Applied Math for Machine Learning

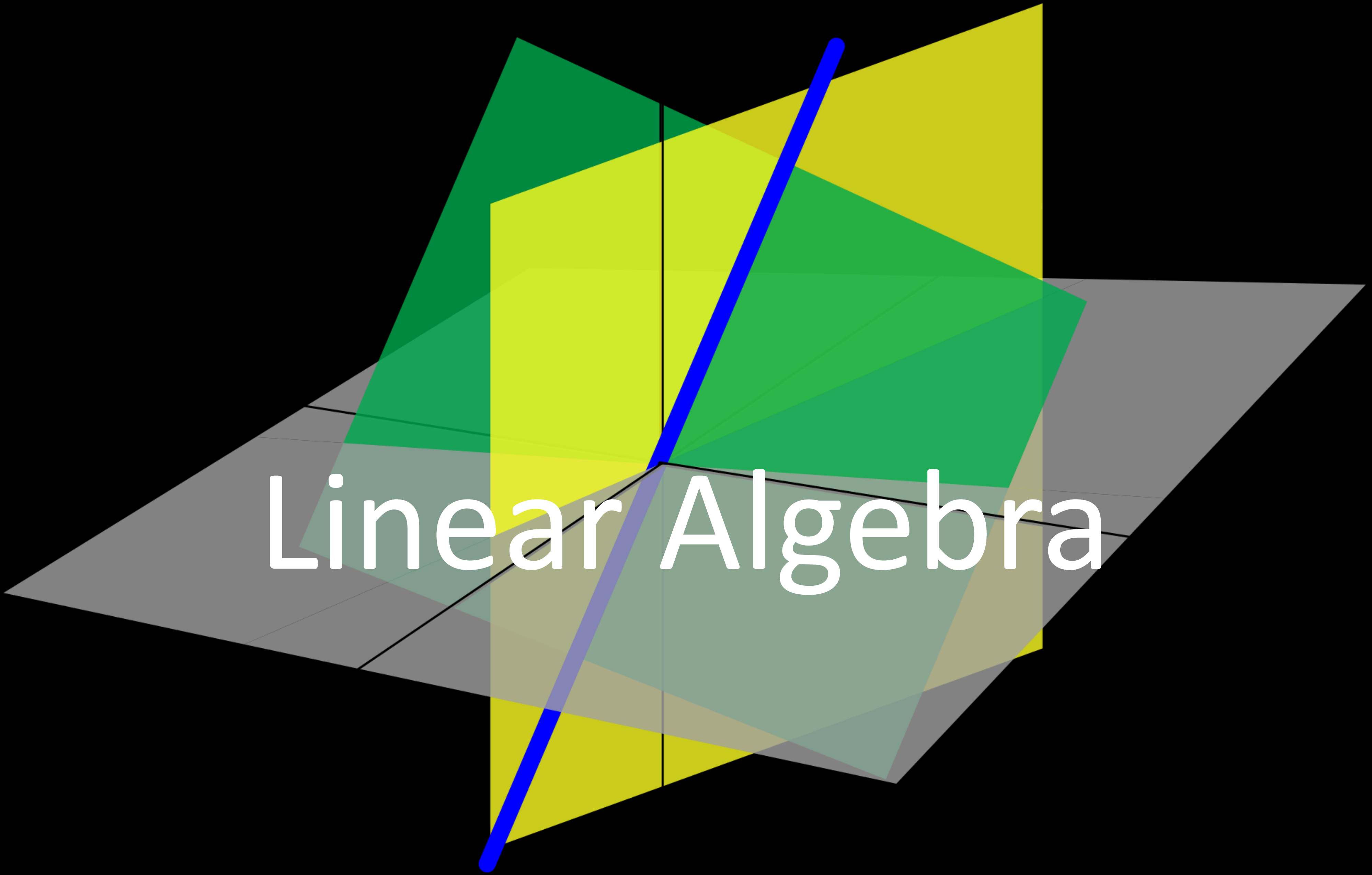
Prof. Kuan-Ting Lai

2021/10/17

Applied Math for Machine Learning

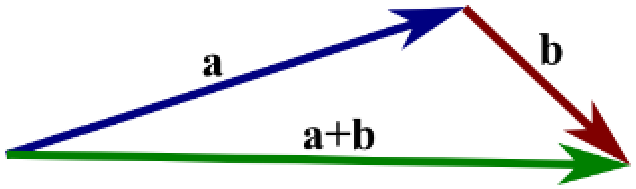
- **Linear Algebra**
- Probability
- Calculus
- Optimization

Linear Algebra



Linear Algebra

- Scalar
  - real numbers
- Vector (1D)
  - Has a magnitude & a direction



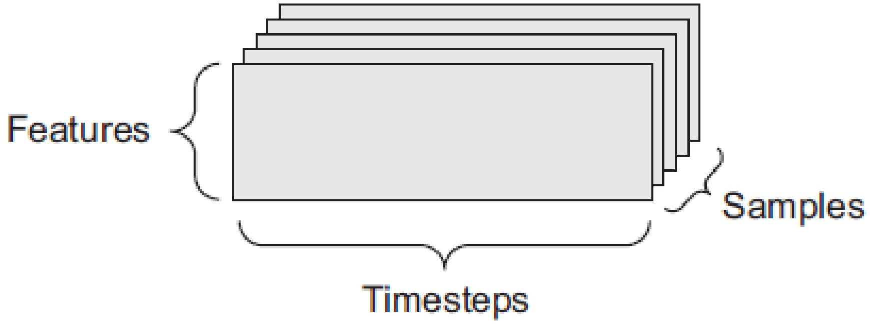
- Matrix (2D)
  - An array of numbers arranges in rows & columns

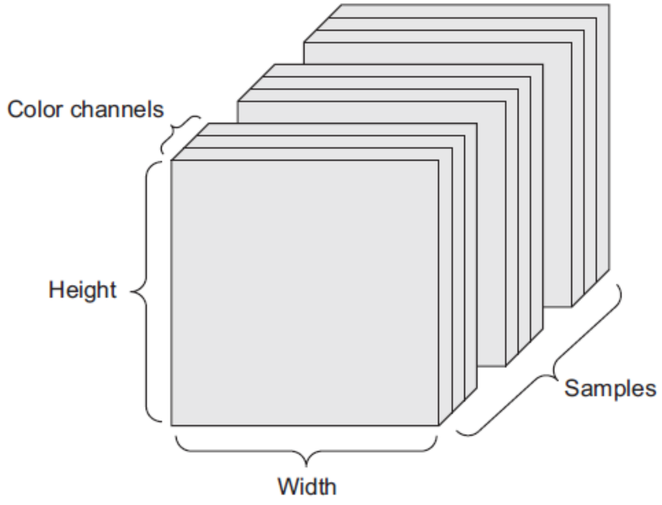
$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

- Tensor (>=3D)
  - Multi-dimensional arrays of numbers

Real-world examples of Data Tensors

- Timeseries Data – 3D (samples, timesteps, features)
- Images – 4D (samples, height, width, channels)
- Video – 5D (samples, frames, height, width, channels)





Vector Dimension vs. Tensor Dimension

- The number of data in a vector is also called “dimension”
- In deep learning , the dimension of Tensor is also called “rank”
- Matrix = 2d array = 2d tensor = rank 2 tensor

