ADIA Course @ Sorbonne

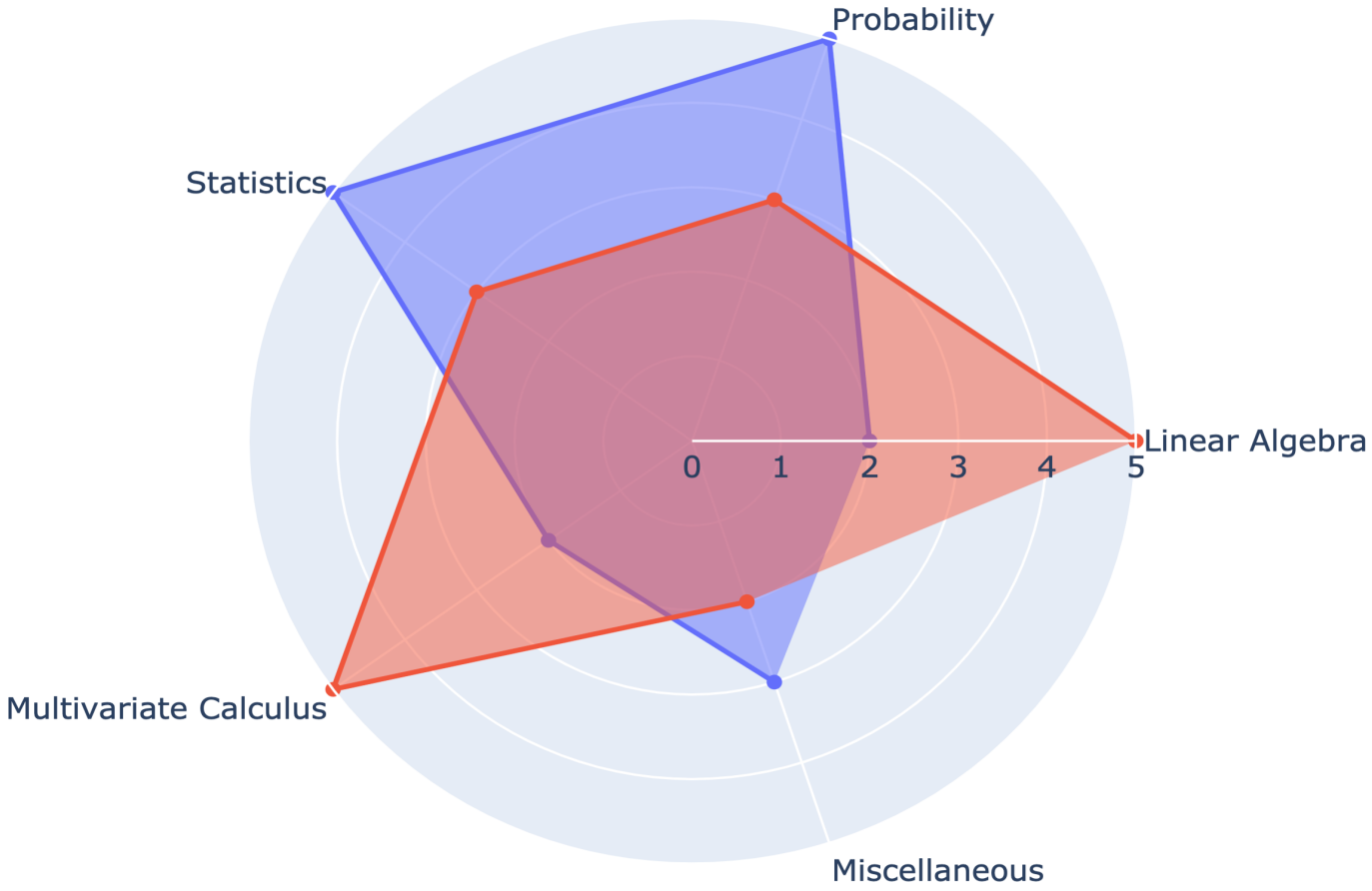
Linear Algebra Applications to Digital Image Processing

