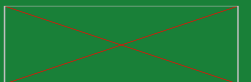


Do not edit
How to change the
design

**Join at slido.com
#3640956**



The Slido app must be installed on every computer you're presenting from



Lecture 3

Dimensionality Reduction

PCA with a review of linear algebra, calculus, and optimization

EECS 189/289, Fall 2026 @ UC Berkeley

Joseph E. Gonzalez and Narges Norouzi

Recap



3640956

In Tuesday's lecture we focused on **Nearest Neighbor** methods:

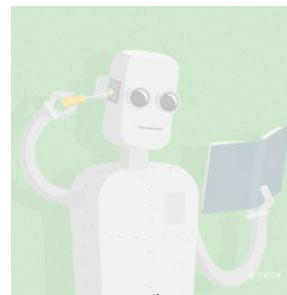
- **Non-parametric Classification and Regression** – use the labels of the k-nearest neighbors to estimate the test point.
- **K-means clustering** – grouping nearby points.

Today we will explore **Principal Component Analysis (PCA)**

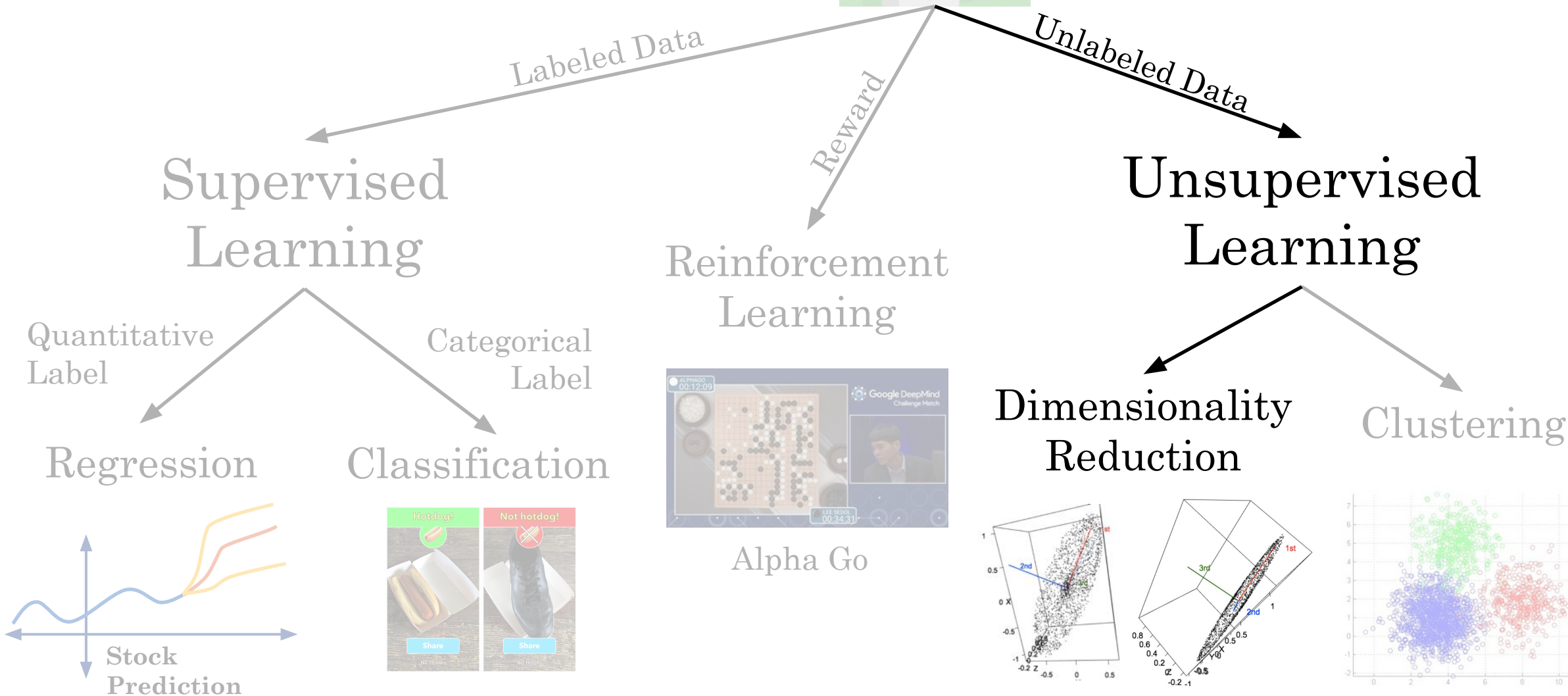
- **Dimensionality reduction** – we will project high-dimensional data into a descriptive low-dimensional space.
- **Parametric method** – there will be a fixed set of parameters

We will review **linear algebra, calculus, and optimization.**

Taxonomy of Machine Learning



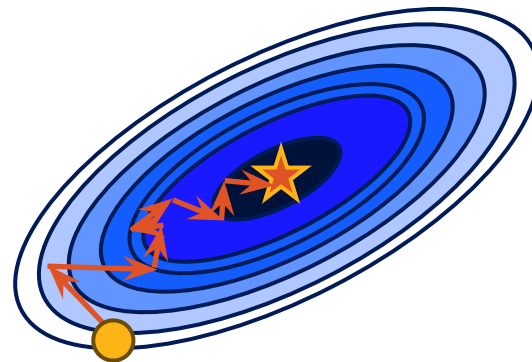
3640956



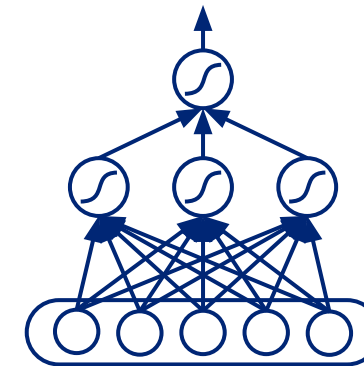
Using PCA and scikit-learn to make sense of congressional votes.

Demo

Algorithm



Model



Application



This Machine Learning Class



3640956

Interpreting the demonstration

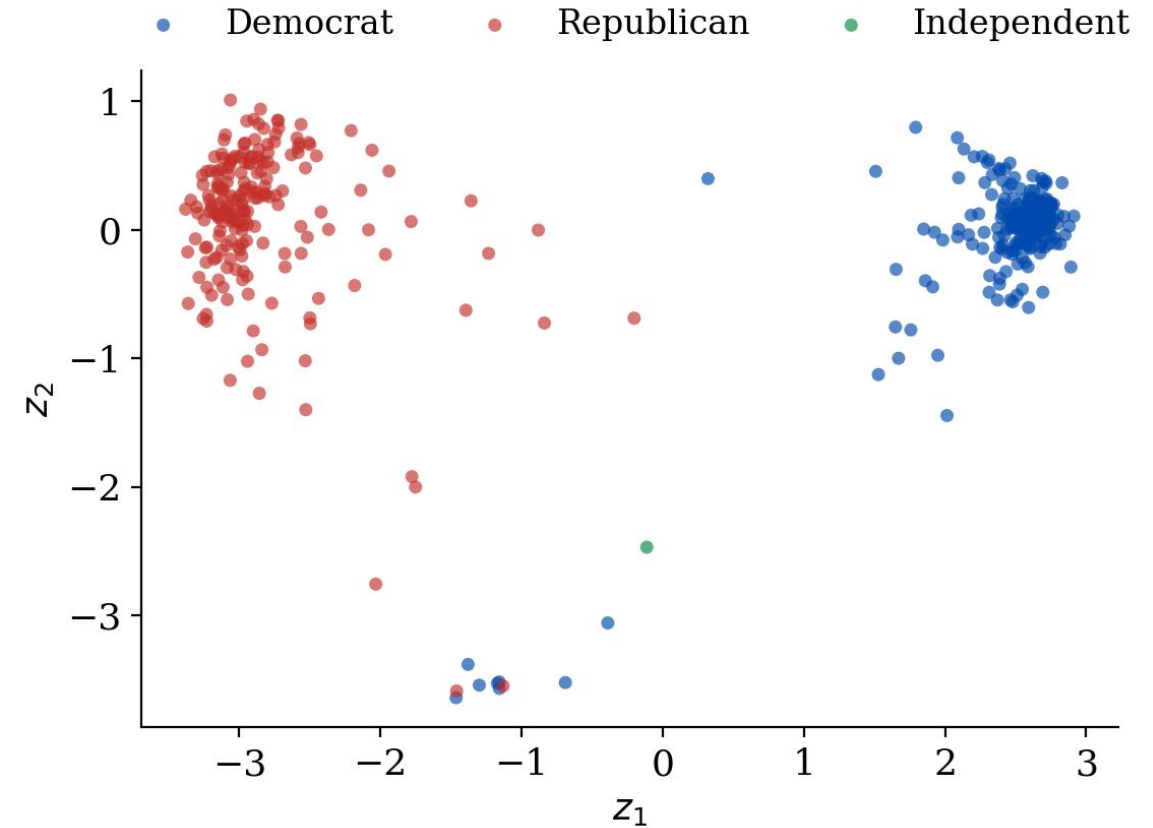
Input Data ($\mathbb{R}^{441 \times 41}$): 441 legislators and 41 roll calls all 0/1s

Transformed Data ($\mathbb{R}^{441 \times 2}$): reduced to 441 points into two dimensions.