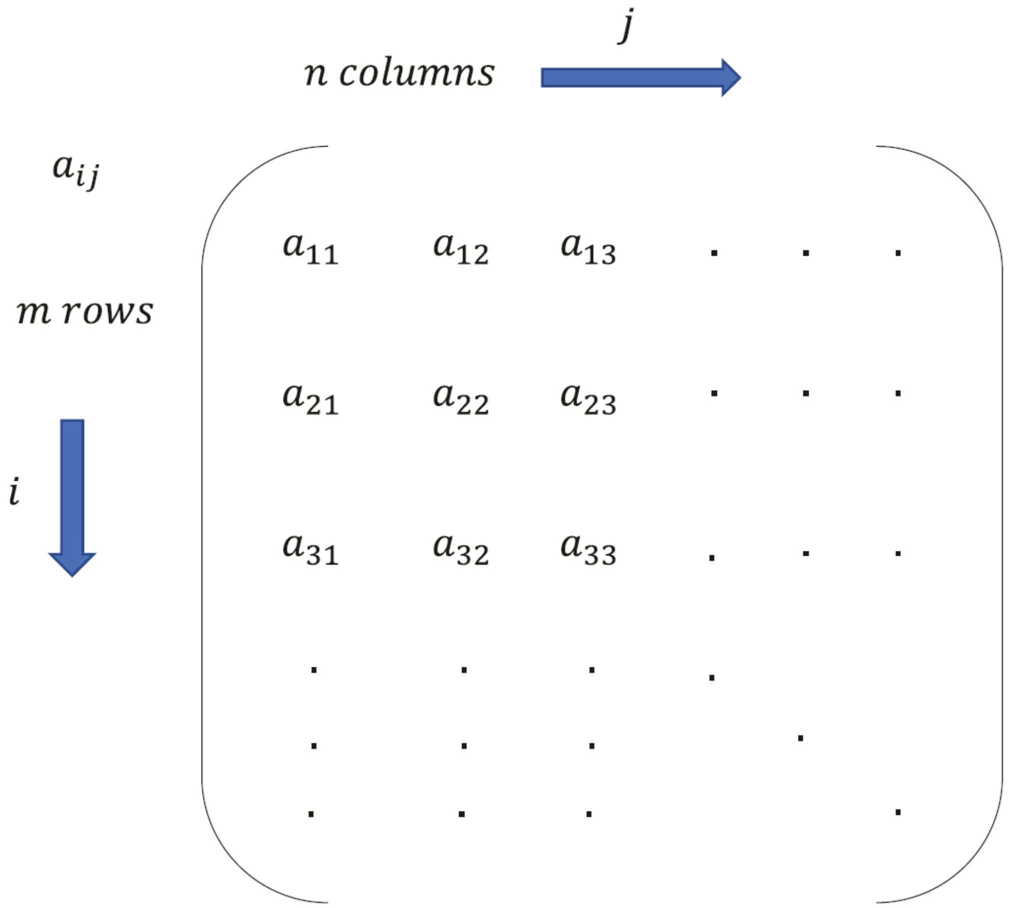
https://deeplizard.com/learn/video/AiyK0idr4uM


The Matrix

Matrix

- Define a matrix with m rows and n columns:

$$A_{m\times n} \in \mathbb{R}^{m\times n}$$



Santanu Pattanayak, "Pro Deep Learning with TensorFlow," Apress, 2017

Matrix Operations

- Addition and Subtraction

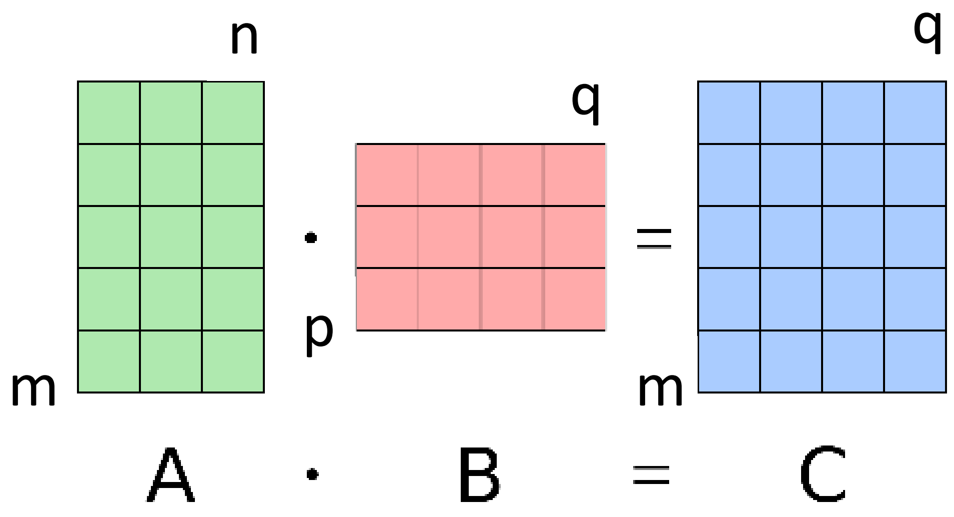
$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

$$A + B = \begin{bmatrix} 1+5 & 2+6 \\ 3+7 & 4+8 \end{bmatrix} = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$$

$$A - B = \begin{bmatrix} 1-5 & 2-6 \\ 3-7 & 4-8 \end{bmatrix} = \begin{bmatrix} -4 & -4 \\ -4 & -4 \end{bmatrix}$$

Matrix Multiplication

- Two matrices A and B, where $A \in \mathbb{R}^{m \times n}$ $B \in \mathbb{R}^{p \times q}$
- The columns of A must be equal to the rows of B, i.e. n == p
- A * B = C, where $C \in \mathbb{R}^{m \times q}$
- $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$



Example of Matrix Multiplication (3-1)

"Dot Product"

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & \\ & \end{bmatrix}$$

$1 \times 7 + 2 \times 9 + 3 \times 11$

$=$

https://www.mathsisfun.com/algebra/matrix-multiplying.html

Example of Matrix Multiplication (3-2)

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ & \end{bmatrix}$$

https://www.mathsisfun.com/algebra/matrix-multiplying.html

Example of Matrix Multiplication (3-3)

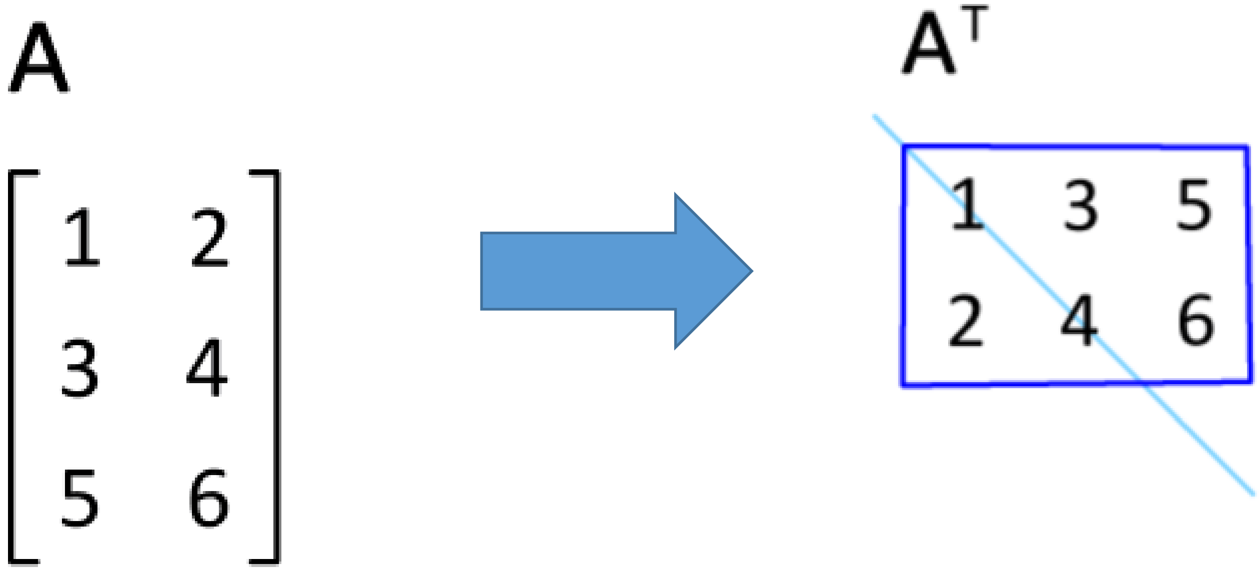
$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix} \checkmark$$

https://www.mathsisfun.com/algebra/matrix-multiplying.html

Matrix Transpose

$$A \in \mathbb{R}^{m \times n} \qquad A^{\mathrm{T}} \in \mathbb{R}^{n \times m}$$

$$a'_{ji} = a_{ij} \quad \forall\, i \in \{1,2,..m\}, \forall\, j \in \{1,2,..n\}$$



https://en.wikipedia.org/wiki/Transpose

Dot Product

- Dot product of two vectors become a **scalar**
- Inner product is a generalization of the dot product
- Notation: $v_1 \cdot v_2$ or $v_1{}^T v_2$

$$v_1 = \begin{bmatrix} v_{11} \\ v_{12} \\ \cdot \\ \cdot \\ \cdot \\ v_{1n} \end{bmatrix} \quad v_2 = \begin{bmatrix} v_{21} \\ v_{22} \\ \cdot \\ \cdot \\ \cdot \\ v_{2n} \end{bmatrix}$$