Ms. Madhurima Panja
Sorbonne University Abu Dhabi.
madhurima.panja@sorbonne.ae

https://github.com/scaiworkshop/ADIA_AMF

ADIA Certificate Course

Roots of Machine Learning



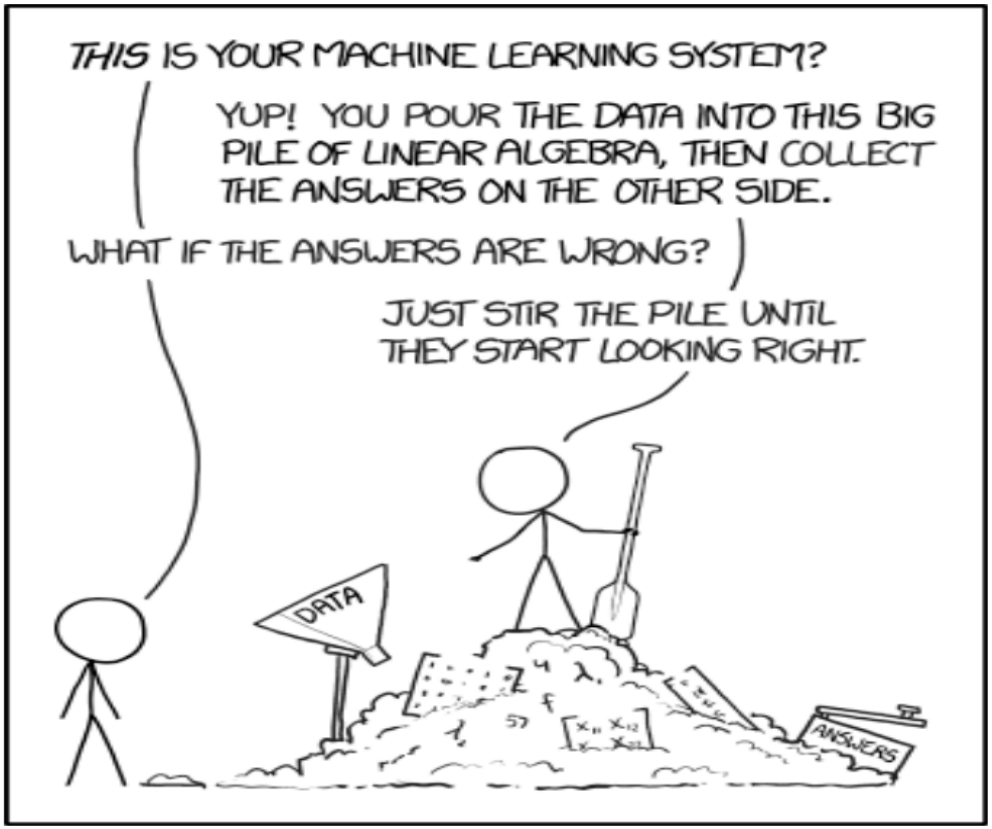
Linear Algebra to Statistical Learning

Linear Algebra to Image Processing

Linear Algebra to Statistical Learning

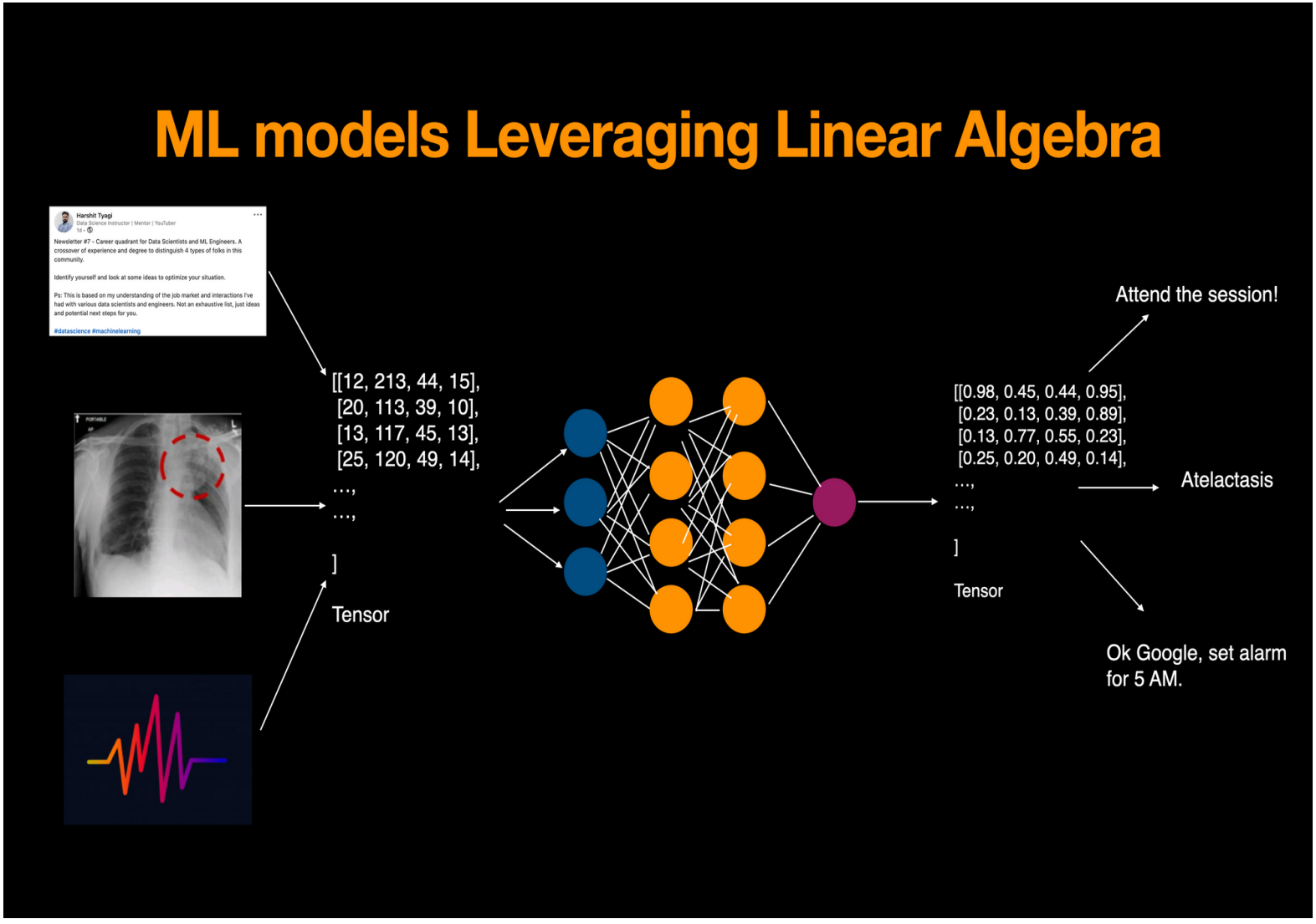
Importance of Linear Algebra in ML

Importance in ML: We convert input vectors $(x_1, .., x_n)$ into outputs by a series of linear transformations.



But what is RIGHT? And is that enough? (Image: Machine Learning, XKCD)

ML Models leveraging Linear Algebra



Why do we need to know Linear Algebra?

- Linear Algebra is used throughout engineering - Because it is based on continuous math rather than discrete math.
- Computer scientists have little experience with it
- Essential for understanding ML algorithms - E.g., We convert input vectors $(x_1, \ldots, x_n)$ into outputs by a series of linear transformations.

