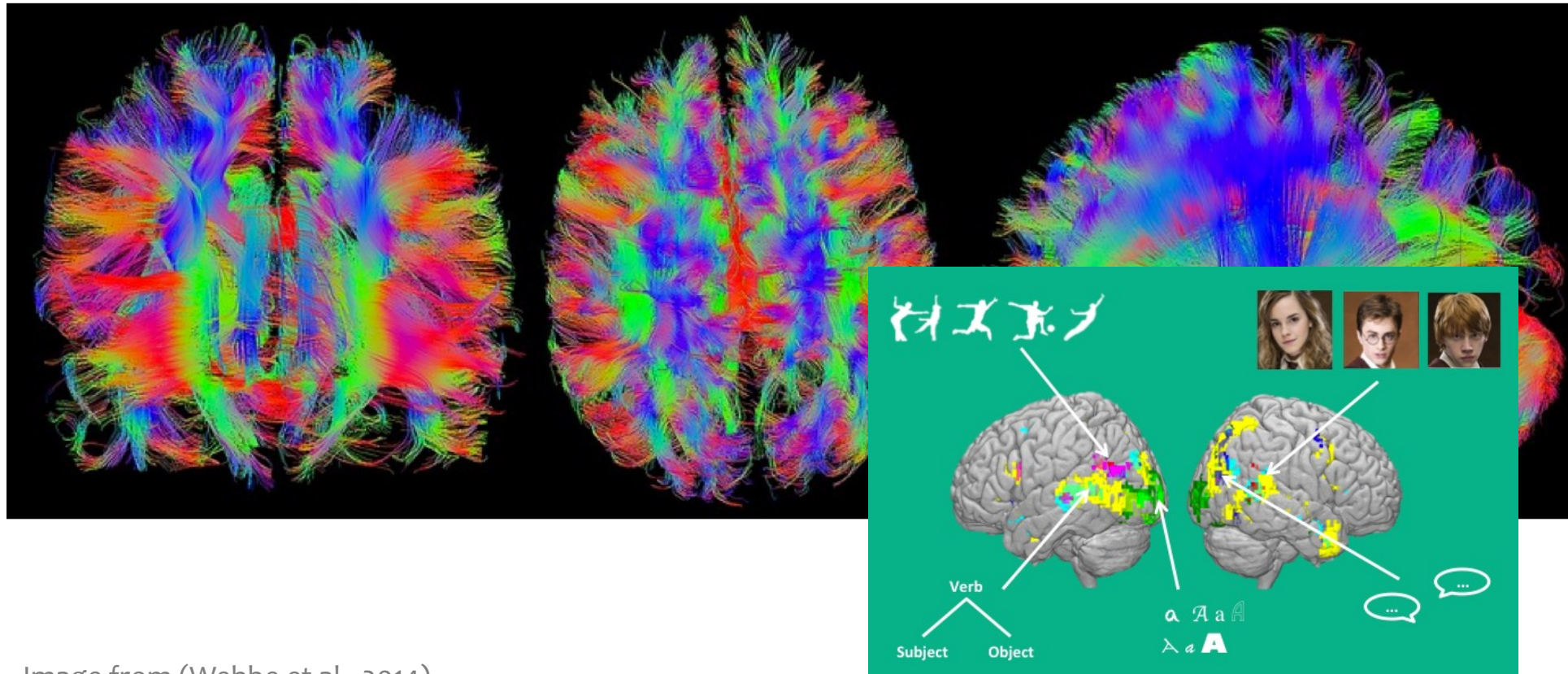


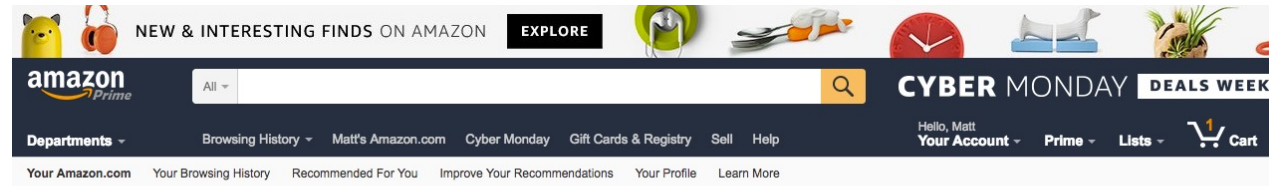
Image from (Wehbe et al., 2014)

Image from <https://pixabay.com/en/brain-mrt-magnetic-resonance-imaging-1728449/>

# High Dimension Data

Examples of high dimensional data:

– Customer Purchase Data



You could be seeing useful stuff here!  
Sign in to get your order status, balances and rewards.

Sign In

Recommended for you, Matt

The image shows a grid of product recommendations categorized into four groups:

- Grocery:** 14 ITEMS. Includes items like Jif peanut butter, maple syrup, and petsh-p.
- Pets:** 6 ITEMS. Includes items like petsh-p, advantage II cat litter, and World's Best cat food.
- Baby Products:** 5 ITEMS. Includes items like Crayola 64 crayons, baby teething rings, and baby teething toys.
- Engineering Books:** 86 ITEMS. Includes the book 'PROBABILISTIC GRAPHICAL MODELS: PRINCIPLES AND TECHNIQUES' by DAPHNE KOLLER AND NIR FRIEDMAN.



# Learning Representations

## Dimensionality Reduction Algorithms:

Powerful (often unsupervised) learning techniques for extracting hidden (potentially lower dimensional) structure from high dimensional datasets.

## Examples:

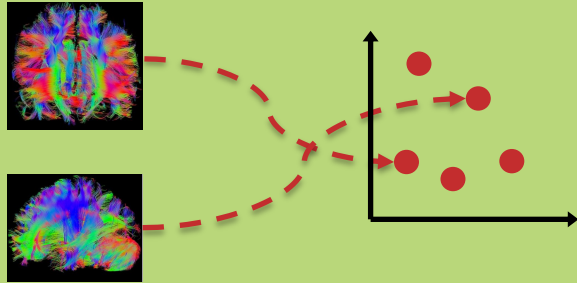
PCA, Kernel PCA, ICA, CCA, t-SNE, Autoencoders, Matrix Factorization

## Useful for:

- Visualization
- More efficient use of resources (e.g., time, memory, communication)
- Statistical: fewer dimensions → better generalization
- Noise removal (improving data quality)

# This section in one slide...

## 1. Dimensionality reduction:



## 2. Random Projection:

① Randomly sample matrix  $V \in \mathbb{R}^{K \times M}$   
② Project down:  $\vec{u}^{(i)} = V \vec{x}^{(i)}$

## 3. Definition of PCA:

Choose the matrix  $V$  that either...

1. minimizes reconstruction error
2. consists of the  $K$  eigenvectors with largest eigenvalue

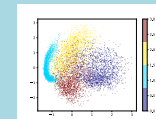
The above are equivalent definitions.

## 4. Algorithm for PCA:

*The option we'll focus on:*

Run Singular Value Decomposition (SVD) to obtain all the eigenvectors. Keep just the top- $K$  to form  $V$ . Play some tricks to keep things efficient.