- The choice of 2 dimensions was a hyper-parameter decision.

Interesting Observations

- The sign of the first coordinate agrees with party affiliation for **98%** of the House
- **No party label** was supplied to the algorithm. The input consisted of **just votes**.
- The fitted **model** consists of **two vectors** of **41 numbers**. (the parameters)



How does PCA construct this
low-dimensional representation of the
data?



3640956

Objective of Dimensionality Reduction

High-dimensional data (many columns) and can be difficult to **visualize** and **model**.

- Challenging to **plot** more than a few dimensions
- Some **modeling** techniques perform poorly **in high-dimensions** (e.g., k-NN and k-Means)

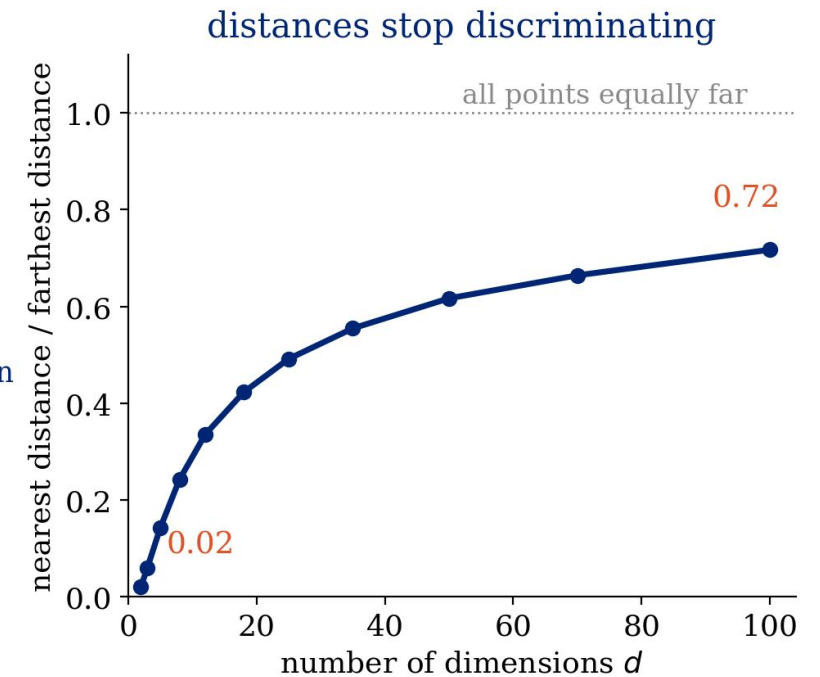
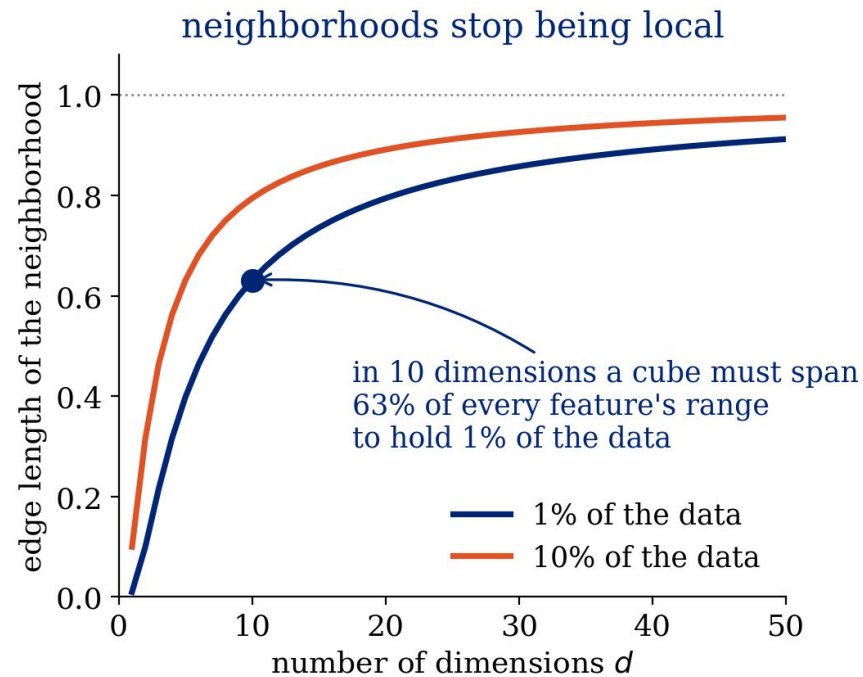
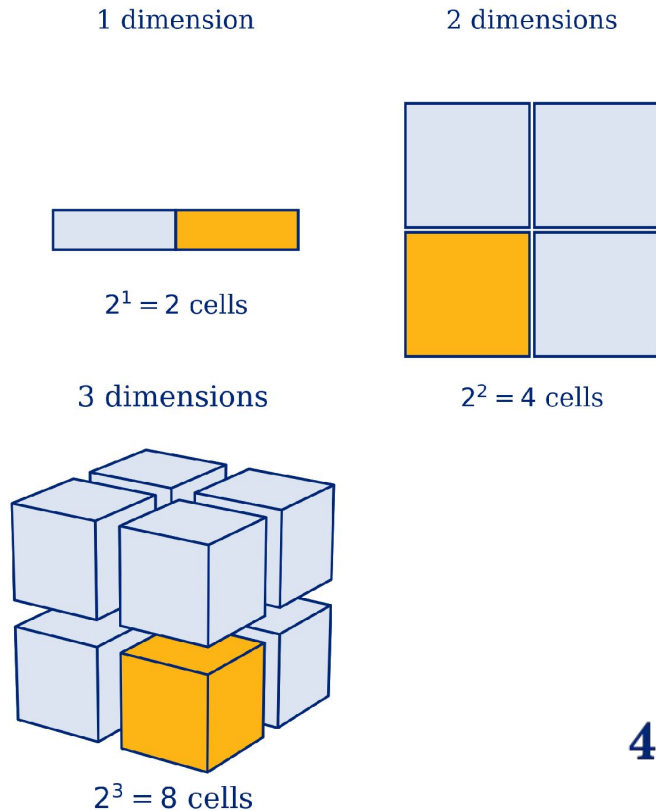
The Curse of Dimensionality



3640956

Volume grows exponentially with d :

- **Neighborhoods stop being local:** a box holding 1% of the data spans most of every axis
- **Distances stop discriminating:** the nearest and farthest points become almost equally far



441 legislators in 41 dimensions: splitting each axis once gives $2^{41} \approx 2$ trillion cells for 441 points.



3640956

Objective of Dimensionality Reduction

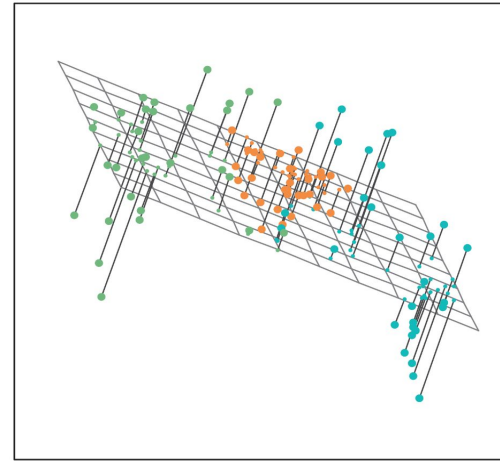
High-dimensional data (many columns) and can be difficult to **visualize** and **model**.

- Challenging to **plot** more than a few dimensions
- Some **modeling** techniques perform poorly **in high-dimensions** (e.g., k-NN and k-Means)

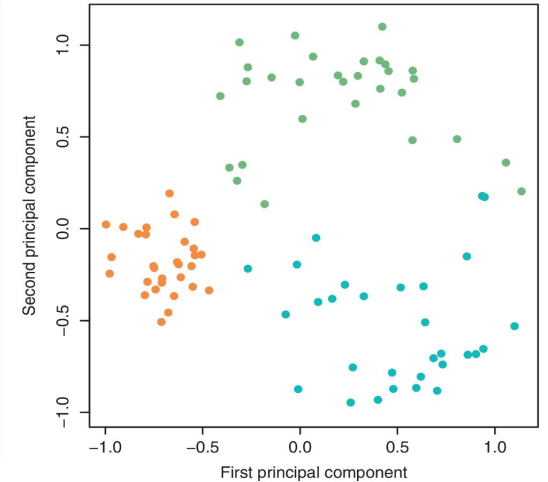
Often high-dimensional data **can be compressed** into a **lower dimensional space** with minimal loss of information.