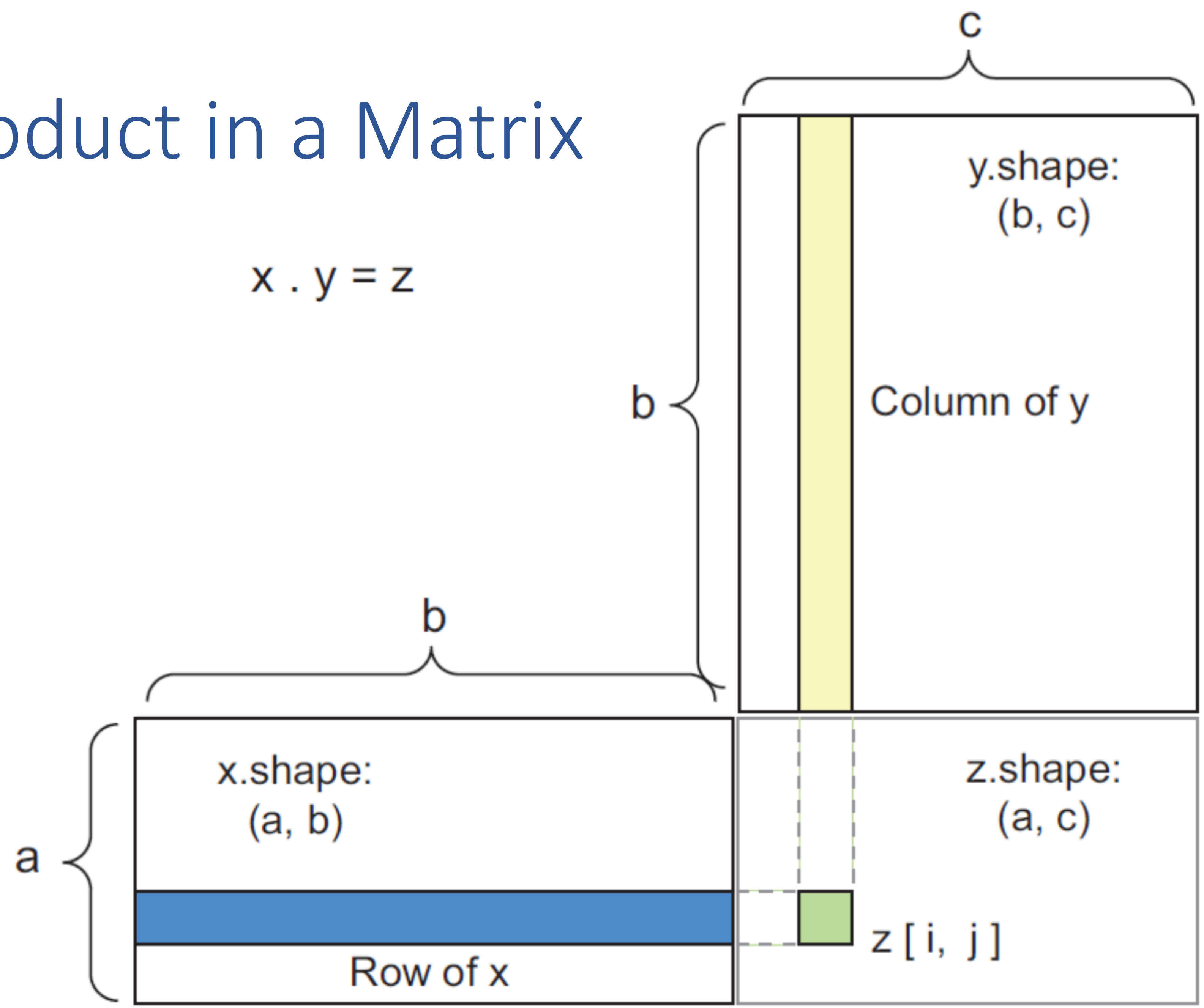
$$v_1 . v_2 = v_1{}^T v_2 = v_2{}^T v_1 = v_{11}v_{21} + v_{12}v_{22} + .. + v_{1n}v_{2n} = \sum_{k=1}^{n} v_{1k}v_{2k}$$

$$V_1^T = [\quad\quad] \begin{bmatrix} \\ \\ \\ \end{bmatrix} = V_1 \cdot V_2$$

V_2

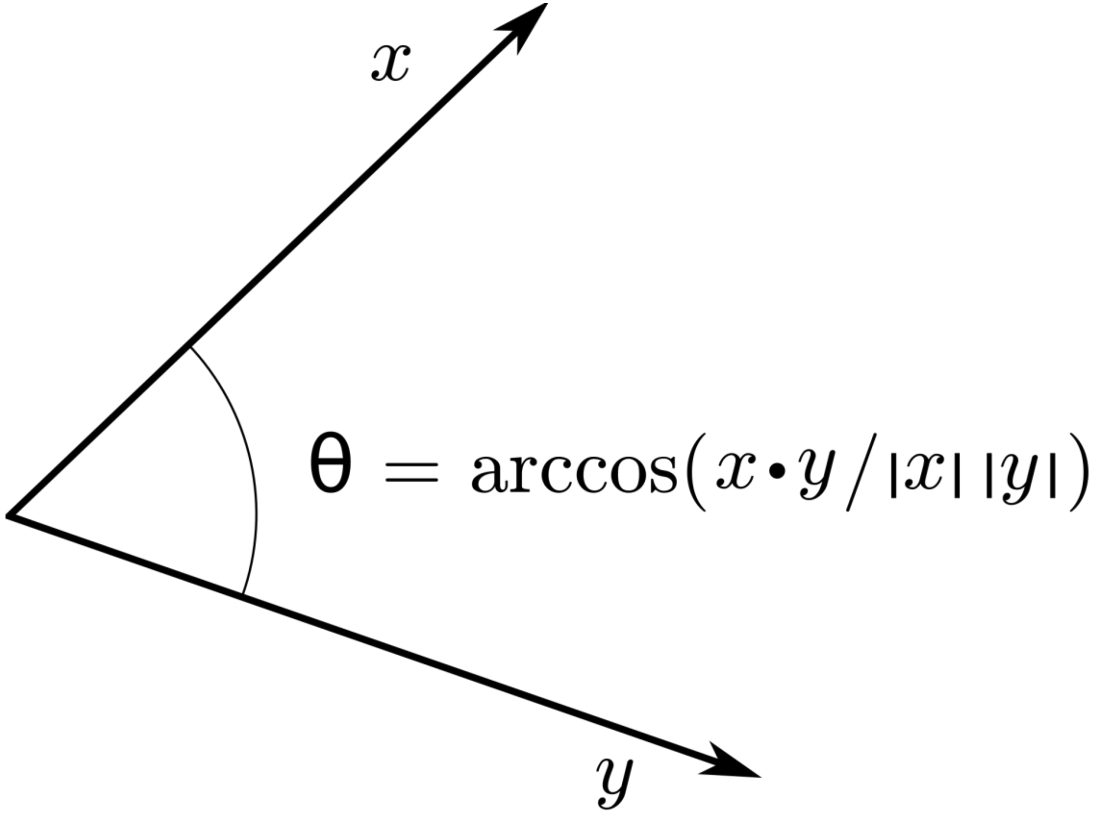
Dot Product in a Matrix

x . y = z



Geometric Definition of Dot Product

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos\theta,$$



Outer Product

$$\mathbf{u} \otimes \mathbf{v} = \mathbf{A} = \begin{bmatrix} u_1 v_1 & u_1 v_2 & \dots & u_1 v_n \\ u_2 v_1 & u_2 v_2 & \dots & u_2 v_n \\ \vdots & \vdots & \ddots & \vdots \\ u_m v_1 & u_m v_2 & \dots & u_m v_n \end{bmatrix}$$