## 5. An Example



# **DIMENSIONALITY REDUCTION BY RANDOM PROJECTION**

# Random Projection

Example: 2D to 1D

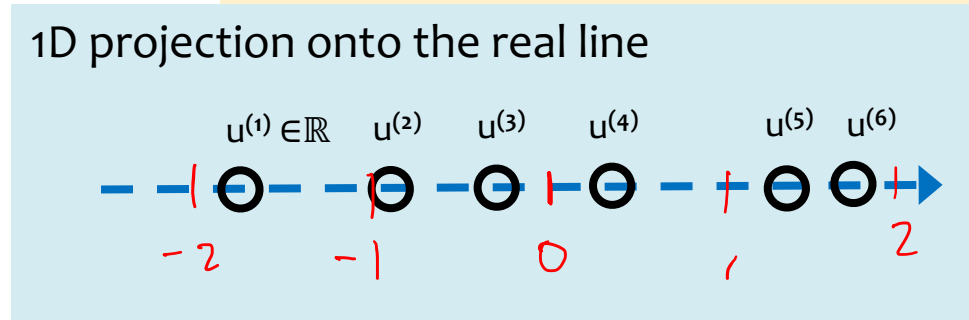
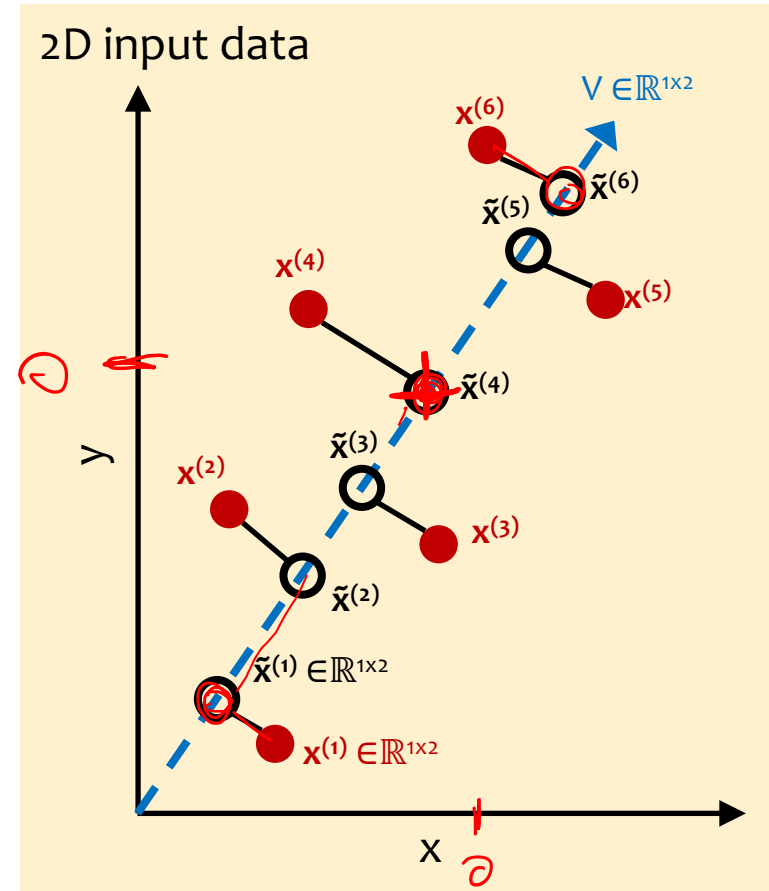
Goal: project from  $M$ -dimensions down to  $K$ -dimensions

Data:

$$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N \text{ where } \mathbf{x}^{(i)} \in \mathbb{R}^M$$

Algorithm:

1. Randomly sample matrix:  $\mathbf{V} \in \mathbb{R}^{K \times M}$   
 $V_{km} \sim \text{Gaussian}(0, 1)$
2. Project down:  $\mathbf{u}^{(i)} = \mathbf{V} \mathbf{x}^{(i)}$
3. Project up:  $\tilde{\mathbf{x}}^{(i)} = \mathbf{V}^T \mathbf{u}^{(i)} = \mathbf{V}^T (\mathbf{V} \mathbf{x}^{(i)})$



# Random Projection

Example: 2D to 1D

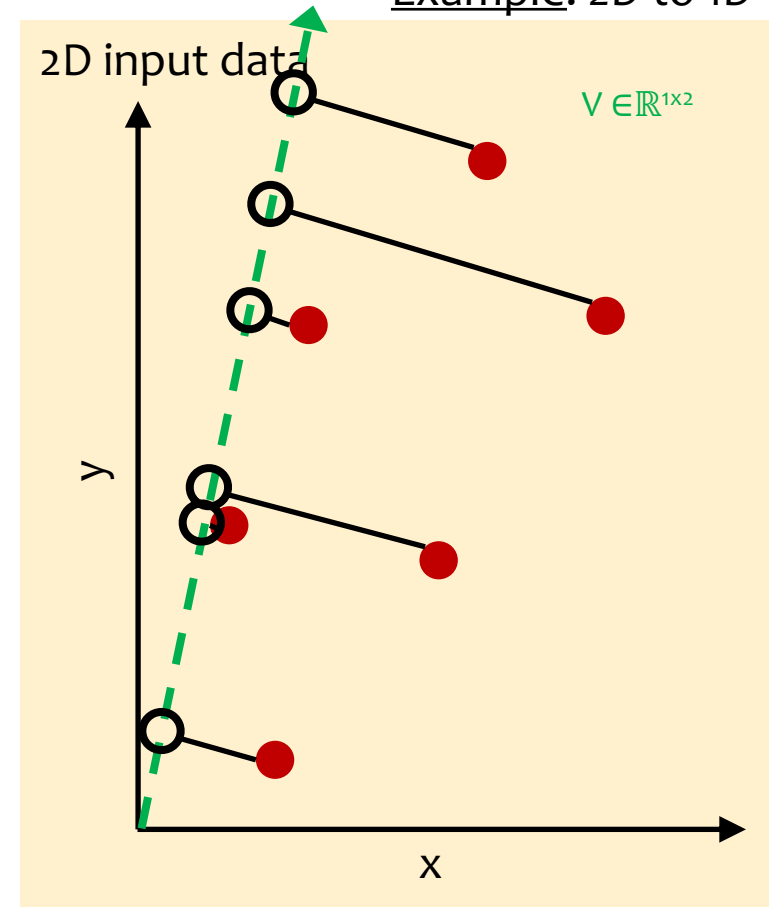
Goal: project from  $M$ -dimensions down to  $K$ -dimensions

Data:

$$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N \text{ where } \mathbf{x}^{(i)} \in \mathbb{R}^M$$

Algorithm:

1. Randomly sample matrix:  $\mathbf{V} \in \mathbb{R}^{K \times M}$   
 $V_{km} \sim \text{Gaussian}(0, 1)$
2. Project down:  $\underbrace{\mathbf{u}^{(i)}}_{K \times 1} = \underbrace{\mathbf{V}}_{K \times M} \underbrace{\mathbf{x}^{(i)}}_{M \times 1}$
3. Project up:  $\underbrace{\mathbf{x}^{(i)}}_{M \times 1} = \underbrace{\mathbf{V}^T}_{M \times K} \underbrace{\mathbf{u}^{(i)}}_{K \times 1} = \mathbf{V}^T (\mathbf{V} \mathbf{x}^{(i)})$



**Problem:** a random projection might give us a poor low dimensional representation of the data

# Johnson-Lindenstrauss Lemma

**Q:** But how could we ever hope to preserve any useful information by randomly projecting into a low-dimensional space?

**A:** Even random projection enjoys some surprisingly impressive properties. In fact, a standard of the J-L lemma starts by assuming we have a random linear projection obtained by sampling each matrix entry from a Gaussian(0,1).