Dimensionality reduction: take high-dimensional data and map it to a lower dimensional space that **preserves as much information** as possible.

3D Data



2D Representation



Formulating the Problem



3640956

The Rank of a Matrix

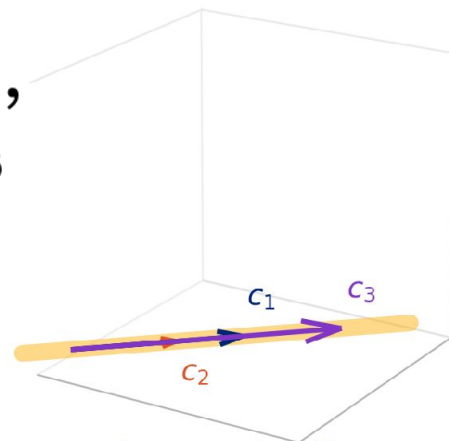
The **rank** of a matrix is:

- the dimension of the **space spanned by the columns**.
- the **maximum number of *linearly independent* columns**.

A **column** is **linearly independent** of all other columns if it cannot be expressed as a linear combination of the other columns.

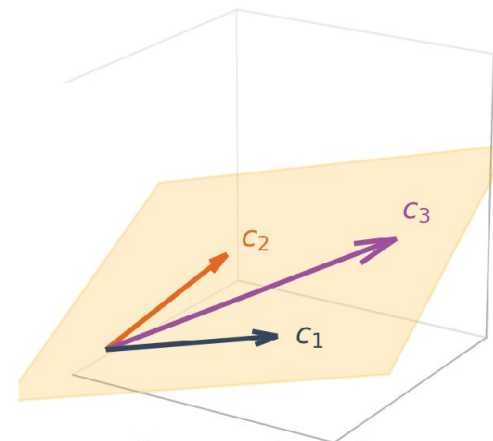
The **$\text{rank}(X) \leq \min(n, d)$** ,
and a matrix that attains
this bound is said to
have **full rank**.

rank 1



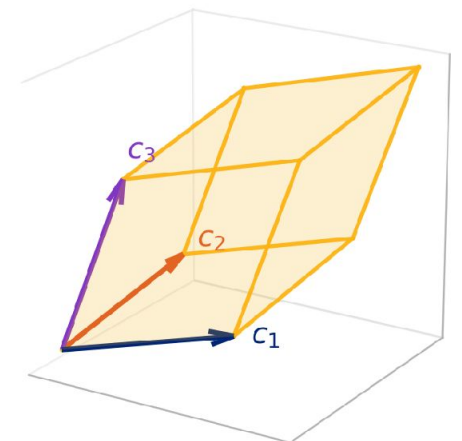
the span is a line
every column is a multiple of one direction

rank 2



the span is a plane
 $c_3 = c_1 + c_2$, adding nothing new

rank 3



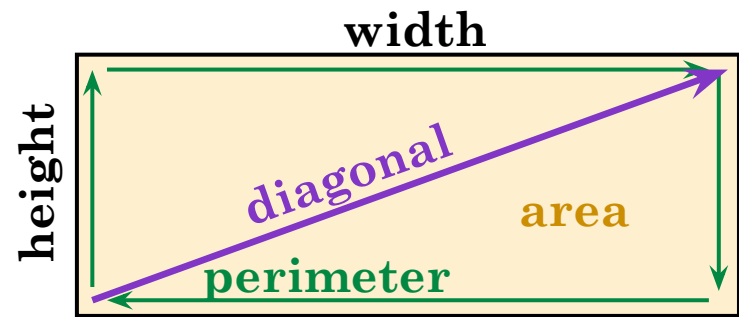
the span is all of $\mathbb{R}^3$
no column is a combination of the others



3640956

Simple Example

Suppose I have a large table of random rectangles along with **properties** of those rectangles:



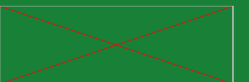
width	height	perimeter	area	diagonal
7	6	26	42	9.22
4	6	20	24	7.21
5	2	14	10	5.39
7	4	22	28	8.06
3	5	16	15	5.83

What is the **rank** of this dataset?

What is the rank of the following dataset?



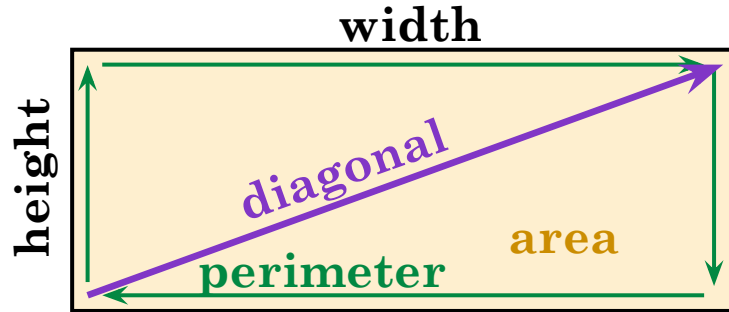
The Slido app must be installed on every computer you're presenting from



Rank of the Rectangles Example



3640956



width	height	perimeter	area	diagonal
7	6	26	42	9.22
4	6	20	24	7.21
5	2	14	10	5.39
7	4	22	28	8.06
3	5	16	15	5.83

5 columns, rank 4

the perimeter column is redundant, so
the matrix is not full rank

$$\text{perimeter} = 2w + 2h$$

*a linear combination of the
other two columns*

$$\text{area} = w \times h$$

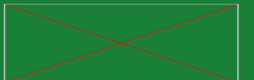
not a linear combination

$$\text{diagonal} = \sqrt{w^2 + h^2}$$

not a linear combination

How many dimensions (columns) are really needed?

How many dimensions are needed to fully describe this dataset?



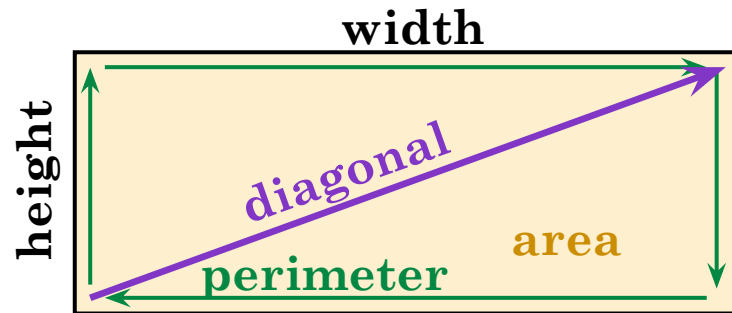


3640956

Intrinsic Dimension and Matrix Rank

The **intrinsic dimension** is the minimum number of variables needed to describe the data.

- Unlike rank, intrinsic dimension allows **nonlinear relationships between variables**.



$$\text{perimeter} = 2w + 2h$$

a linear combination of the other two columns

$$\text{area} = w \times h$$

not a linear combination

$$\text{diagonal} = \sqrt{w^2 + h^2}$$

not a linear combination

5 columns, rank 4

the perimeter column is redundant, so the matrix is not full rank

Intrinsic Dimension is 2

every column is determined by w and h alone

Every column of that table is determined by the **width** w and the **height** h .



3640956

Real data is rarely *exactly* low rank

Measure the perimeter with a tape measure instead of computing it, and the exact relationship is destroyed □ **the rank returns to 5.**

- Small amounts of **noise** can make data full rank.

Example: the congressional vote matrix from the demo

Full Rank (41)

nothing may be discarded

Rank 2 Approx. → 86%

of the spread in the data

Typically, datasets are nearly **full rank** but are often accurately approximated by a **lower dimensional sub-space** (close to the **intrinsic dimension**).

How do we find a lower dimensional representation that accurately approximates the data?