Or in index notation:

$$(\mathbf{u} \otimes \mathbf{v})_{ij} = u_i v_j$$

https://en.wikipedia.org/wiki/Outer_product

Outer Product for Recommendation System

- Collaborative Filtering can be viewed as outer product of user vectors and item vectors

https://developers.google.com/machine-learning/recommendation/collaborative/basics

Linear Independence

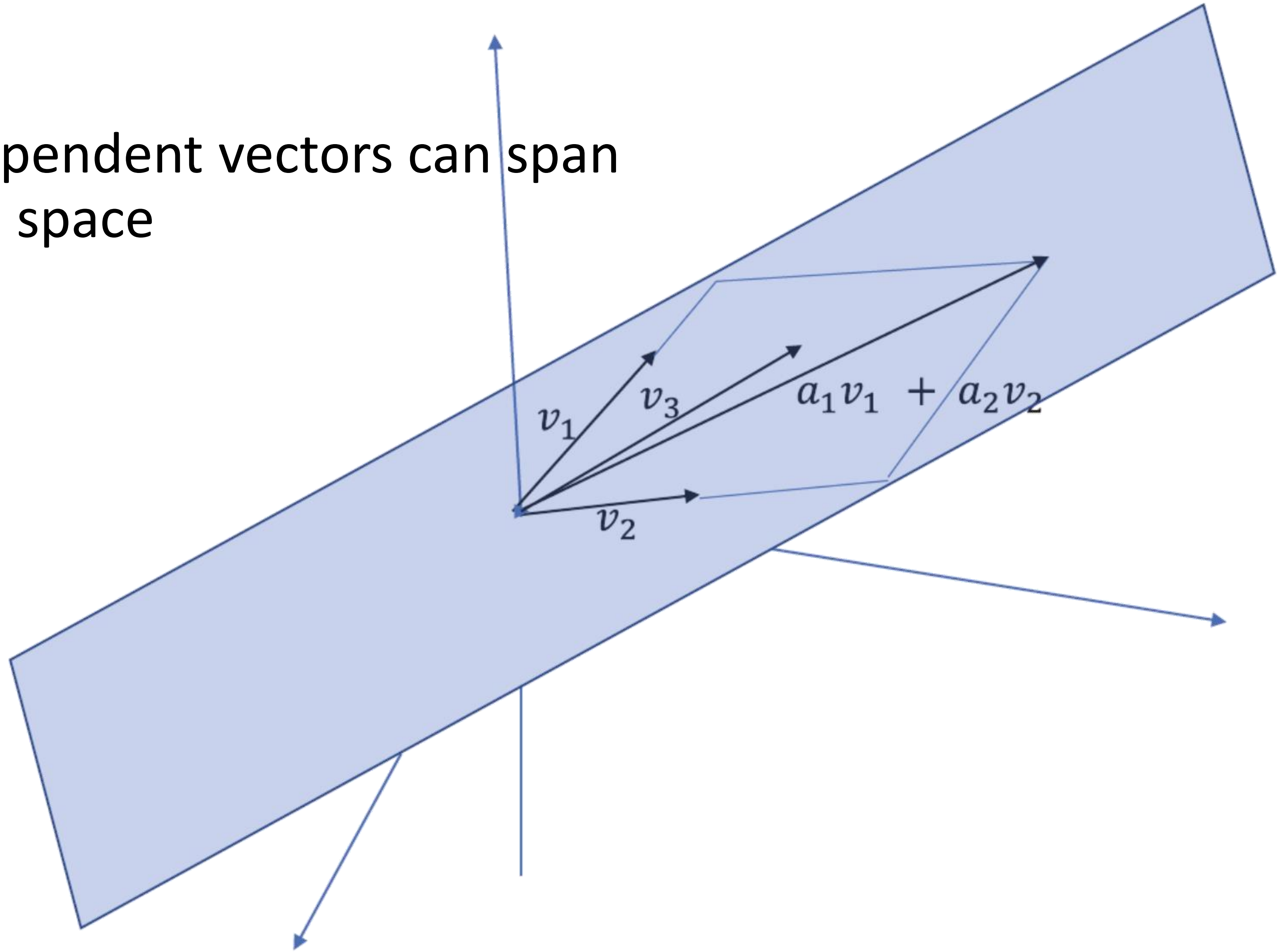
- A vector is **linearly dependent** on other vectors if it can be expressed as the linear combination of other vectors
- A set of vectors $v_1, v_2, \cdots, v_n$ is **linearly independent** if $a_1 v_1 + a_2 v_2 + \cdots + a_n v_n = 0$ implies all $a_i = 0, \forall i \in \{1,2,\cdots n\}$

$$\begin{bmatrix}1\\0\\0\end{bmatrix} \begin{bmatrix}0\\1\\0\end{bmatrix} \begin{bmatrix}0\\0\\1\end{bmatrix}$$

$$[v_1 v_2 \ldots . v_n]\begin{bmatrix}a_1\\a_2\\.\\.\\a_n\end{bmatrix} = 0 \; where \; v_i \in \mathbb{R}^{m\times 1} \; \forall i \in \{1,2,\ldots,n\}, \begin{bmatrix}a_1\\a_2\\.\\.\\a_n\end{bmatrix} \in \mathbb{R}^{n\times 1}$$

Span the Vector Space

- *n* linearly independent vectors can span *n*-dimensional space



Rank of a Matrix

- Rank is:
  - The number of linearly independent row or column vectors
  - The dimension of the vector space generated by its columns
- Row rank = Column rank
- Example:

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

Row-echelon form

$$\begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & 3 \\ 0 & 0 & 0 \end{bmatrix}$$

https://en.wikipedia.org/wiki/Rank_(linear_algebra)

Identity Matrix I

- Any vector or matrix multiplied by I remains unchanged
- For a matrix A, $AI = IA = A$

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3}$$

$$Iv = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$$

Inverse of a Matrix

- The product of a square matrix A and its inverse matrix A^{-1} produces the identity matrix I
- $AA^{-1} = A^{-1}A = I$
- Inverse matrix is square, but not all square matrices has inverses

Pseudo Inverse

- Non-square matrix and have left-inverse or right-inverse matrix
- Example:

$$Ax = b, A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^n$$

  – Create a square matrix $A^T A$

$$A^T A x = A^T b$$

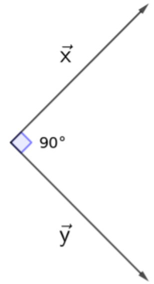
  – Multiplied both sides by inverse matrix $(A^T A)^{-1}$

$$x = (A^T A)^{-1} A^T b$$

  – $(A^T A)^{-1} A^T$ is the pseudo inverse function

Special Vectors and Matrices

- Symmetric matrix: $A = A^T$
- Unit vector: $\|x\|_2 = 1$
- Vector $\boldsymbol{x}$ and $\boldsymbol{y}$ are orthogonal if $\boldsymbol{x}^T\boldsymbol{y} = 0$
  - and if $\|\boldsymbol{x}\|_2 = 1$ and $\|\boldsymbol{y}\|_2 = 1$ => orthonormal



$$\vec{x} \cdot \vec{y} = |\vec{x}||\vec{y}|\cos(90^\circ) = 0$$

- Orthogonal matrix:
  - A square matrix whose rows and columns are mutually orthonormal

$$A^T A = A A^T = I$$

$$\Rightarrow \quad A^{-1} = A^T$$

https://medium.com/linear-algebra/part-23-orthonormal-vectors-orthogonal-matrices-and-hadamard-matrix-bee6857c05c

Norms

- Norm is a measure of a vector's magnitude
- l_2 norm

$$\|x\|_2 = \left(|x_1|^2 + |x_2|^2 + \ldots + |x_n|^2\right)^{1/2} = (x.x)^{1/2} = \left(x^T x\right)^{1/2}$$