## *An Elementary Proof of a Theorem of Johnson and Lindenstrauss*

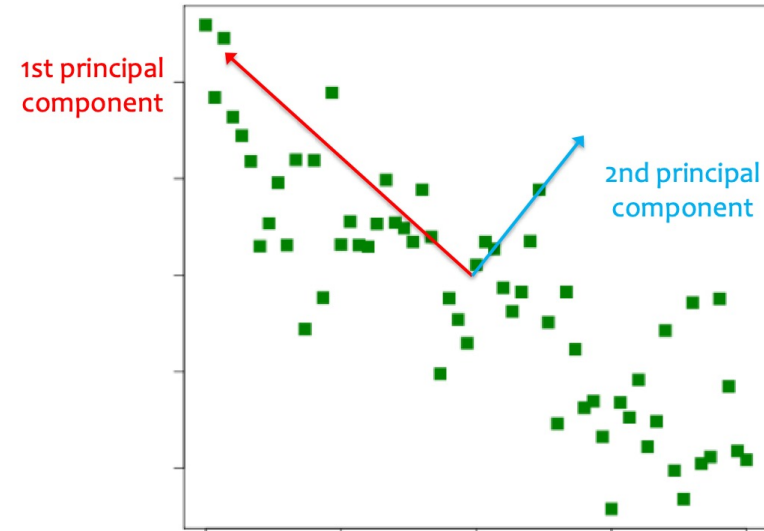
Sanjoy Dasgupta,<sup>1</sup> Anupam Gupta<sup>2</sup>

**ABSTRACT:** A result of Johnson and Lindenstrauss [13] shows that a set of  $n$  points in high dimensional Euclidean space can be mapped into an  $O(\log n/\epsilon^2)$ -dimensional Euclidean space such that the distance between any two points changes by only a factor of  $(1 \pm \epsilon)$ . In this note, we prove this theorem using elementary probabilistic techniques. © 2003 Wiley Periodicals, Inc. *Random Struct. Alg.*, 22: 60–65, 2002

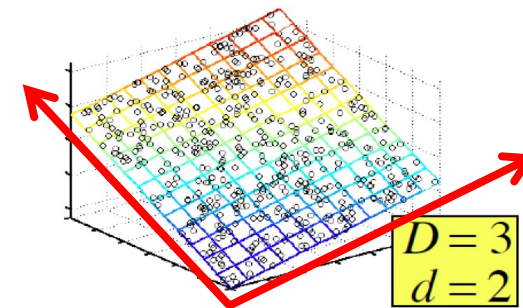
# **DEFINITION OF PRINCIPAL COMPONENT ANALYSIS (PCA)**

# Principal Component Analysis (PCA)

- **Assumption:** the data lies on a low  $K$ -dimensional linear subspace
- **Goal:** identify the axes of that subspace, and project each point onto hyperplane
- **Algorithm:** find the  $K$  eigenvectors with largest eigenvalue using classic matrix decomposition tools



PCA Example: 2D Gaussian Data





# Data for PCA

$$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$$

$$\mathbf{x}^{(i)} \in \mathbb{R}^M$$

$$\mathbf{X} = \begin{bmatrix} (\mathbf{x}^{(1)})^T \\ (\mathbf{x}^{(2)})^T \\ \vdots \\ (\mathbf{x}^{(N)})^T \end{bmatrix}$$

We assume the data is **centered**,  
i.e. the **sample mean** is zero

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

**Q:** What if  
your data is  
**not** centered?

**A:** Subtract off the sample mean

$$\tilde{\mathbf{x}}^{(i)} = \mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}}, \forall i$$

# Background: Sample Variance

Suppose we have a sequence of random samples  $\{x^{(1)}, \dots, x^{(N)}\}$  from a random variable  $X$ .

The (biased) **sample variance**  $\hat{\sigma}^2$  is given by:

$$\hat{\sigma}^2 = \frac{1}{N} \sum_{i=1}^N (x^{(i)} - \hat{\mu})^2$$

where  $\hat{\mu}$  is the sample mean.

# Sample Covariance Matrix

The **sample covariance matrix**  $\Sigma \in \mathbb{R}^{M \times M}$  is given by:

$$\Sigma_{jk} = \frac{1}{N} \sum_{i=1}^N (\underline{x_j^{(i)}} - \underline{\mu_j})(\underline{x_k^{(i)}} - \underline{\mu_k})$$

$$\Sigma_{jj} = \frac{1}{N} \sum_{i=1}^N (x_j^{(i)} - \mu_j)^2 = \text{sample var. } J_j^{\text{th}} \text{ feat}$$

Since the data matrix is centered, we rewrite as:

$$\Sigma = \frac{1}{N} \mathbf{X}^T \mathbf{X}$$

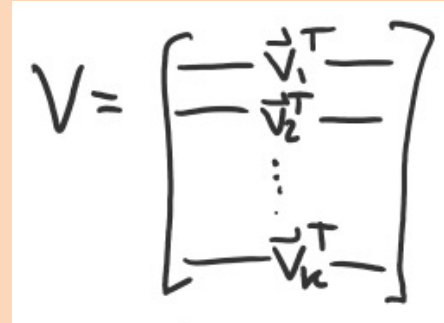
$$\mathbf{X} = \begin{bmatrix} (\mathbf{x}^{(1)})^T \\ (\mathbf{x}^{(2)})^T \\ \vdots \\ (\mathbf{x}^{(N)})^T \end{bmatrix}$$

# Principal Component Analysis (PCA)

## Linear Projection:

Given  $K \times M$  matrix  $\mathbf{V}$ , and  $M \times 1$  vector  $\mathbf{x}^{(i)}$  we obtain the  $K \times 1$  projection  $\mathbf{u}^{(i)}$  by:

$$\mathbf{u}^{(i)} = \mathbf{V} \mathbf{x}^{(i)}$$


$$\mathbf{V} = \begin{bmatrix} \vdots & \vec{v}_1^T & \vdots \\ \vdots & \vec{v}_2^T & \vdots \\ \vdots & \vdots & \vdots \\ \vdots & \vec{v}_k^T & \vdots \end{bmatrix}$$

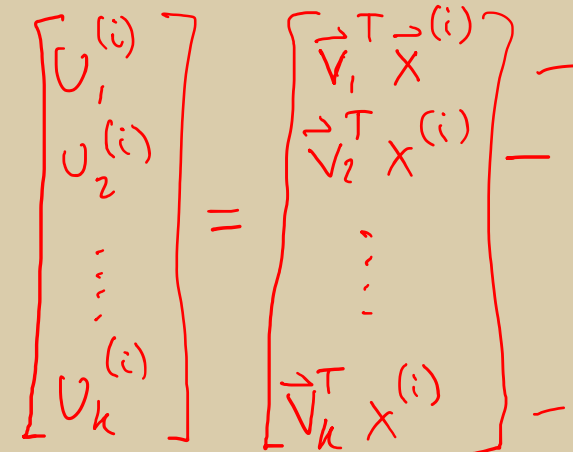
## Definition of PCA:

PCA repeatedly chooses a next vector  $\mathbf{v}_j$  that **minimizes the reconstruction error** s.t.  $\mathbf{v}_j$  is orthogonal to  $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{j-1}$ .

Vector  $\mathbf{v}_j$  is called the  **$j$ th principal component**.

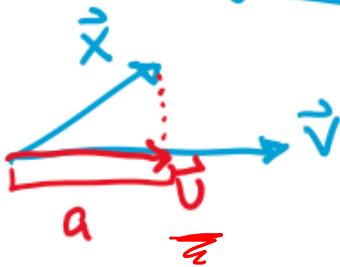
Notice: Two vectors  $\mathbf{a}$  and  $\mathbf{b}$  are **orthogonal** if  $\mathbf{a}^T \mathbf{b} = 0$ .