Deriving PCA

The input Data Matrix



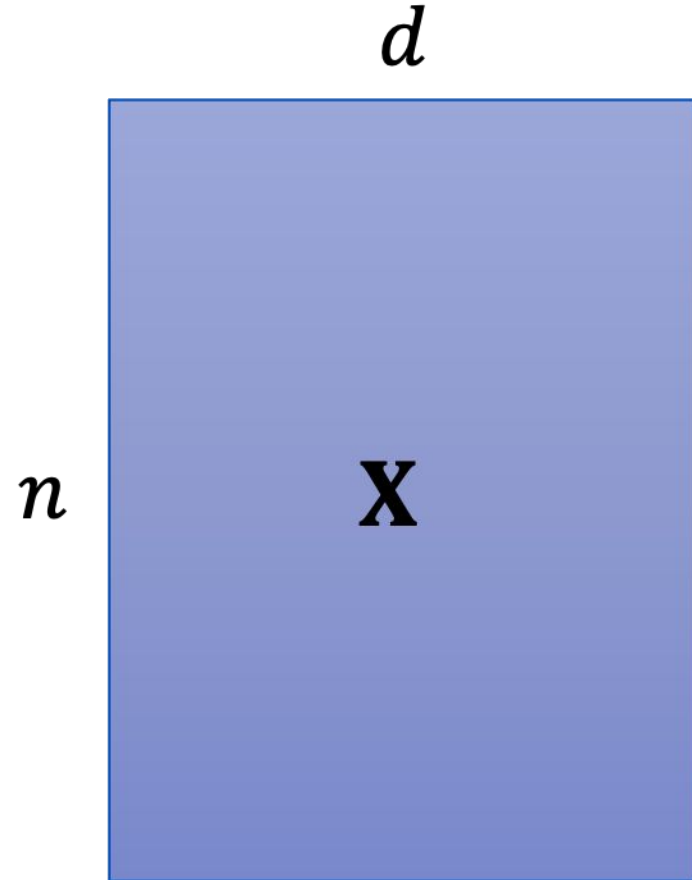
3640956

For n datapoints and d features:

$$\mathbf{X} \in \mathbb{R}^{n \times d}$$

The i -th row is a **single datapoint**, written as
a **row vector** with d entries:

$$\mathbf{X}_i \in \mathbb{R}^{1 \times d}$$



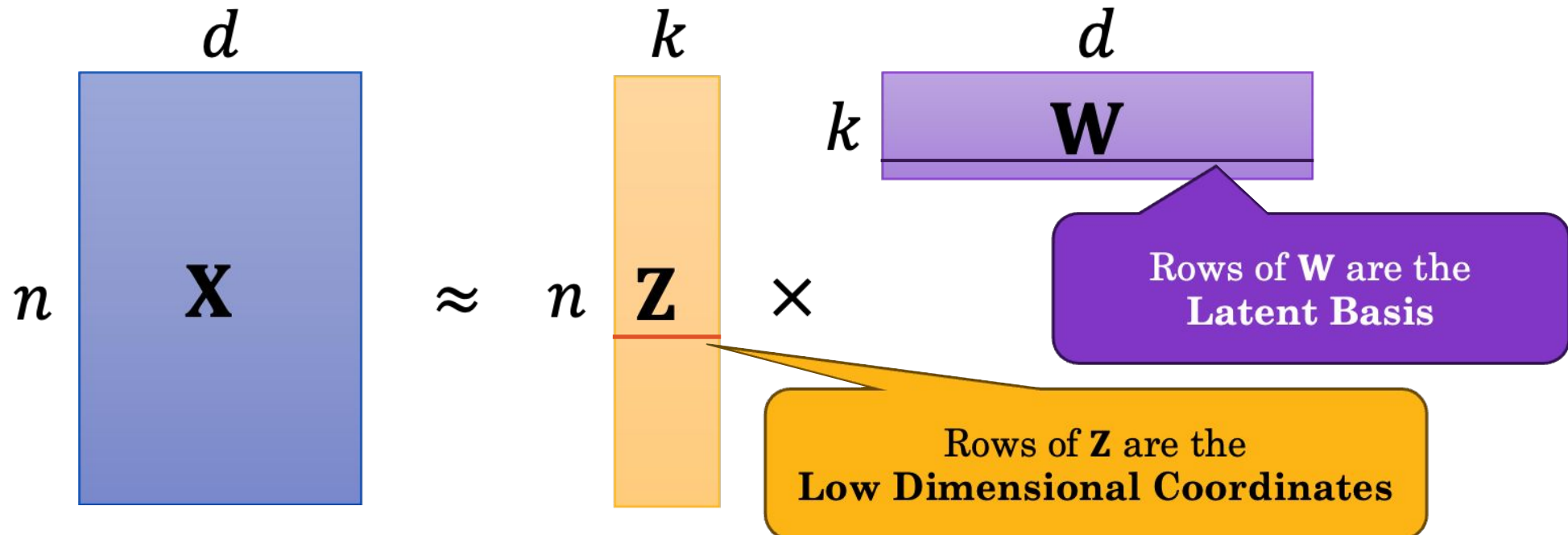
Dimensionality reduction as Matrix Factorization



3640956

Goal: Construct a matrix $\mathbf{Z} \in \mathbb{R}^{n \times k}$ of size $n \times k$ and a matrix $\mathbf{W} \in \mathbb{R}^{k \times d}$ of size $k \times d$ whose product reproduces $\mathbf{X}$ as closely as possible:

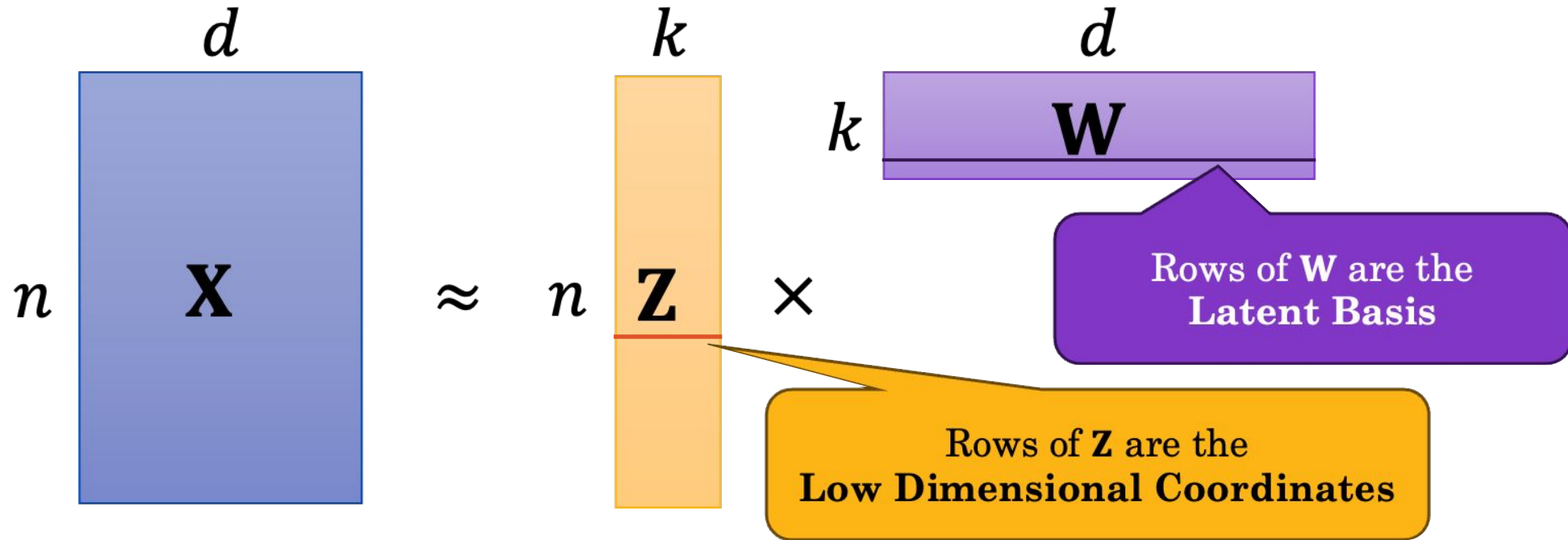
$$\mathbf{X} \approx \mathbf{Z}\mathbf{W}$$





3640956

Interpretation of Z and W



Each **row of W** determines the basis of the **subspace of X** used to approximate the data. (*These will become our Principal Components give the opt. problem.*)

Each **row of Z** determines the “mix” of the rows of W used to reconstruct a row in the original data.



3640956

Which of these products can equal $\mathbf{X}$ exactly?

Suppose $\mathbf{X} \in \mathbb{R}^{3 \times 3}$ is a any 3×3 matrix with full rank. Which factorizations could always exactly reconstruct $\mathbf{X}$.

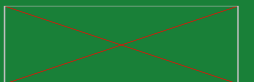
- A. $(3 \times 2) \times (2 \times 3)$
- B. $(3 \times 3) \times (3 \times 3)$
- C. $(3 \times 1) \times (1 \times 3)$
- D. $(3 \times 4) \times (4 \times 3)$
- E. Something else



Suppose $\mathbf{X} \in \mathbb{R}^{(3 \times 3)}$ is a any 3×3 matrix with full rank. Which factorizations could always exactly reconstruct $\mathbf{X}$.



The Slido app must be installed on every computer you're presenting from





3640956

The rank of the product is bounded by k

Suppose $\mathbf{X} \in \mathbb{R}^{3 \times 3}$ is a any 3×3 matrix with full rank. Which factorizations could always exactly reconstruct $\mathbf{X}$.