- l_1 norm

$$\|x\|_1 = |x_1| + |x_2| + \ldots + |x_n|$$

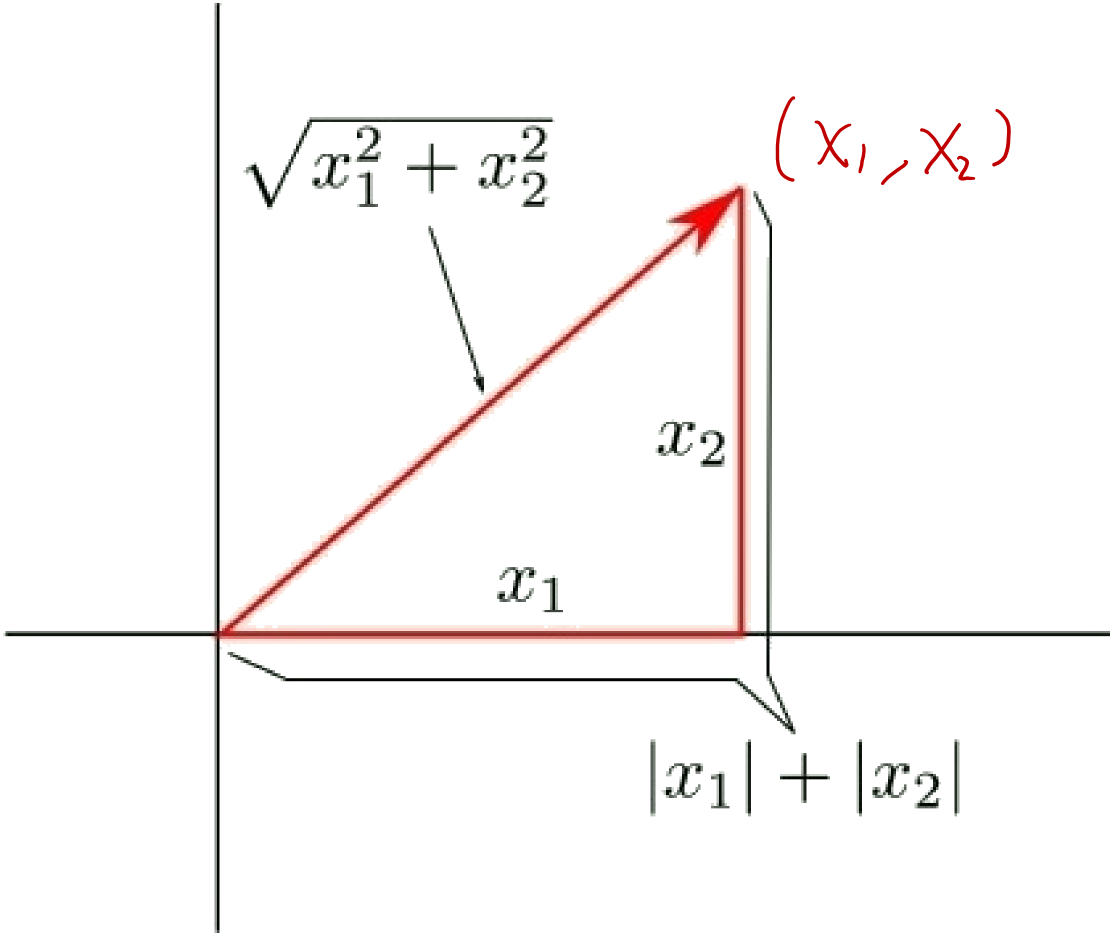
- l_p norm

$$\left(|x_1|^p + |x_2|^p + \ldots + |x_n|^p\right)^{1/p}$$

- l_∞ norm

$$\lim_{p\to\infty} \|x\|_p = \lim_{p\to\infty}\left(|x_1|^p + |x_2|^p + \ldots + |x_n|^p\right)^{1/p} = max(x_1, x_2, \ldots., x_n)$$

Compare l_1 norm and l_2 norm



https://www.slideshare.net/AndresMendezVazquez/03-machine-learning-linear-algebra

Formal Definition of a Norm

- Triangular inequality: $f(x+y) \leq f(x) + f(y)$
- Absolute homogeneity: $f(\alpha x) = |\alpha| f(x), \forall \alpha \in \mathbb{R}$
- Positive definiteness: $f(x) = 0 \Rightarrow x = 0$

https://en.wikipedia.org/wiki/Norm_(mathematics)

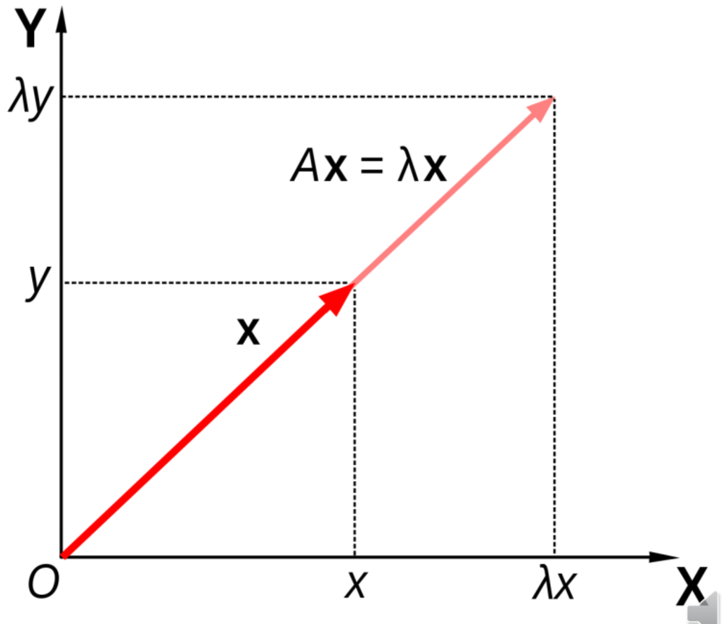
Eigenvector

- Eigenvector is a non-zero vector that changed by only a scalar factor λ when linear transformation A is applied to:

$$Ax = \lambda x, A \in \mathbb{R}^{n \times n}, x \in \mathbb{R}^{n}$$

where x is an Eigenvector and λ is an Eigenvalue

- Important in machine learning, ex:
  - Principle Component Analysis (PCA)
  - Eigenvector centrality
  - PageRank
  - …



Characteristic Polynomial

not invertible ⇒ det $(A - \lambda I)$

$$A\mathbf{v} = \lambda\mathbf{v}, \Rightarrow (A - \lambda I)\,\mathbf{v} = \mathbf{0},$$

- Calculate Eigenvalues and Eigenvectors of A:

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}.$$

- Characteristics polynomial

$$|A - \lambda I| = \begin{vmatrix} 2 - \lambda & 1 \\ 1 & 2 - \lambda \end{vmatrix} = 3 - 4\lambda + \lambda^2.$$

$$= (2-\lambda)^2 - 1$$

- Solve the polynomial:

$$\lambda_1 = 1, \quad \lambda_2 = 3 \Rightarrow \mathbf{v}_{\lambda=1} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad \mathbf{v}_{\lambda=3} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

https://en.wikipedia.org/wiki/Eigenvalues_and_eigenvectors

Power Iteration Method for Computing Eigenvector

1. Start with random vector v
2. Calculate iteratively: $v^{(k+1)} = A^k v$
3. After v^k converges, $v^{(k+1)} \cong v^k$
4. v^k will be the Eigenvector with largest Eigenvalue