→ the  $K$ -dimensions in PCA are uncorrelated


$$\begin{bmatrix} u_1^{(i)} \\ u_2^{(i)} \\ \vdots \\ u_k^{(i)} \end{bmatrix} = \begin{bmatrix} \vec{v}_1^T \mathbf{x}^{(i)} \\ \vec{v}_2^T \mathbf{x}^{(i)} \\ \vdots \\ \vec{v}_k^T \mathbf{x}^{(i)} \end{bmatrix} \begin{bmatrix} - \\ - \\ - \\ - \end{bmatrix}$$

# Vector Projection

Recall: Projection



length of projection of  $\vec{x}$  onto  $\vec{v}$

$$a = \frac{\vec{v}^T \vec{x}}{\|\vec{v}\|_2} \quad \text{if } \|\vec{v}\|_2 = 1 \\ \text{otherwise}$$

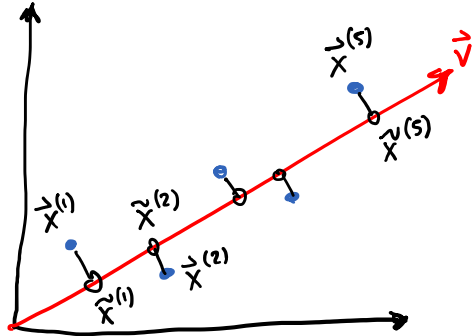
projection of  $\vec{x}$  onto  $\vec{v}$

$$\vec{u} = a\vec{v} = \frac{(\vec{v}^T \vec{x})}{\|\vec{v}\|_2^2} \vec{v} \quad \text{if } \|\vec{v}\|_2 = 1 \\ \text{otherwise}$$

Q: What is the first principal component for PCA?

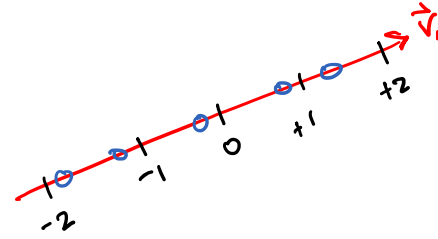
# Objectives for PCA

## Minimize the Reconstruction Error



$$\begin{aligned} \vec{v}_1 &= \operatorname{argmin}_{\vec{v}} \frac{1}{N} \sum_{i=1}^N \text{distance}(\vec{x}^{(i)}, \hat{x}^{(i)})^2 \\ &= \operatorname{argmin}_{\vec{v}} \frac{1}{N} \sum_i \left\| \vec{x}^{(i)} - \left( \frac{\text{Vector projection of } \vec{x}^{(i)} \text{ onto } \vec{v}}{\|\vec{v}\|_2} \right) \right\|_2^2 \\ &= \operatorname{argmin}_{\vec{v}} \frac{1}{N} \sum_{i=1}^N \left\| \vec{x}^{(i)} - \frac{(\vec{v}^T \vec{x}^{(i)}) \vec{v}}{\|\vec{v}\|_2^2} \right\|_2^2 \end{aligned}$$

## Maximize the Variance



$$\begin{aligned} v_1 &= \operatorname{argmax}_{\vec{v}} \frac{1}{N} \sum_{i=1}^N (\text{length of vector proj. of } \vec{x}^{(i)} \text{ onto } \vec{v})^2 \\ &= \operatorname{argmax}_{\vec{v}} \frac{1}{N} \sum_{i=1}^N (\vec{v}^T \vec{x}^{(i)})^2 \\ &\quad \text{s.t. } \|\vec{v}\|_2 = 1 \\ &= \operatorname{argmax}_{\vec{v}} \frac{1}{N} (\vec{v}^T \sum_i \vec{x}^{(i)} \vec{x}^{(i)T} \vec{v}) \\ &\quad \text{s.t. } \|\vec{v}\|_2 = 1 \\ &= \operatorname{argmax}_{\vec{v}} \frac{1}{N} \vec{v}^T \sum_i \vec{v} \end{aligned}$$

$\Sigma = \frac{1}{N} X^T X$

# Projection Example

## Question:

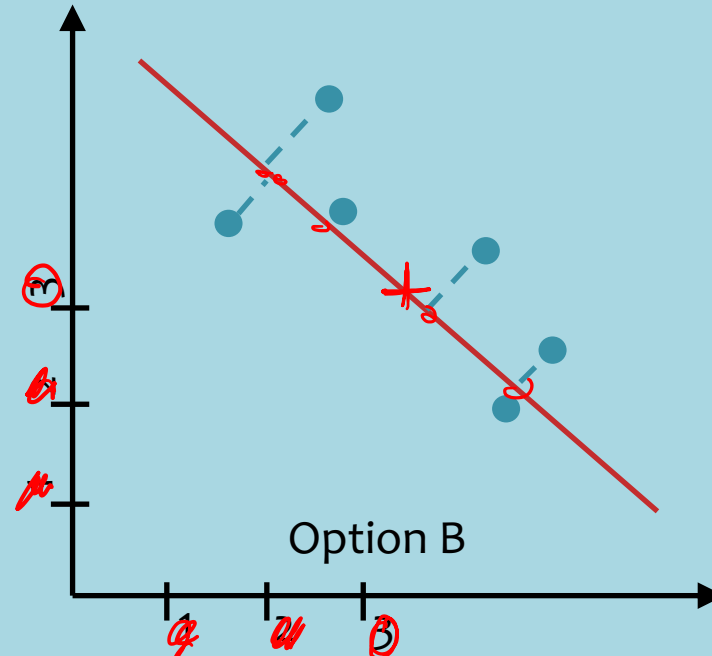
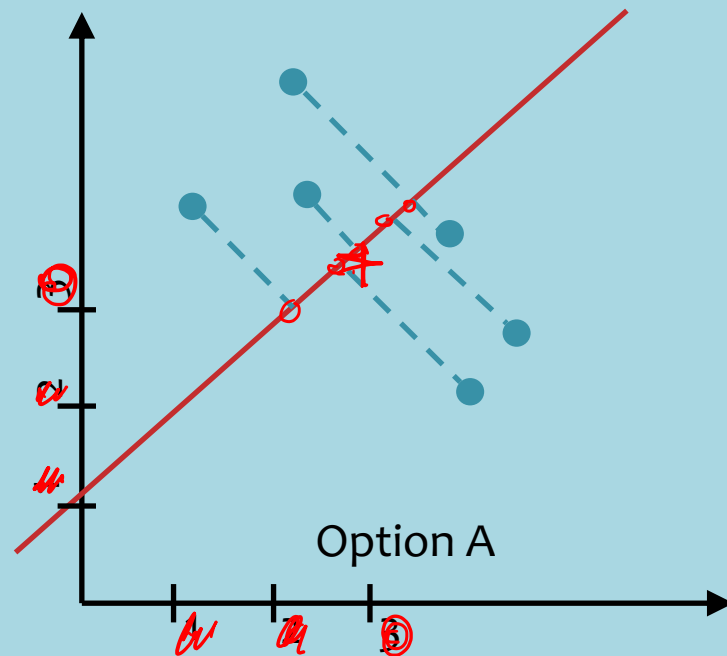
Below are two plots of the same dataset D. Consider the two projections shown.

Q1 1. Which maximizes the variance?

Q2 2. Which minimizes the reconstruction error?

A	B
51%	49%
15%	85%

## Answer:



C = toxic

# PCA Objective Functions

What is the first principal component  $\mathbf{v}_1$  chosen by PCA?