A. $(3 \times 2) \times (2 \times 3)$

B. $(3 \times 3) \times (3 \times 3)$

C. $(3 \times 1) \times (1 \times 3)$

D. $(3 \times 4) \times (4 \times 3)$

$$\text{rank}(\mathbf{ZW}) \leq \min(\text{rank } \mathbf{Z}, \text{rank } \mathbf{W}) \leq k$$

- $\mathbf{B}$ and $\mathbf{D}$ can reproduce $\mathbf{X}$ exactly, but achieve no reduction
- $\mathbf{A}$ and $\mathbf{C}$ cannot (in general), since their inner dimension is smaller than the rank of $\mathbf{X}$

In practice k is chosen below the rank of $\mathbf{X}$, so that $\mathbf{ZW}$ is necessarily an approximation.

**Among all factorizations of rank k ,
which one is best?**



3640956

Defining an objective function

The product $\mathbf{Z}_i \mathbf{W}$ is our reconstruction of the datapoint $\mathbf{X}_i$. We evaluate a choice of $(\mathbf{Z}, \mathbf{W})$ by the **average squared error** of that reconstruction:

$$\text{Loss}(\mathbf{Z}, \mathbf{W}) = \frac{1}{n} \sum_{i=1}^n \|\mathbf{X}_i - \mathbf{Z}_i \mathbf{W}\|^2$$

Squared
Euclidean Norm

$$\|\mathbf{a}\|^2 = \sum_{i=1}^d a_i^2$$

$\mathbf{X}$ is assumed to be centered

that is, the mean of each column has been subtracted.

This is an optimization problem

The parameters are $\mathbf{Z}$ and $\mathbf{W}$, and there is a single quantity to be minimized. It is the first of many such problems this semester.

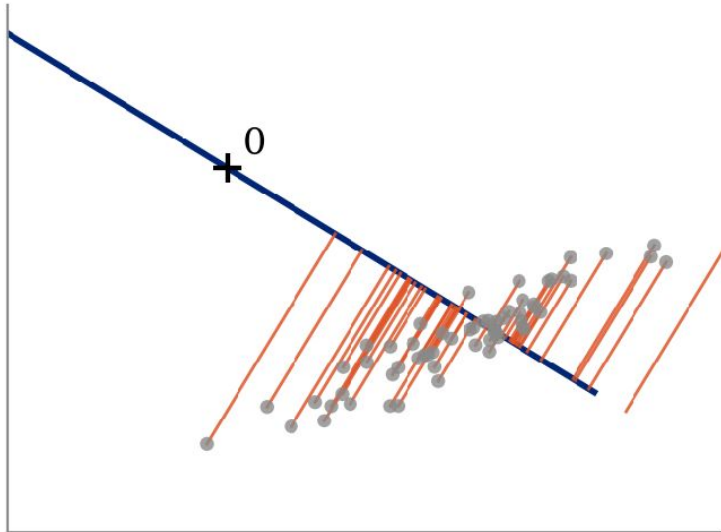


3640956

Why we subtract the mean

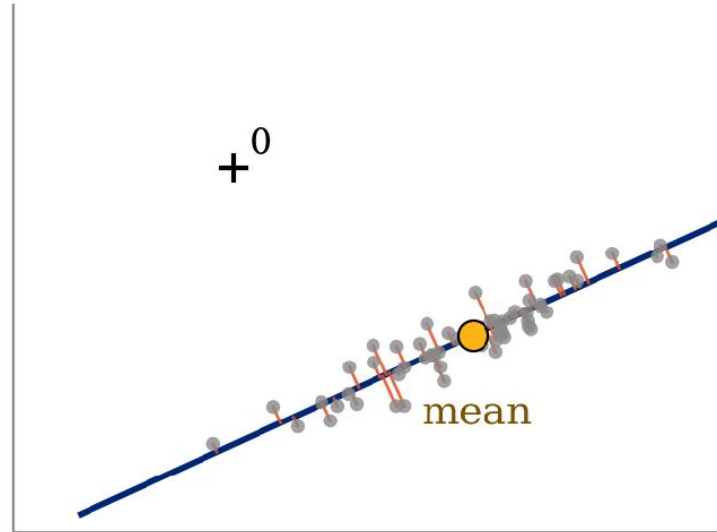
As written, the model $\mathbf{X} \approx \mathbf{Z}\mathbf{W}$ places every reconstruction on a subspace through the origin. The origin is an arbitrary point, and the data has no reason to lie near it.

the model as written:
the line must pass through 0



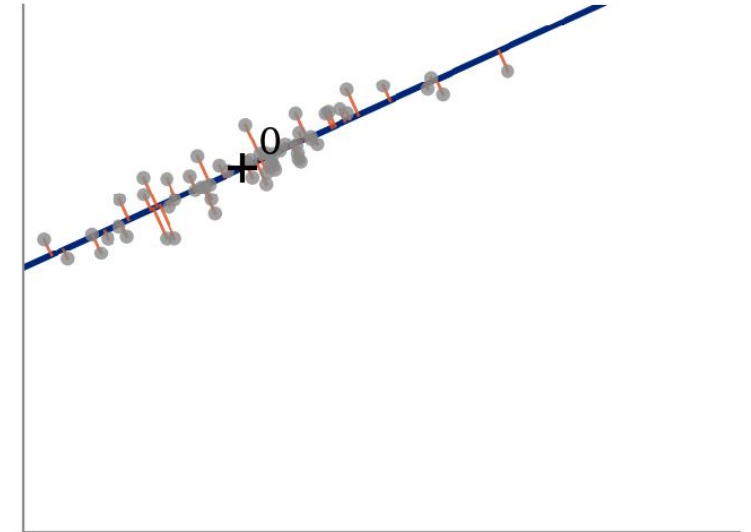
average squared error = 3.73

what we actually want:
the line may sit anywhere



average squared error = 0.11

subtract the mean, and the two
problems become the same one



average squared error = 0.11

The best fitting subspace always passes through the mean,
so we subtract the mean first.



3640956

The Factorization is Not Unique

For any invertible $k \times k$ matrix $\mathbf{A}$:

$$\mathbf{Z}\mathbf{W} = (\mathbf{Z}\mathbf{A}^{-1})(\mathbf{A}\mathbf{W})$$

- The reconstruction is identical, while $\mathbf{Z}$ and $\mathbf{W}$ are entirely different.
- Alternatively, $\mathbf{W}$ may be scaled down and $\mathbf{Z}$ scaled up without affecting the objective.

We remove the scale ambiguity by requiring the rows of $\mathbf{W}$ to be orthonormal

$$\mathbf{W}\mathbf{W}^T = \mathbf{I}$$

- Row vectors of $\mathbf{W}$ are unit length: $\mathbf{w}_i \mathbf{w}_i^T = 1$
- Row vectors of $\mathbf{W}$ are orthogonal: $\mathbf{w}_i \mathbf{w}_j^T = 0$ for all $i \neq j$

Nothing is lost by doing so, since every k -dimensional subspace admits an **orthonormal basis**.



3640956

The Full Optimization Problem

Determine the values of $(\mathbf{Z}, \mathbf{W})$ that **minimize** the **average squared error** of that reconstruction:

$$(\mathbf{Z}^*, \mathbf{W}^*) = \arg \min_{\mathbf{Z} \in \mathbb{R}^{n \times k}, \mathbf{W} \in \mathbb{R}^{k \times d}} \left(\frac{1}{n} \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{z}_i \mathbf{W}\|^2 \right)$$

subject to the **row orthonormality constraints**:

$$\mathbf{W} \mathbf{W}^\top = \mathbf{I}$$

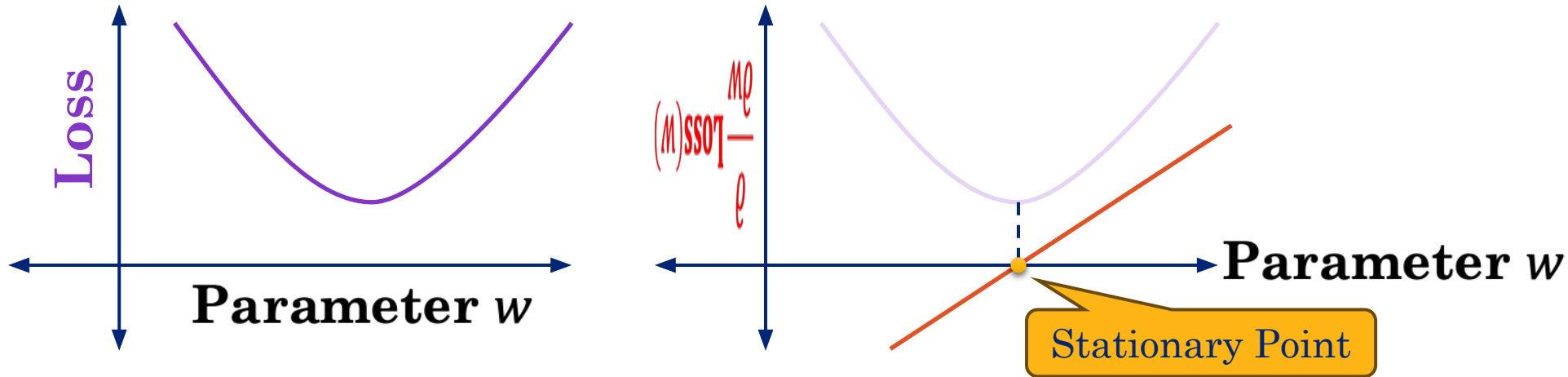
Plan:

1. Optimize $\mathbf{Z}$ in terms of $\mathbf{W}$ (**Stationary Point Method**)
2. Optimize $\mathbf{W}$ with constraints (**Method of Lagrange Multipliers**)



3640956

The *Basic* Stationary Point Method



- **Differentiate** with respect to each parameter
- Set the **derivatives to zero and solve**
- **(Optional)** verify that second derivative is positive.

$$\frac{\partial}{\partial w} \mathbf{Loss}(w) = 0$$

This procedure applies when the objective is **smooth** and the **resulting equations can be solved**. All conditions hold today, and we obtain an exact solution.