$A v^k \cong v^k$

$\Uparrow$

$\Rightarrow A^k v = A^{k-1} v$

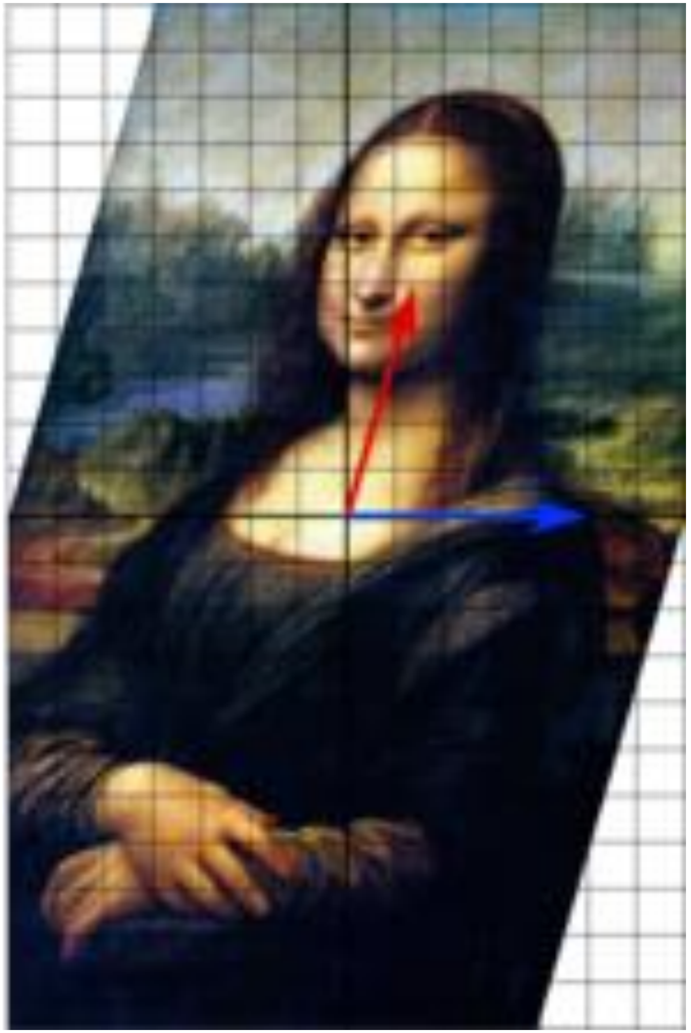
Example: Shear Mapping

- Horizontal axis is the Eigenvector

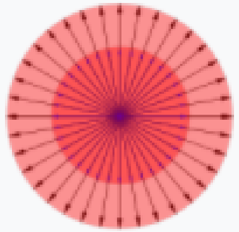
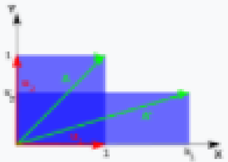
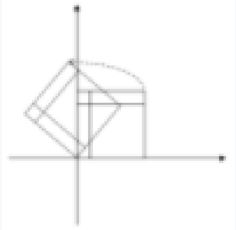
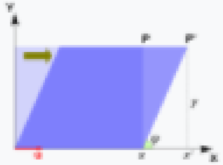
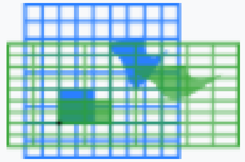
A = $\begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix}$

$$(\lambda - 1)^2$$

$$\mathbf{u}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$



Eigenvalues of Geometric Transformations

	Scaling	Unequal scaling	Rotation	Horizontal shear	Hyperbolic rotation
Illustration					
Matrix	$\begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix}$	$\begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix}$	$\begin{bmatrix} c & -s \\ s & c \end{bmatrix}$ $c = \cos\theta$ $s = \sin\theta$	$\begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} c & s \\ s & c \end{bmatrix}$ $c = \cosh\varphi$ $s = \sinh\varphi$
Characteristic polynomial	$(\lambda - k)^2$	$(\lambda - k_1)(\lambda - k_2)$	$\lambda^2 - 2c\lambda + 1$	$(\lambda - 1)^2$	$\lambda^2 - 2c\lambda + 1$
Eigenvalues, λ_i	$\lambda_1 = \lambda_2 = k$	$\lambda_1 = k_1$ $\lambda_2 = k_2$	$\lambda_1 = e^{i\theta} = c + si$ $\lambda_2 = e^{-i\theta} = c - si$	$\lambda_1 = \lambda_2 = 1$	$\lambda_1 = e^{\varphi} = c + s$ $\lambda_2 = e^{-\varphi} = c - s$
Algebraic mult., $\mu_i = \mu(\lambda_i)$	$\mu_1 = 2$	$\mu_1 = 1$ $\mu_2 = 1$	$\mu_1 = 1$ $\mu_2 = 1$	$\mu_1 = 2$	$\mu_1 = 1$ $\mu_2 = 1$
Geometric mult., $\gamma_i = \gamma(\lambda_i)$	$\gamma_1 = 2$	$\gamma_1 = 1$ $\gamma_2 = 1$	$\gamma_1 = 1$ $\gamma_2 = 1$	$\gamma_1 = 1$	$\gamma_1 = 1$ $\gamma_2 = 1$
Eigenvectors	All nonzero vectors	$\mathbf{u}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\mathbf{u}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$	$\mathbf{u}_1 = \begin{bmatrix} 1 \\ -i \end{bmatrix}$ $\mathbf{u}_2 = \begin{bmatrix} 1 \\ +i \end{bmatrix}$	$\mathbf{u}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$	$\mathbf{u}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ $\mathbf{u}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$

Eigen decomposition

- Let **A** be a square $n \times n$ matrix with n linearly independent eigenvectors q_i (where $i = 1, ..., n$). Then **A** can be factorized as

$$A = Q\Lambda Q^{-1}$$

- **Q** is the square $n \times n$ matrix whose ith column is the eigenvector q_i of **A**
- **Λ** is the diagonal matrix whose diagonal elements are the eigenvalues $\Lambda_{ii} = \lambda_i$.

https://en.wikipedia.org/wiki/Eigendecomposition_of_a_matrix

Calculating Eigendecomposition

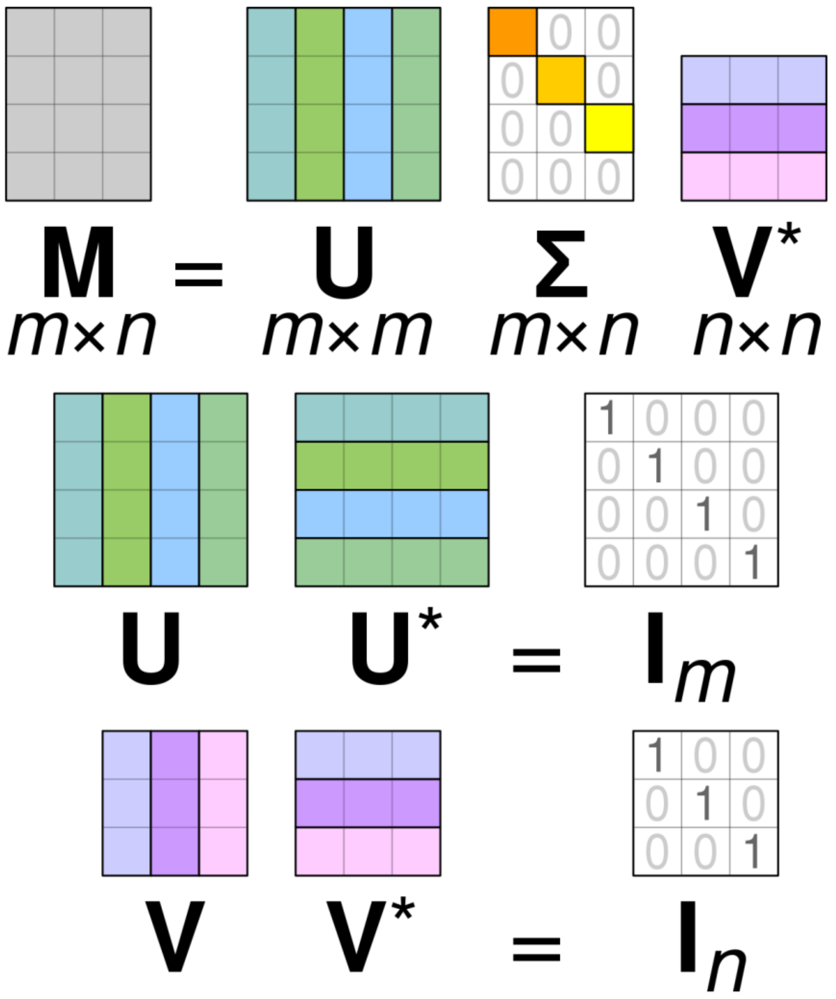
- Reformulate: $Q^{-1}AQ = \Lambda$
- Suppose $\mathrm{A} = \begin{bmatrix} 1 & 0 \\ 1 & 3 \end{bmatrix}, Q = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \Lambda = \begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix}$
- Solve $$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix},$$

https://en.wikipedia.org/wiki/Eigendecomposition_of_a_matrix

Singular Value Decomposition (SVD)