Option 1: The vector that *minimizes* the **reconstruction error**

$$\mathbf{v}_1 = \operatorname{argmin}_{\mathbf{v}: \|\mathbf{v}\|^2=1} \frac{1}{N} \sum_{i=1}^N \|\mathbf{x}^{(i)} - (\mathbf{v}^T \mathbf{x}^{(i)}) \mathbf{v}\|^2$$

Option 2: The vector that *maximizes* the **variance**

$$\mathbf{v}_1 = \operatorname{argmax}_{\mathbf{v}: \|\mathbf{v}\|^2=1} \frac{1}{N} \sum_{i=1}^N (\mathbf{v}^T \mathbf{x}^{(i)})^2$$



## Equivalence of Maximizing Variance and Minimizing Reconstruction Error

# PCA

**Claim:** Minimizing the reconstruction error is equivalent to maximizing the variance.

**Proof:** First, note that:  $\|a-b\|_2^2 = a^T a - 2a^T b + b^T b$

$$\|\mathbf{x}^{(i)} - (\mathbf{v}^T \mathbf{x}^{(i)})\mathbf{v}\|^2 = \underbrace{\|\mathbf{x}^{(i)}\|^2 - (\mathbf{v}^T \mathbf{x}^{(i)})^2}_{\text{red bracket}} \quad (1)$$

since  $\mathbf{v}^T \mathbf{v} = \|\mathbf{v}\|^2 = 1$ .

Substituting into the minimization problem, and removing the extraneous terms, we obtain the maximization problem.

$$\mathbf{v}^* = \operatorname{argmin}_{\mathbf{v}: \|\mathbf{v}\|^2=1} \frac{1}{N} \sum_{i=1}^N \|\mathbf{x}^{(i)} - (\mathbf{v}^T \mathbf{x}^{(i)})\mathbf{v}\|^2 \quad (2)$$

$$= \operatorname{argmin}_{\mathbf{v}: \|\mathbf{v}\|^2=1} \frac{1}{N} \sum_{i=1}^N \underbrace{\|\mathbf{x}^{(i)}\|^2}_{\text{red bracket}} - (\mathbf{v}^T \mathbf{x}^{(i)})^2 \quad (3)$$

$$= \operatorname{argmax}_{\mathbf{v}: \|\mathbf{v}\|^2=1} \frac{1}{N} \sum_{i=1}^N (\mathbf{v}^T \mathbf{x}^{(i)})^2 \quad (4)$$

# PCA Objective Functions

What is the first principal component  $\mathbf{v}_1$  chosen by PCA?

Option 1: The vector that *minimizes* the **reconstruction error**

$$\mathbf{v}_1 = \underset{\mathbf{v}: \|\mathbf{v}\|^2=1}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^N \|\mathbf{x}^{(i)} - (\mathbf{v}^T \mathbf{x}^{(i)}) \mathbf{v}\|^2$$

Option 2: The vector that *maximizes* the **variance**

$$\mathbf{v}_1 = \underset{\mathbf{v}: \|\mathbf{v}\|^2=1}{\operatorname{argmax}} \frac{1}{N} \sum_{i=1}^N (\mathbf{v}^T \mathbf{x}^{(i)})^2$$

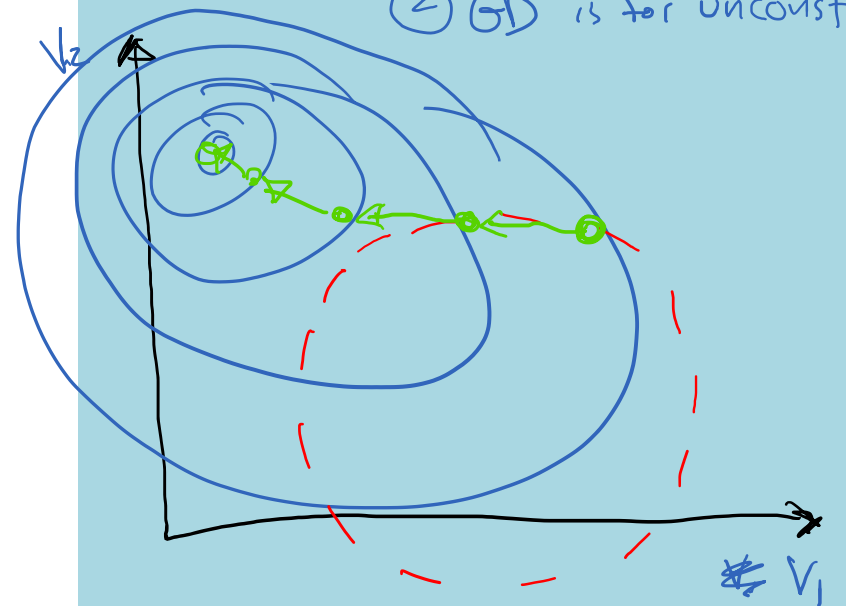
**Question:**

Q3

Why can't we just use gradient descent to find the minimum of the PCA optimization problem?

**Answer:**

- ① non convex
- ② GD is for unconstrained

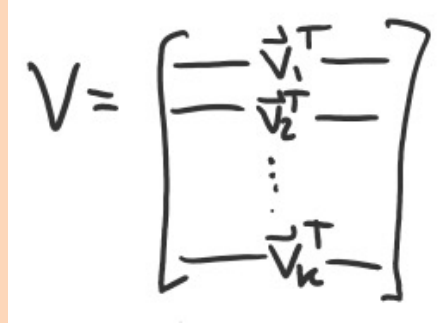


# Principal Component Analysis (PCA)

## Linear Projection:

Given  $K \times M$  matrix  $\mathbf{V}$ , and  $M \times 1$  vector  $\mathbf{x}^{(i)}$  we obtain the  $K \times 1$  projection  $\mathbf{u}^{(i)}$  by:

$$\mathbf{u}^{(i)} = \mathbf{V} \mathbf{x}^{(i)}$$


$$\mathbf{V} = \begin{bmatrix} \text{---} \mathbf{v}_1^T \text{---} \\ \text{---} \mathbf{v}_2^T \text{---} \\ \vdots \\ \text{---} \mathbf{v}_k^T \text{---} \end{bmatrix}$$

## Question:

Why can't we just use gradient descent to find the minimum of the PCA optimization problem?

## Definition of PCA:

PCA repeatedly chooses a next vector  $\mathbf{v}_j$  that **minimizes the reconstruction error** s.t.  $\mathbf{v}_j$  is orthogonal to  $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{j-1}$ .

Vector  $\mathbf{v}_j$  is called the  **$j$ th principal component**.

Notice: Two vectors  $\mathbf{a}$  and  $\mathbf{b}$  are **orthogonal** if  $\mathbf{a}^T \mathbf{b} = 0$ .

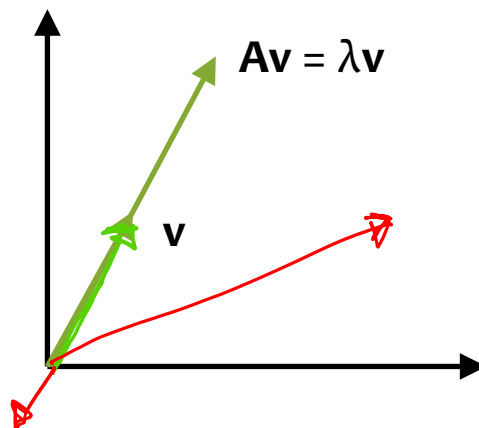
→ the  $K$ -dimensions in PCA are uncorrelated

## Answer:

# Background: Eigenvectors & Eigenvalues

For a square matrix  $\mathbf{A}$  ( $n \times n$  matrix), the vector  $\mathbf{v}$  ( $n \times 1$  matrix) is an **eigenvector** iff there exists **eigenvalue**  $\lambda$  (scalar) such that:

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



The linear transformation  $\mathbf{A}$  is only stretching vector  $\mathbf{v}$ .