3640956

Case $k = 1$ (Rank 1 Factorization)

A single unit direction $\mathbf{w} \in \mathbb{R}^{1 \times d}$, with $\mathbf{w}\mathbf{w}^\top = \mathbf{1}$

A single coefficient $z_i \in \mathbb{R}$ for each datapoint (row) in $\mathbf{X}$

$$\begin{matrix} & d \\ & \mathbf{X} \\ n & \end{matrix} \approx \begin{matrix} 1 \\ n \\ \mathbf{z} \end{matrix} \times \begin{matrix} 1 & d \\ \mathbf{w} \end{matrix}$$

$$\text{Loss}(\mathbf{z}, \mathbf{w}) = \frac{1}{n} \sum_{i=1}^n \|\mathbf{X}_i - z_i \mathbf{w}\|^2$$

Expanding the Loss

$$\begin{aligned} \|\mathbf{X}_i - z_i \mathbf{w}\|^2 &= (\mathbf{X}_i - z_i \mathbf{w})(\mathbf{X}_i - z_i \mathbf{w})^\top \\ &= \underbrace{\mathbf{X}_i \mathbf{X}_i^\top}_{\text{Constant with respect to } (\mathbf{z}, \mathbf{w})} - 2z_i \mathbf{X}_i \mathbf{w}^\top + z_i^2 \underbrace{\mathbf{w} \mathbf{w}^\top}_{\substack{1 \\ \text{(orthonormality constraint)}}} \end{aligned}$$

Simplified Loss

$$\text{Loss}(\mathbf{z}, \mathbf{w}) = C + \frac{1}{n} \sum_{i=1}^n (-2z_i \mathbf{X}_i \mathbf{w}^\top + z_i^2)$$

Computing the Stationary Point z_i^*

$$\text{Loss}(\mathbf{z}, \mathbf{w}) = C + \frac{1}{n} \sum_{i=1}^n (-2z_i \mathbf{X}_i \mathbf{w}^\top + z_i^2)$$



3640956

Take the **derivative of the loss** with respect to the **parameter** (z_i) that we want to optimize

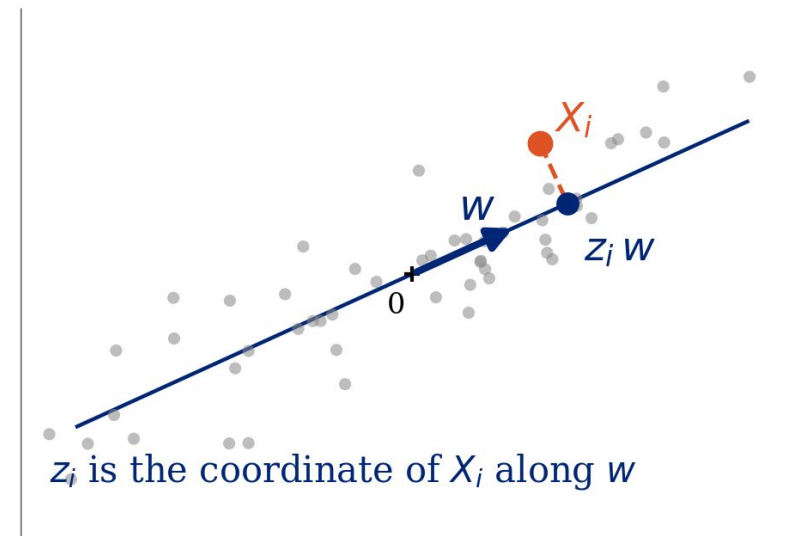
$$\frac{\partial}{\partial z_i} \text{Loss}(\mathbf{z}, \mathbf{w}) = -2 \frac{1}{n} \mathbf{X}_i \mathbf{w}^\top + 2 \frac{1}{n} z_i$$

Set the derivative equal to zero and solve for the parameter

$$-2 \frac{1}{n} \mathbf{X}_i \mathbf{w}^\top + 2 \frac{1}{n} z_i = 0 \quad \Rightarrow \quad z_i = \mathbf{X}_i \mathbf{w}^\top$$

The optimal value for z_i is **the projection**:

$$z_i^* = \mathbf{X}_i \mathbf{w}^\top$$



Substituting the solution for $\mathbf{z}^*$

$$\text{Loss}(\mathbf{z}, \mathbf{w}) = C + \frac{1}{n} \sum_{i=1}^n (-2z_i \mathbf{X}_i \mathbf{w}^\top + z_i^2)$$



3640956

We derived z_i^* in terms of $\mathbf{w}$ so we can rewrite the loss in terms of only $\mathbf{w}$:

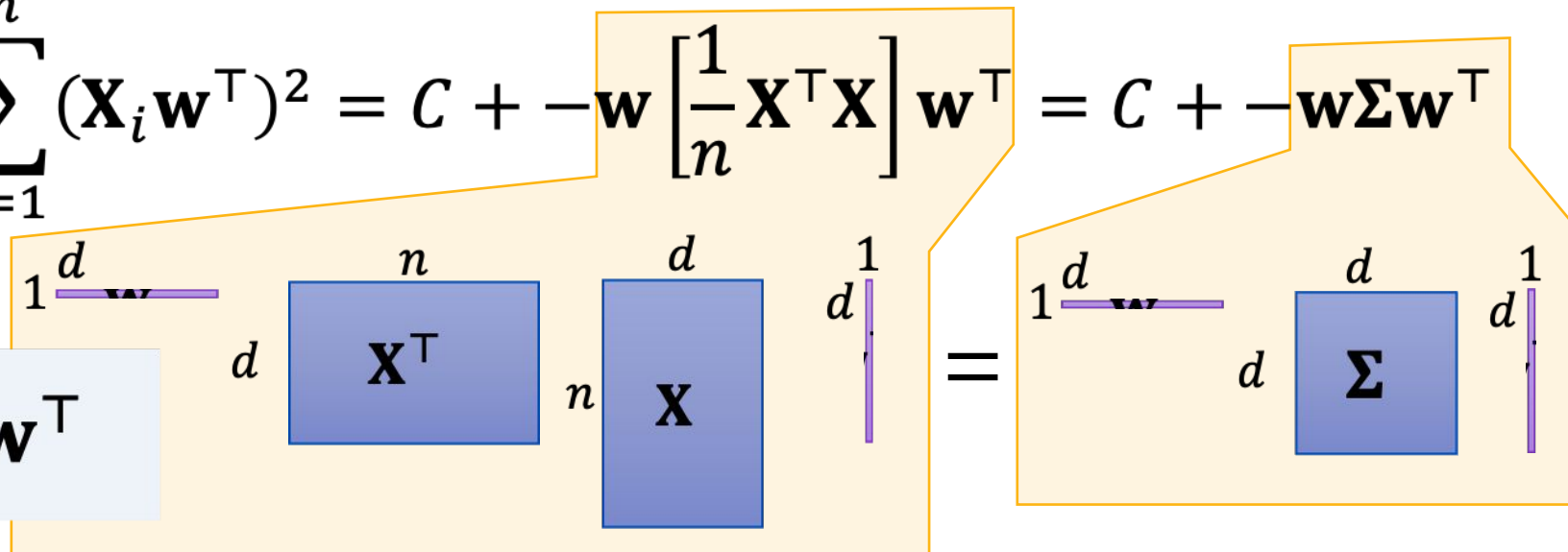
$$z_i^* = \mathbf{X}_i \mathbf{w}^\top$$

$$\text{Loss}(\mathbf{w}) = C + \frac{1}{n} \sum_{i=1}^n (-2(\mathbf{X}_i \mathbf{w}^\top) \mathbf{X}_i \mathbf{w}^\top + (\mathbf{X}_i \mathbf{w}^\top)^2)$$

Simplifying terms:

$$\text{Loss}(\mathbf{w}) = C + -\frac{1}{n} \sum_{i=1}^n (\mathbf{X}_i \mathbf{w}^\top)^2 = C + -\mathbf{w} \left[\frac{1}{n} \mathbf{X}^\top \mathbf{X} \right] \mathbf{w}^\top = C + -\mathbf{w} \Sigma \mathbf{w}^\top$$

$$\text{Loss}(\mathbf{w}) = C - \mathbf{w} \Sigma \mathbf{w}^\top$$





3640956

Constrained Optimization for $\mathbf{w}$

We want to minimize the loss with respect to $\mathbf{w}$ (what could go wrong?)

$$\text{Loss}(\mathbf{w}) = \mathbf{C} - \mathbf{w}\Sigma\mathbf{w}^\top$$

If you naively optimize this loss with respect to $\mathbf{w}$ you can make it arbitrarily negative by making $\mathbf{w}$ **arbitrarily large**.