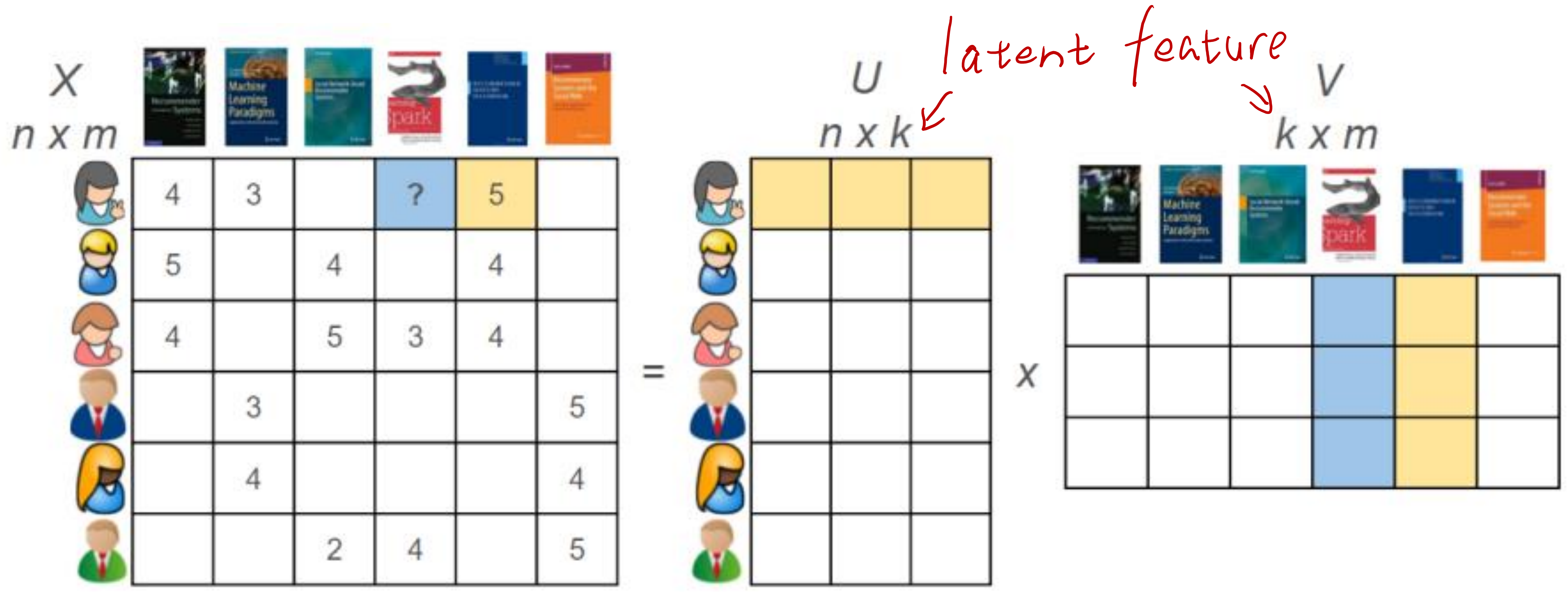
- Factorize matrix into **singular vectors** and **singular values**
- Every real matrix has a SVD

$$A = UDV^T = U\Sigma V^*$$



https://en.wikipedia.org/wiki/Singular_value_decomposition

SVD for Recommender System

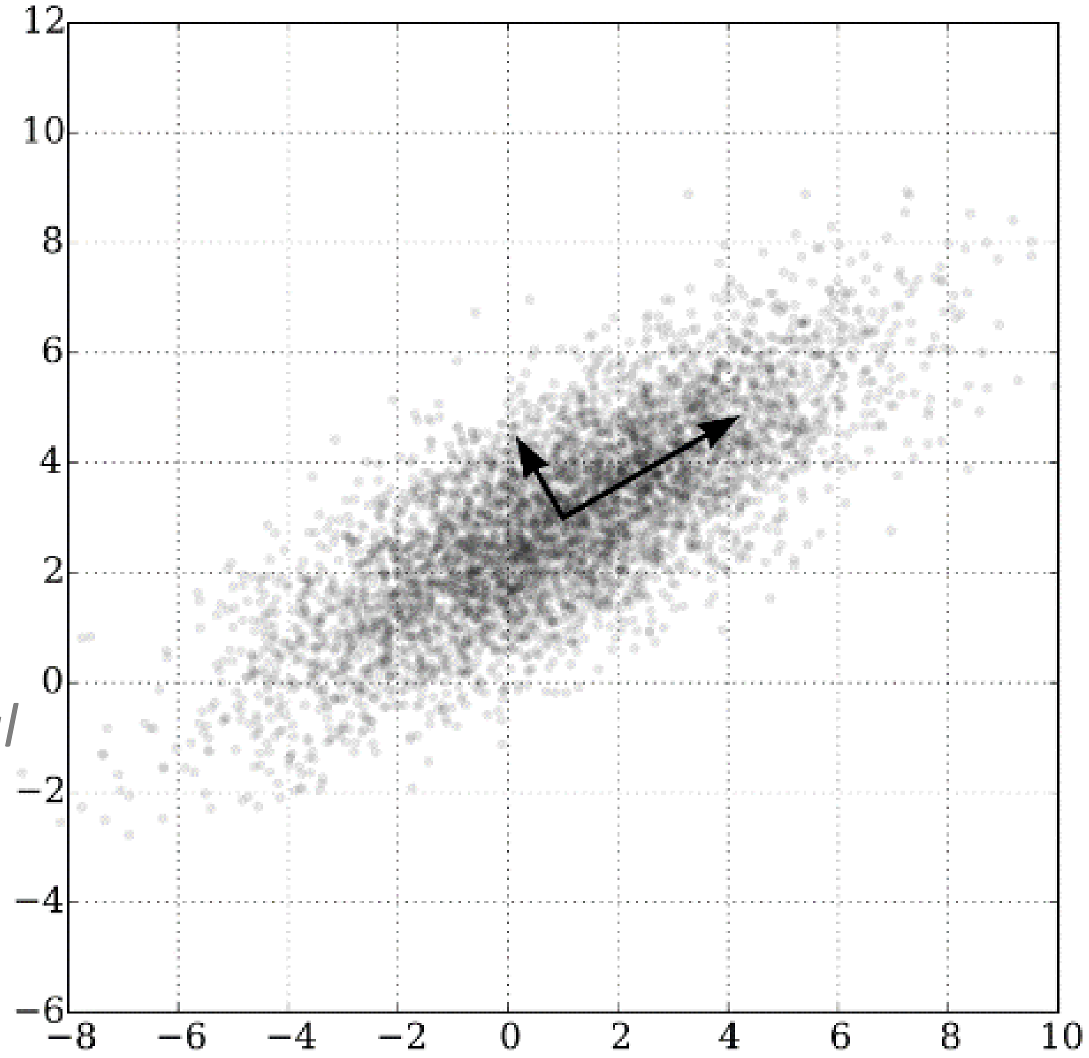


4	3		?	5	
5		4		4	
4		5	3	4	
	3				5
	4				4
		2	4		5

https://heartbeat.comet.ml/recommender-systems-with-python-part-iii-collaborative-filtering-singular-value-decomposition-5b5dcb3f242b