That is,  $\lambda\mathbf{v}$  is a *scalar multiple* of  $\mathbf{v}$ .

# Background: Eigenvectors & Eigenvalues

Fact #1: The eigenvectors of a **symmetric matrix** are orthogonal to each other.

Fact #2: The **covariance matrix  $\Sigma$**  is symmetric.

# The First Principal Component

# PCA

**Claim:** The vector that maximizes the variances is the eigenvector of  $\Sigma$  with largest eigenvalue.

**Proof Sketch:** To find the first principal component, we wish to solve the following constrained optimization problem (variance ~~maximization~~ <sup>Max</sup>).

$$\mathbf{v}_1 = \operatorname{argmax}_{\mathbf{v}: \|\mathbf{v}\|^2=1} \mathbf{v}^T \Sigma \mathbf{v} \quad (1)$$

*Handwritten note:  $\mathbf{v}^T \mathbf{v} = 1 \Rightarrow (\mathbf{v}^T \mathbf{v} - 1) = 0$*

So we turn to the method of Lagrange multipliers. The Lagrangian is:

$$\mathcal{L}(\mathbf{v}, \lambda) = \mathbf{v}^T \Sigma \mathbf{v} - \lambda(\mathbf{v}^T \mathbf{v} - 1) \quad (2)$$

Taking the derivative of the Lagrangian and setting to zero gives:

$$\frac{d}{d\mathbf{v}} (\mathbf{v}^T \Sigma \mathbf{v} - \lambda(\mathbf{v}^T \mathbf{v} - 1)) = 0 \quad (3)$$

$$\Sigma \mathbf{v} - \lambda \mathbf{v} = 0 \quad (4)$$

$$\Sigma \mathbf{v} = \lambda \mathbf{v} \quad (5)$$

Recall: For a square matrix  $\mathbf{A}$ , the vector  $\mathbf{v}$  is an **eigenvector** iff there exists **eigenvalue**  $\lambda$  such that:

$$\mathbf{A} \mathbf{v} = \lambda \mathbf{v} \quad (6)$$

Rewriting the objective of the maximization shows that not only will the optimal vector  $\mathbf{v}_1$  be an eigenvector, it will be one with maximal eigenvalue.

$$\mathbf{v}^T \Sigma \mathbf{v} = \mathbf{v}^T \lambda \mathbf{v} \quad (7)$$

$$= \lambda \mathbf{v}^T \mathbf{v} \quad (8)$$

$$= \lambda \|\mathbf{v}\|^2 \quad (9)$$

$$= \lambda \quad (10)$$

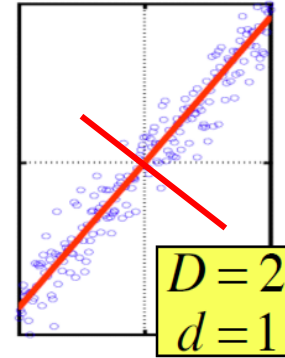
# Principal Component Analysis (PCA)

$(X X^T)v = \lambda v$ , so  $v$  (the first PC) is the eigenvector of sample correlation/covariance matrix  $X X^T$

Sample variance of projection  $v^T X X^T v = \lambda v^T v = \lambda$

$$v^T \Sigma v = \lambda$$

Thus, the eigenvalue  $\lambda$  denotes the amount of variability captured along that dimension (aka amount of energy along that dimension).



Eigenvalues  $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \dots$

- The 1<sup>st</sup> PC  $v_1$  is the the eigenvector of the sample covariance matrix  $X X^T$  associated with the largest eigenvalue
- The 2<sup>nd</sup> PC  $v_2$  is the the eigenvector of the sample covariance matrix  $X X^T$  associated with the second largest eigenvalue
- And so on ...

# **ALGORITHMS FOR PCA**



# Algorithms for PCA

How do we find principal components (i.e. eigenvectors)?

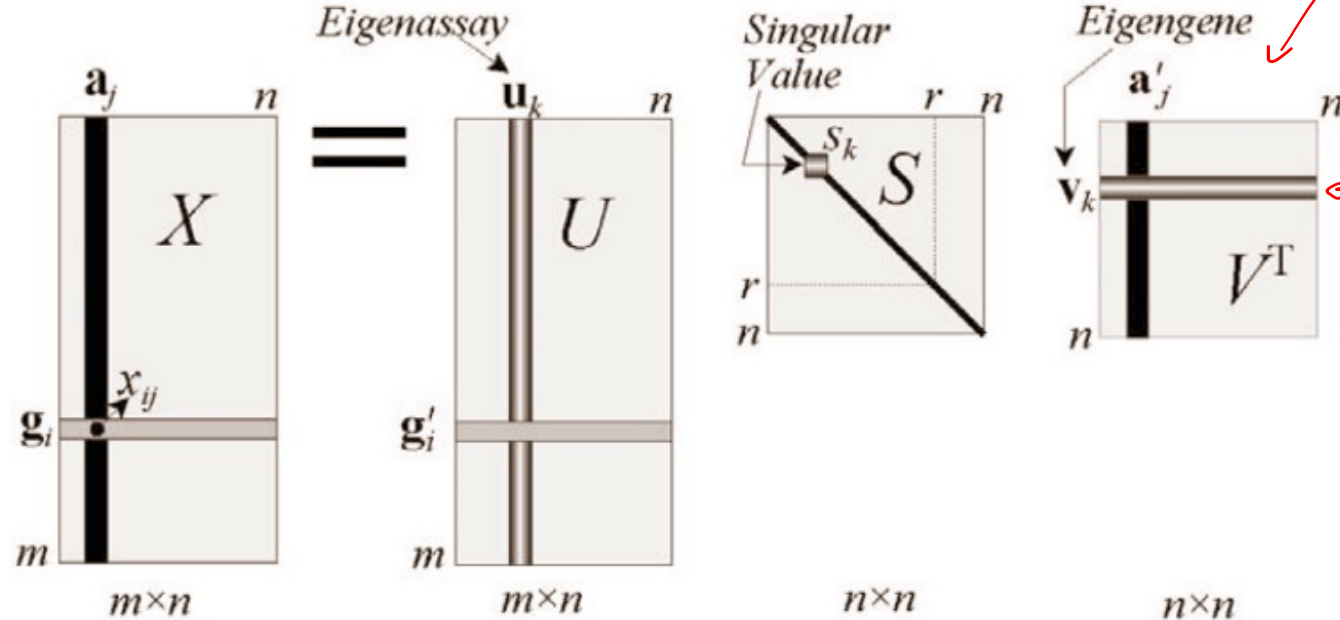
- Power iteration (aka. Von Mises iteration)
  - finds **each** principal component **one at a time** in order
- Singular Value Decomposition (SVD)
  - finds **all** the principal components **at once**
  - two options:
    - Option A: run SVD on  $X^T X = \Sigma$   $M \times M$
    - Option B: run SVD on  $X$   $N \times M$   
(not obvious why Option B should work...)
- Stochastic Methods (approximate)
  - **very efficient** for high dimensional datasets with lots of points