We still need to account for the **row orthonormality constraint**:

$$\mathbf{w}\mathbf{w}^\top = \mathbf{1}$$

The constraint is essential: it is what makes the problem well posed.



3640956

The Method of Lagrange Multipliers

$$\text{Loss}(\mathbf{w}) = \mathbf{C} - \mathbf{w}\Sigma\mathbf{w}^\top$$

Convert a **constrained optimization** into an **unconstrained optimization problem** by introducing the **Lagrangian**:

$$\mathcal{L}(\mathbf{w}, \lambda) = \text{Loss}(\mathbf{w}) + \lambda(\mathbf{w}\mathbf{w}^\top - 1)$$

The term $(\mathbf{w}\mathbf{w}^\top - 1)$ measures how much the **orthonormality constraint** ($\mathbf{w}\mathbf{w}^\top = 1$) is violated.

The **Lagrange multiplier** λ is chosen automatically by applying the **stationary point method** to the **Lagrangian**

- Differentiating with respect to $\mathbf{w}$:

$$\nabla_{\mathbf{w}}\text{Loss}(\mathbf{w}) + \lambda\nabla_{\mathbf{w}}(\mathbf{w}\mathbf{w}^\top - 1) = 0 \quad \Rightarrow \quad \nabla_{\mathbf{w}}\text{Loss}(\mathbf{w}) = -\lambda\nabla_{\mathbf{w}}(\mathbf{w}\mathbf{w}^\top - 1)$$

- Differentiating with respect to λ :

$$\frac{\partial}{\partial \lambda}\mathcal{L}(\mathbf{w}, \lambda) = (\mathbf{w}\mathbf{w}^\top - 1) = 0 \quad \Rightarrow \quad \mathbf{w}\mathbf{w}^\top = 1$$

Gradients of the **loss** and **scaled constraint violation** should be parallel!



3640956

A Small Example

Minimize $f(x, y) = x + y$ subject to $x^2 + y^2 = 1$

$$\mathcal{L}(x, y, \lambda) = x + y + \lambda(x^2 + y^2 - 1)$$

$$\frac{\partial \mathcal{L}}{\partial x} = 1 + 2\lambda x = 0$$

$$\frac{\partial \mathcal{L}}{\partial y} = 1 + 2\lambda y = 0$$

so $x = y$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = x^2 + y^2 - 1 = 0$$

with $x = y$ this gives

$$x = y = \pm 1/\sqrt{2}$$

The minimum is $x = y = -1/\sqrt{2}$.



3640956

Applying the Method of Lagrange Multipliers

Constructing the **Lagrangian**:

$$\text{Loss}(\mathbf{w}) = \mathbf{C} - \mathbf{w}\Sigma\mathbf{w}^\top$$

$$\mathcal{L}(\mathbf{w}, \lambda) = \mathbf{C} - \mathbf{w}\Sigma\mathbf{w}^\top + \lambda(\mathbf{w}\mathbf{w}^\top - 1)$$

Differentiating with respect to $\mathbf{w}$:

$$\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}, \lambda) = -\nabla_{\mathbf{w}}\mathbf{w}\Sigma\mathbf{w}^\top + \lambda\nabla_{\mathbf{w}}(\mathbf{w}\mathbf{w}^\top - 1) = -\nabla_{\mathbf{w}}\mathbf{w}\Sigma\mathbf{w}^\top + \lambda\nabla_{\mathbf{w}}\mathbf{w}\mathbf{w}^\top$$

Compute the gradient of $\mathbf{w}\Sigma\mathbf{w}^\top$ by taking the partial derivatives:

$$\frac{\partial}{\partial w_i}\mathbf{w}\Sigma\mathbf{w}^\top = \frac{\partial}{\partial w_i}\sum_{j=1}^d\sum_{k=1}^dw_j\Sigma_{jk}w_k = \sum_{j=1}^dw_j\Sigma_{ji} + \sum_{k=1}^d\Sigma_{ik}w_k$$

$$\nabla_{\mathbf{w}}\mathbf{w}\Sigma\mathbf{w}^\top = \Sigma\mathbf{w}^\top + \Sigma^\top\mathbf{w}^\top = (\underbrace{\Sigma + \Sigma^\top})\mathbf{w}^\top = 2\Sigma\mathbf{w}^\top$$

Doing the same for $\nabla_{\mathbf{w}}\mathbf{w}\mathbf{w}^\top$ we obtain:

$\Sigma = \mathbf{X}^\top\mathbf{X}$ is symmetric

$$\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}, \lambda) = -2\Sigma\mathbf{w}^\top + 2\lambda\mathbf{w}^\top$$



3640956

Differentiating with respect to $\mathbf{w}$:

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \lambda) = -\nabla_{\mathbf{w}} \mathbf{w} \Sigma \mathbf{w}^{\top} + \lambda \nabla_{\mathbf{w}} (\mathbf{w} \mathbf{w}^{\top} - 1) = -\nabla_{\mathbf{w}} \mathbf{w} \Sigma \mathbf{w}^{\top} + \lambda \nabla_{\mathbf{w}} \mathbf{w} \mathbf{w}^{\top}$$

Compute the gradient of $\mathbf{w} \Sigma \mathbf{w}^{\top}$ by taking the partial derivatives:

$$\frac{\partial}{\partial w_i} \mathbf{w} \Sigma \mathbf{w}^{\top} = \frac{\partial}{\partial w_i} \sum_{j=1}^d \sum_{k=1}^d w_j \Sigma_{jk} w_k = \sum_{j=1}^d w_j \Sigma_{ji} + \sum_{k=1}^d \Sigma_{ik} w_k$$

$$\nabla_{\mathbf{w}} \mathbf{w} \Sigma \mathbf{w}^{\top} = \Sigma \mathbf{w}^{\top} + \underbrace{\Sigma^{\top} \mathbf{w}^{\top}}_{\Sigma = \mathbf{X}^{\top} \mathbf{X} \text{ is symmetric}} = 2 \Sigma \mathbf{w}^{\top}$$

Doing the same for $\nabla_{\mathbf{w}} \mathbf{w} \mathbf{w}^{\top}$ we obtain:

$\Sigma = \mathbf{X}^{\top} \mathbf{X}$ is symmetric

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \lambda) = -2 \Sigma \mathbf{w}^{\top} + 2 \lambda \mathbf{w}^{\top}$$

Setting $\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \lambda) = 0$ we obtain a key result:

$$\Sigma \mathbf{w}^{\top} = \lambda \mathbf{w}^{\top}$$

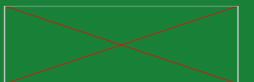


What is this?

What does this equation
 $(\Sigma \mathbf{w}^T = \lambda \mathbf{w}^T)$ say about $\mathbf{w}$?



The Slido app must be installed on every computer you're presenting from





3640956

The Eigenvector Equation

Therefore, we know that $\mathbf{w}^*$ must be an eigenvector.

$$\Sigma \mathbf{w}^\top = \lambda \mathbf{w}^\top$$

It must also be **unit length** to satisfy the $\mathbf{w}\mathbf{w}^\top = 1$ constraint.

If we return to the original loss definition and substitute $\Sigma \mathbf{w}^\top = \lambda \mathbf{w}^\top$

$$\text{Loss}(\mathbf{w}) = -\mathbf{w}\Sigma \mathbf{w}^\top = -\mathbf{w}(\lambda \mathbf{w}^\top) = -\lambda \mathbf{w}\mathbf{w}^\top = -\lambda$$

We see that $\mathbf{w}^*$ must be the Eigenvector with the **largest Eigenvalue**.

The **first principal component** is the **unit eigenvector** of $\Sigma = \mathbf{X}^\top \mathbf{X}$ associated with the **largest eigenvalue**.