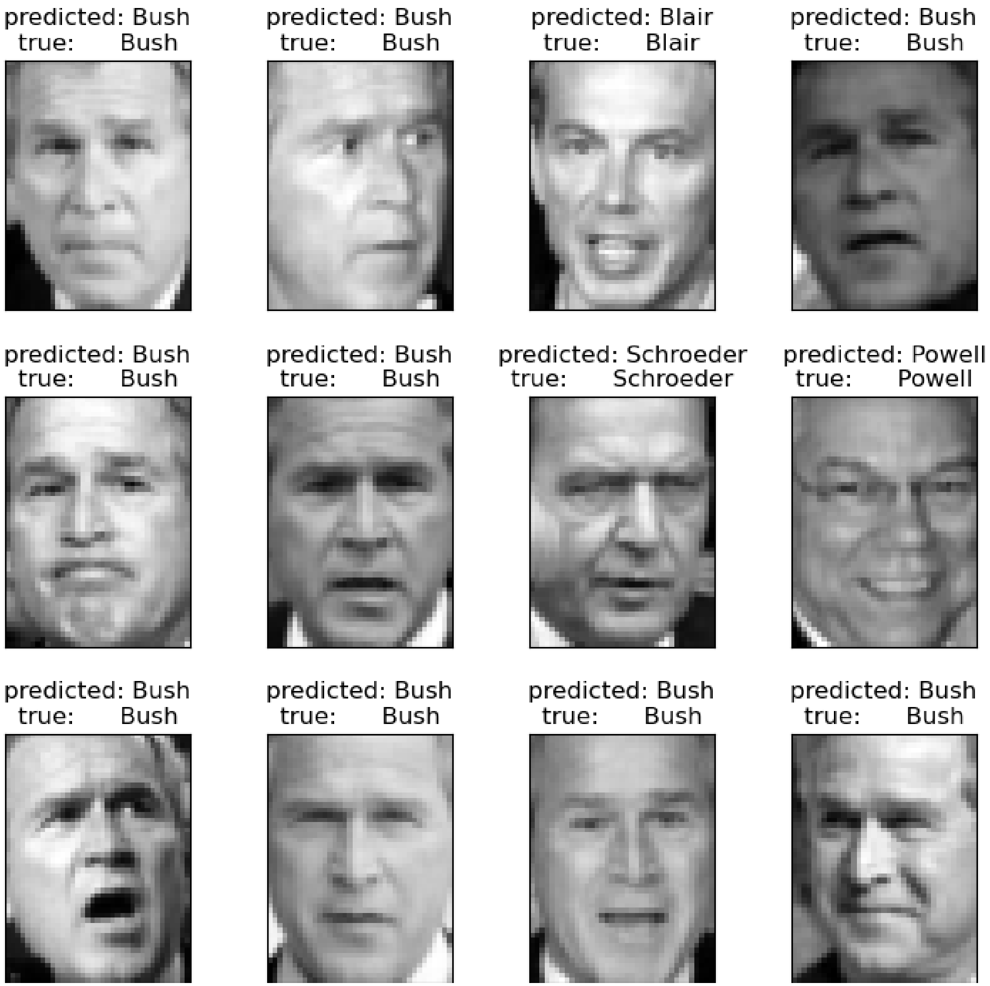
Principle Component Analysis (PCA)

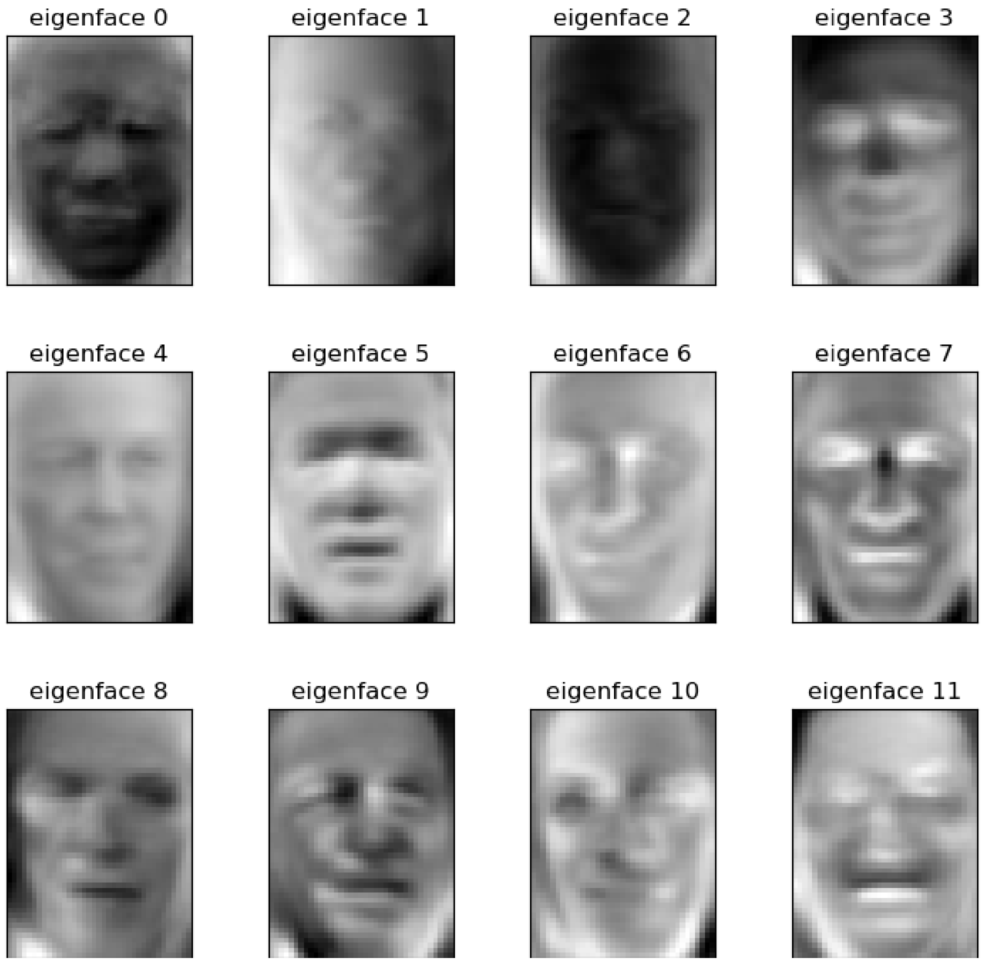
- Find the important (principle) axes (components)
- Used for Dimensionality Reduction
- Assumptions
  - *Linearity*
  - *Mean and Variance are sufficient statistics*
  - *The principal components are orthogonal*



https://en.wikipedia.org/wiki/Principal_component_analysis

Face Recognition using Eigenfaces and SVM





https://scikit-learn.org/stable/auto_examples/applications/plot_face_recognition.html#sphx-glr-auto-examples-applications-plot-face-recognition-py

NumPy for Linear Algebra

- NumPy is the fundamental package for scientific computing with Python.
  - a powerful N-dimensional array object
  - sophisticated (broadcasting) functions
  - tools for integrating C/C++ and Fortran code
  - useful linear algebra, Fourier transform, and random number capabilities

Create Tensors

Scalars (0D tensors)

```
>>> import numpy as np
>>> x = np.array(12)
>>> x
array(12)
>>> x.ndim
0
```

Vectors (1D tensors)

```
>>> x = np.array([12, 3, 6, 14])
>>> x
array([12, 3, 6, 14])
>>> x.ndim
1
```

Matrices (2D tensors)

```
>>> x = np.array([[5, 78, 2, 34, 0],
                  [6, 79, 3, 35, 1],
                  [7, 80, 4, 36, 2]])
>>> x.ndim
2
```

Create 3D Tensor

```
>>> x = np.array([[[5, 78, 2, 34, 0],
                   [6, 79, 3, 35, 1],
                   [7, 80, 4, 36, 2]],
                  [[5, 78, 2, 34, 0],
                   [6, 79, 3, 35, 1],
                   [7, 80, 4, 36, 2]],
                  [[5, 78, 2, 34, 0],
                   [6, 79, 3, 35, 1],
                   [7, 80, 4, 36, 2]]])
>>> x.ndim
3
```

Attributes of a Numpy Tensor

- Number of axes (dimensions, rank)
  - x.ndim
- Shape
  - This is a tuple of integers showing how many data the tensor has along each axis
- Data type
  - uint8, float32 or float64

Numpy Multiplication

In [ ]:
```
import numpy as np
```

In [4]:
```
x = np.array([[1, 2, 3], [4, 5, 6]])
x
```

Out[4]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [5]:
```
y = np.array([[7, 8], [9, 10], [11, 12]])
y
```

Out[5]:
```
array([[ 7,  8],
       [ 9, 10],
       [11, 12]])
```

In [10]:
```
np.matmul(x, y)
```

Out[10]:
```
array([[ 58,  64],
       [139, 154]])
```

Unfolding the Manifold

- Tensor operations are complex geometric transformation in high-dimensional space
  - Dimension reduction