# SVD

# points

# features

$$X = \underline{U} \underline{S} \underline{V}^T$$



# feat x # feat

keep the top-k → Proj. matrix  $V^{1:k}$

Data  $X$ , one row per data point

$US$  gives coordinates of rows of  $X$  in the space of principle components

$S$  is diagonal,  $S_k > S_{k+1}$ ,  $S_k^2$  is  $k$ th largest eigenvalue

Rows of  $V^T$  are unit length eigenvectors of  $X^T X$

If cols of  $X$  have zero mean, then  $X^T X = c \Sigma$  and eigenvects are the Principle Components

# Singular Value Decomposition

To generate principle components:

- Subtract mean  $\bar{x} = \frac{1}{N} \sum_{n=1}^N x^n$  from each data point, to create zero-centered data
- Create matrix  $X$  with one row vector per (zero centered) data point
- Solve SVD:  $X = USV^T$
- Output Principle components: columns of  $V$  (= rows of  $V^T$ )
  - Eigenvectors in  $V$  are sorted from largest to smallest eigenvalues
  - $S$  is diagonal, with  $s_k^2$  giving eigenvalue for  $k$ th eigenvector

# Singular Value Decomposition

To project a point (column vector  $x$ ) into PC coordinates:

$$V^T x$$

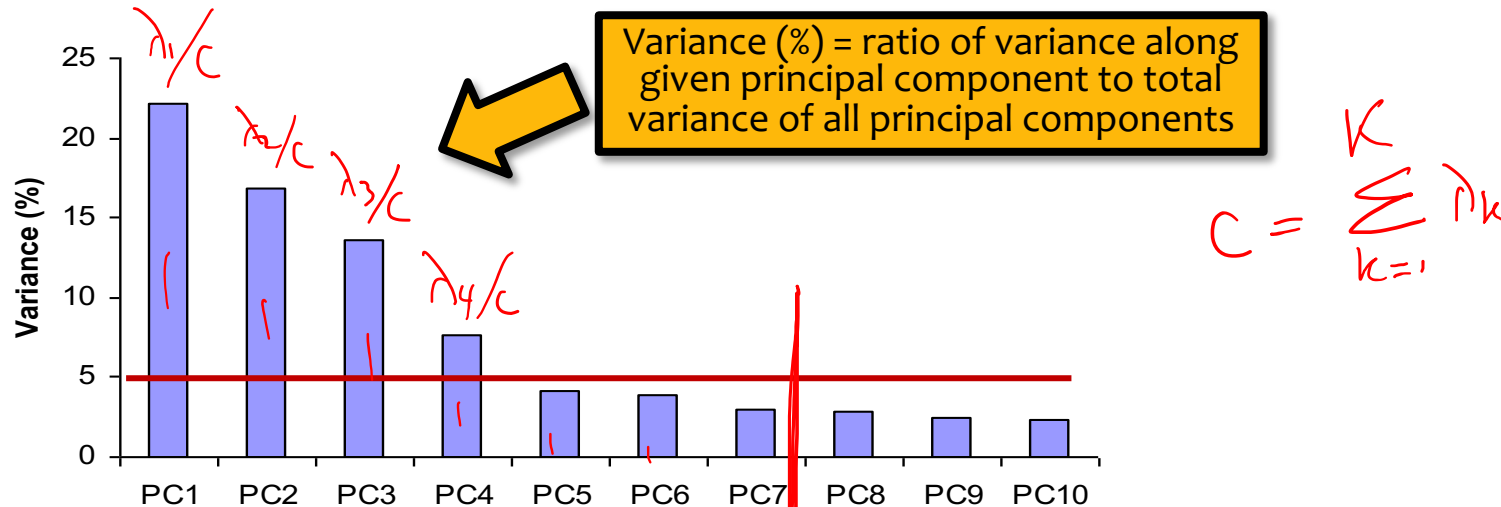
If  $x_i$  is  $i^{\text{th}}$  row of data matrix  $X$ , then

- ( $i^{\text{th}}$  row of  $US$ ) =  $V^T x_i^T$
- $(US)^T = V^T X^T$

To project a column vector  $x$  to  $M$  dim Principle Components subspace, take just the first  $M$  coordinates of  $V^T x$

# How Many PCs?

- For  $M$  original dimensions, sample covariance matrix is  $M \times M$ , and has up to  $M$  eigenvectors. So  $M$  principal components (PCs).
- Where does dimensionality reduction come from?  
Can ignore the components of lesser significance.



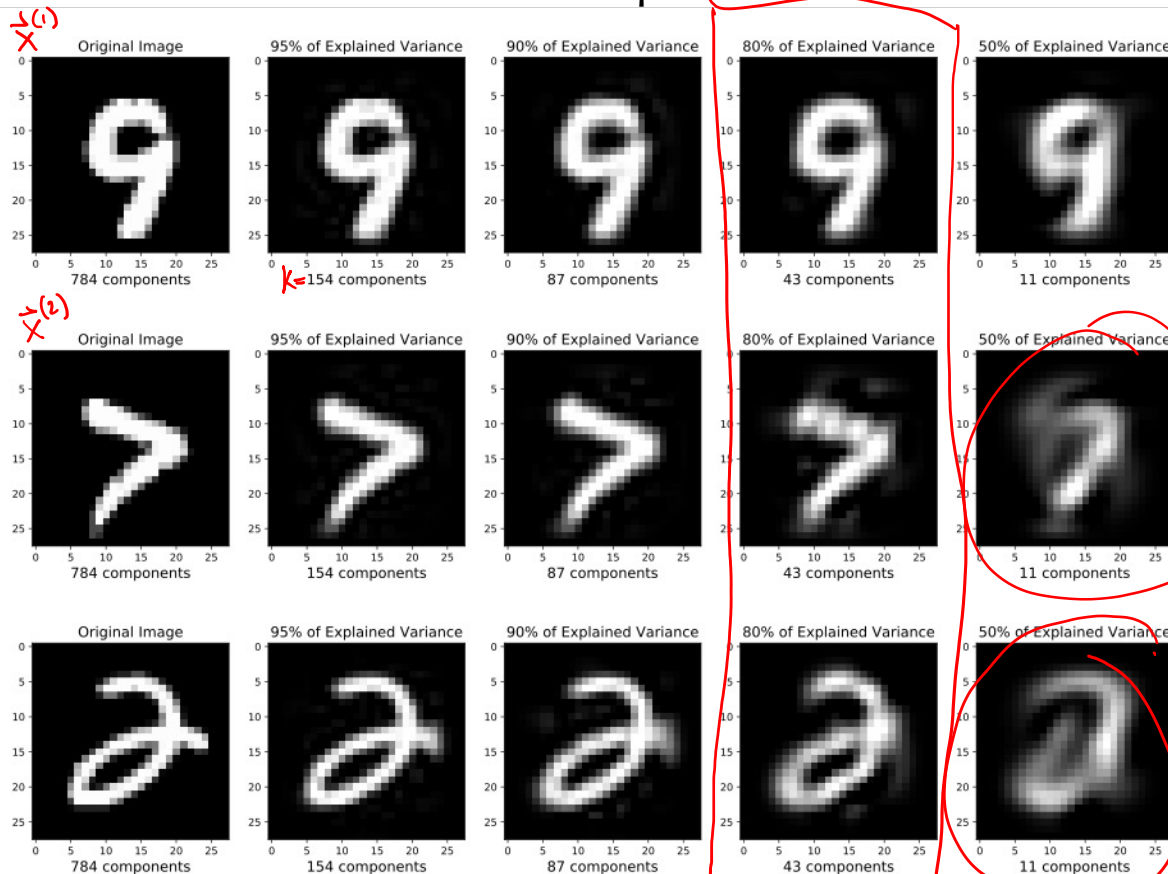
- You do lose some information, but if the eigenvalues are small, you don't lose much
  - $M$  dimensions in original data
  - calculate  $M$  eigenvectors and eigenvalues
  - choose only the first  $D$  eigenvectors, based on their eigenvalues
  - final data set has only  $D$  dimensions