3640956

Subsequent Principal Components

Repeating for the next principal component $\mathbf{w}_2$, with one additional constraint:

$$\mathbf{w}_2 \mathbf{w}_1^\top = \mathbf{0}$$

This introduces an additional Lagrange multiplier.

$$\mathcal{L}(\mathbf{w}_2, \lambda_2, \gamma) = -\mathbf{w}_2 \mathbf{\Sigma} \mathbf{w}_2^\top + \lambda_2 (\mathbf{w}_2 \mathbf{w}_2^\top - 1) + \gamma (\mathbf{w}_2 \mathbf{w}_1^\top)$$

Differentiating with respect to $\mathbf{w}_2$ and left multiply by $\mathbf{w}_1$ you can show that $\gamma = 0$ and again (*try doing this yourself*):

$$\mathbf{\Sigma} \mathbf{w}_2^\top = \lambda_2 \mathbf{w}_2^\top$$

Hence $\mathbf{w}_2$ is the eigenvector associated with the second largest eigenvalue.

The rows of $\mathbf{W}$ are the **leading k eigenvectors** of $\mathbf{\Sigma}$, and the **latent factors (coordinates)** are:

$$\mathbf{Z} = \mathbf{XW}^\top$$



3640956

Computing the Principal Components

Constructing $\mathbf{X}^\top \mathbf{X}$ and computing its eigen decomposition is correct but expensive.

Instead, we typically use the **singular value decomposition**:

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

```
U, S, Vt = np.linalg.svd(X, full_matrices=False)
```

The columns of $\mathbf{V}$ are precisely the eigenvectors of $\mathbf{X}^\top \mathbf{X}$.

- *This result is stated without proof today (but you can do it for homework). The [EECS 16B notes on the SVD](#) are a great reference, and [Steve Brunton's video series on the PCA](#) is excellent.*



3640956

Returning to the vote data

Given the singular value decomposition $\mathbf{X} = \mathbf{U} \mathbf{S} \mathbf{V}^\top$.

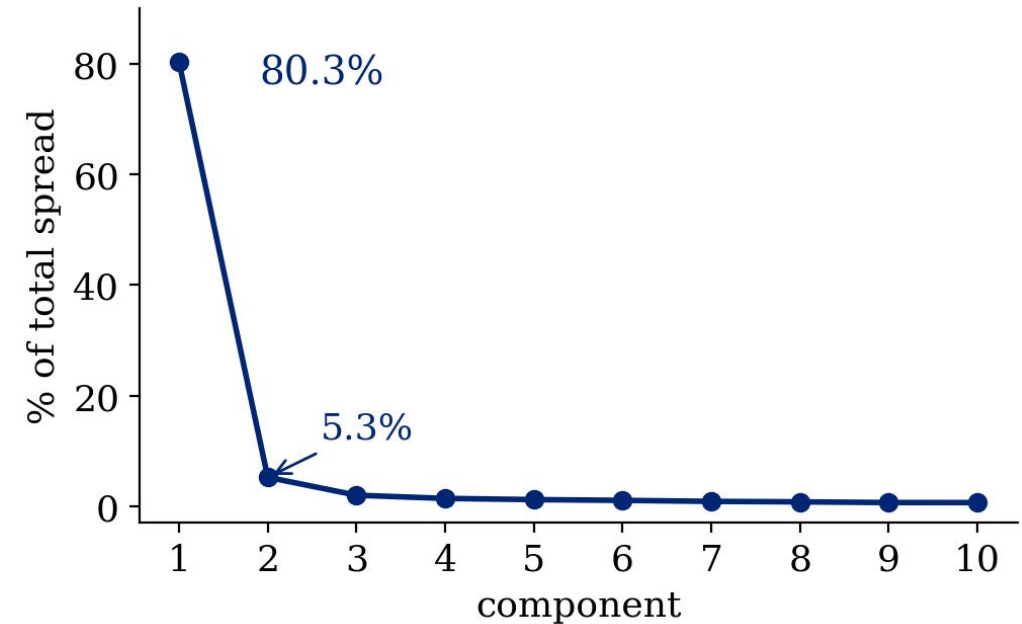
The **Explained Variance Ratio** is

$$\frac{s_{ii}^2}{\sum_{d=1}^d s_{dd}^2}$$

the share of the **total spread** accounted for by each principal component.

PC1: 80.3% PC2: 5.3% together: 85.6%

- The **sharp decrease** after the first component is what makes the two-dimensional plot trustworthy.
- The centered matrix nonetheless has full rank 41. The data is **approximately two-dimensional**.



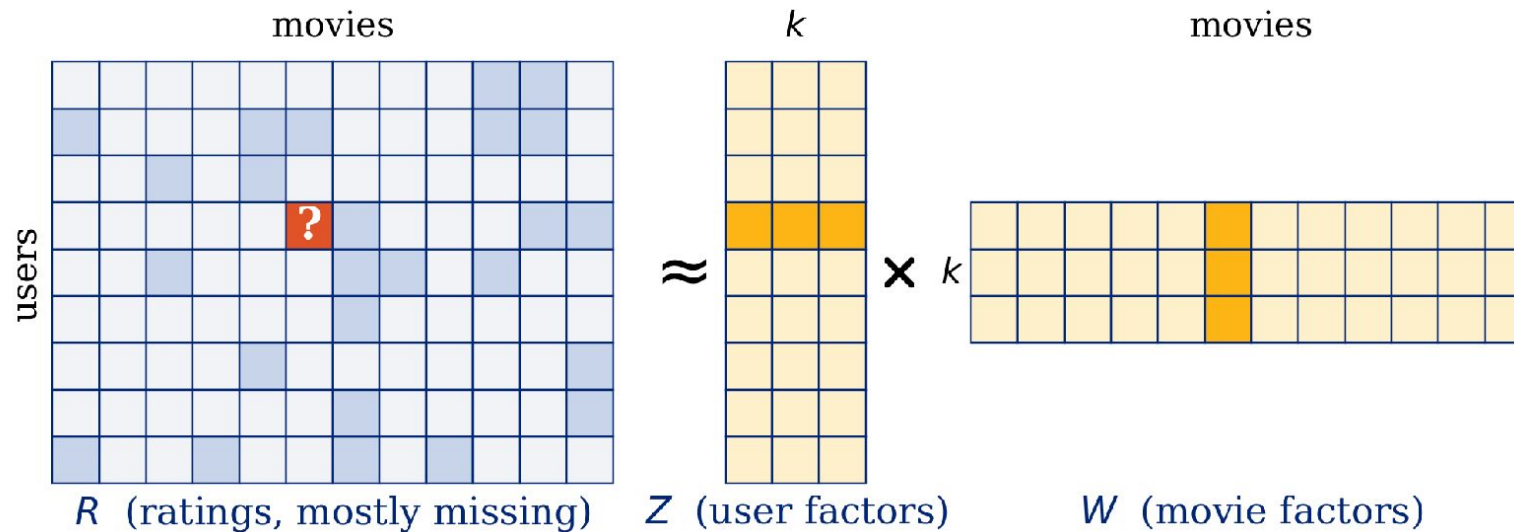
Bonus

Matrix Factorization for Content Recommendation



3640956

- **R holds ratings:** one row per user, one column per movie, and almost every entry is missing
- **Netflix Prize:** 8.5 billion user and movie pairs, of which about 1 in 85 was ever rated



A missing rating is predicted by the dot product of that user's row with that movie's column

$$\text{Loss}(Z, W) = \sum_{(i,j) \in \Omega} (R_{ij} - Z_i W_j)^2 + \alpha (\|Z\|^2 + \|W\|^2)$$

Drop unit vector constraint and add a **penalty term**.

Only the **observed pairs** Ω are part of the loss.

The **learned factors** are then used to **fill in every rating** that was never given.

Summary



3640956

- **Dimensionality reduction** was posed as a **factorization**,

$$\mathbf{X} \approx \mathbf{Z} \mathbf{W}$$

- Optimality was defined as **smallest reconstruction error**, which proved equivalent to greatest spread
- Applied **stationary point method** to optimize the loss:
 - Differentiating with respect to $\mathbf{z}$ showed that the **optimal coefficient** is a **projection**
 - Differentiating with respect to $\mathbf{w}$, subject to a constraint, produced an **eigenvector problem**
- The principal components are the **leading eigenvectors** of $\frac{1}{n} \mathbf{X}^\top \mathbf{X}$, and the **latent factors (coordinates)** are the **projections** $\mathbf{Z} = \mathbf{X} \mathbf{W}^\top$

Next week we turn to probability!

Lecture 3

Dimensionality Reduction

Credit: Joseph E. Gonzalez and Narges Norouzi

Reference Book Chapters:

- **Linear Algebra, Matrix Calculus, and Eigenvectors:** Appendix (A.1, A.3, A.4)
- **Lagrange Multipliers:** Appendix C
- **PCA:** Chapter 16.1

Have Fun!!!