# PCA EXAMPLES

# Projecting MNIST digits

## Task Setting:

1. Take each  $28 \times 28$  image of a digit (i.e. a vector  $\mathbf{x}^{(i)}$  of length 784) and project it down to  $K$  components (i.e. a vector  $\mathbf{u}^{(i)}$ )
2. Report percent of variance explained for  $K$  components
3. Then project back up to  $28 \times 28$  image (i.e. a vector  $\tilde{\mathbf{x}}^{(i)}$  of length 784) to visualize how much information was preserved



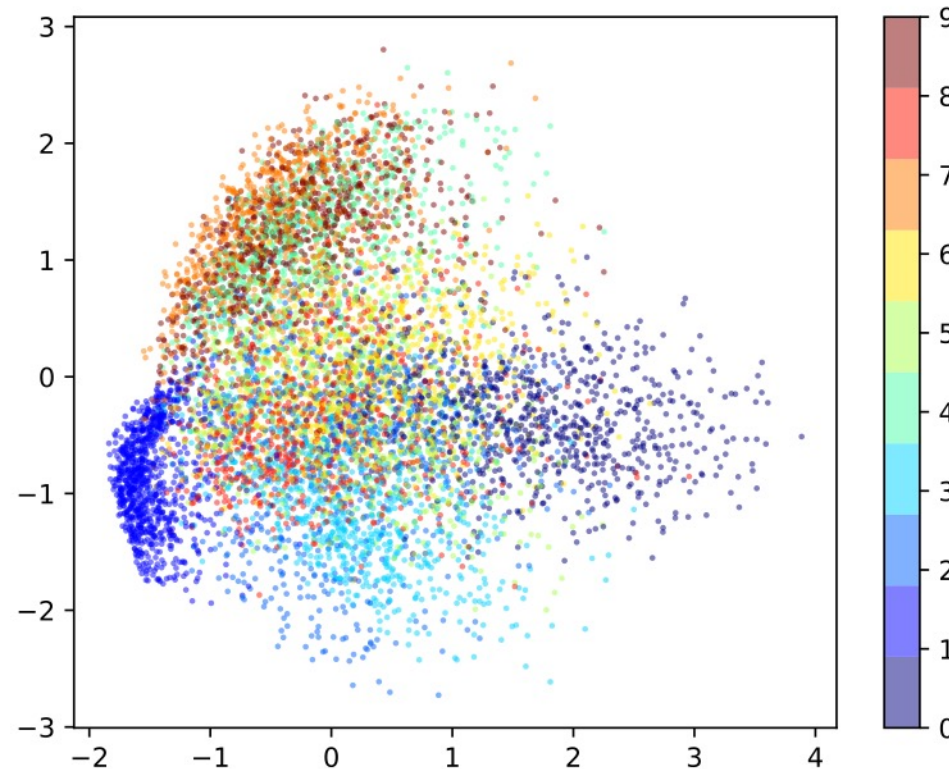
**Takeaway:**  
Using fewer principal components  $K$  leads to higher reconstruction error.

But even a small number (say 43) still preserves a lot of information about the original image.

# Projecting MNIST digits

## Task Setting:

1. Take each 28x28 image of a digit (i.e. a vector  $\mathbf{x}^{(i)}$  of length 784) and project it down to  $K=2$  components (i.e. a vector  $\mathbf{u}^{(i)}$ )
2. Plot the 2 dimensional points  $\mathbf{u}^{(i)}$  and label with the (unknown to PCA) label  $y^{(i)}$  as the color
3. Here we look at all ten digits 0 - 9



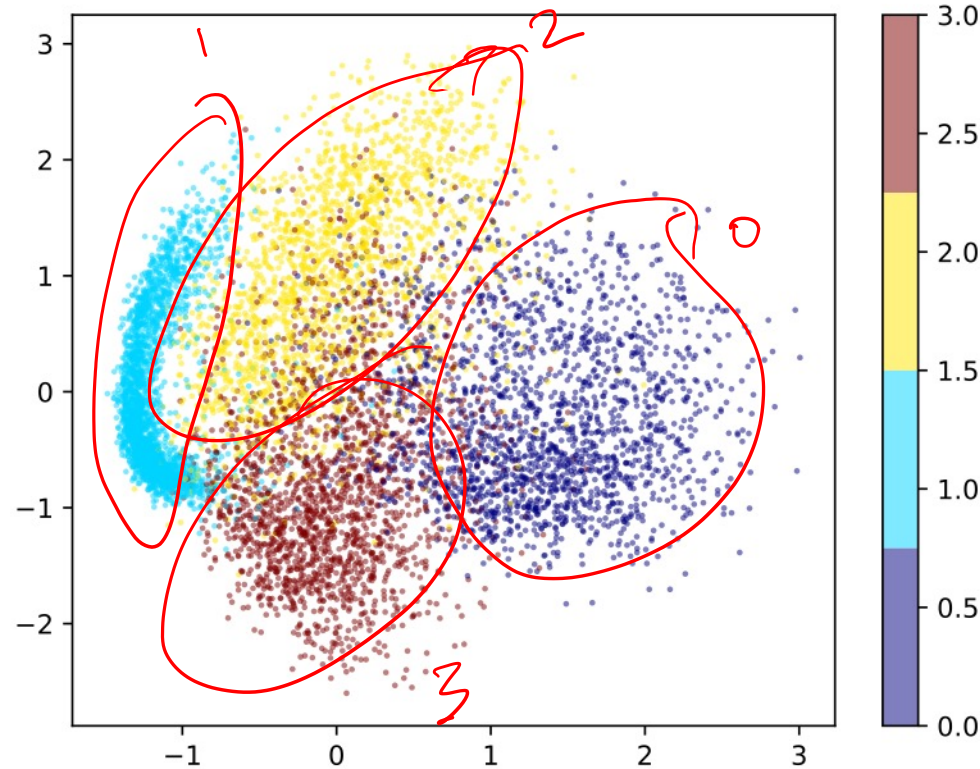
**Takeaway:**  
Even with a tiny number of principal components  $K=2$ , PCA learns a representation that captures the *latent* information about the type of digit



# Projecting MNIST digits

## Task Setting:

1. Take each 28x28 image of a digit (i.e. a vector  $\mathbf{x}^{(i)}$  of length 784) and project it down to  $K=2$  components (i.e. a vector  $\mathbf{u}^{(i)}$ )
2. Plot the 2 dimensional points  $\mathbf{u}^{(i)}$  and label with the (unknown to PCA) label  $y^{(i)}$  as the color
3. Here we look at just four digits 0, 1, 2, 3



**Takeaway:**  
Even with a tiny number of principal components  $K=2$ , PCA learns a representation that captures the *latent* information about the type of digit

# Learning Objectives

## Dimensionality Reduction / PCA

*You should be able to...*

1. Define the sample mean, sample variance, and sample covariance of a vector-valued dataset
2. Identify examples of high dimensional data and common use cases for dimensionality reduction
3. Draw the principal components of a given toy dataset
4. Establish the equivalence of minimization of reconstruction error with maximization of variance
5. Given a set of principal components, project from high to low dimensional space and do the reverse to produce a reconstruction
6. Explain the connection between PCA, eigenvectors, eigenvalues, and covariance matrix
7. Use common methods in linear algebra to obtain the principal components