Here we discuss:

- Concepts of linear algebra needed for ML
- Omit other aspects of linear algebra

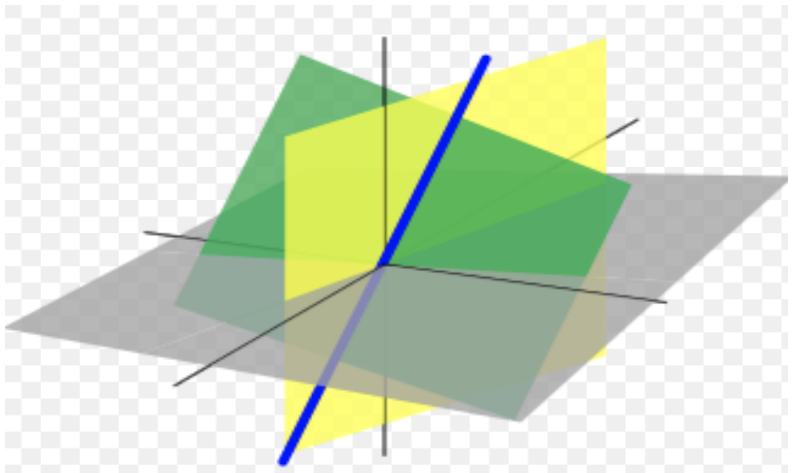
What is linear algebra?

- Linear algebra is the branch of mathematics concerning linear equations such as

$$a_1 x_1 + \ldots + a_n x_n = b$$

  - In vector notation we say $a^T x = b$
  - Called a linear transformation of x
- Linear algebra is fundamental to geometry, for defining objects such as lines, planes, rotations.

- Linear equation $a_1 x_1 + \ldots + a_n x_n = b$ defines a plane in $(x_1, \ldots, x_n)$ space Straight lines define common solutions to equations.



Standard Definitions

- Scalar: Single number (represented by x)
  - In contrast to other objects in linear algebra, which are usually arrays of numbers.
  - They can be real-valued or be integers.
- Vector: An array of numbers arranged in order (represented by $\mathbf{x}$)

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

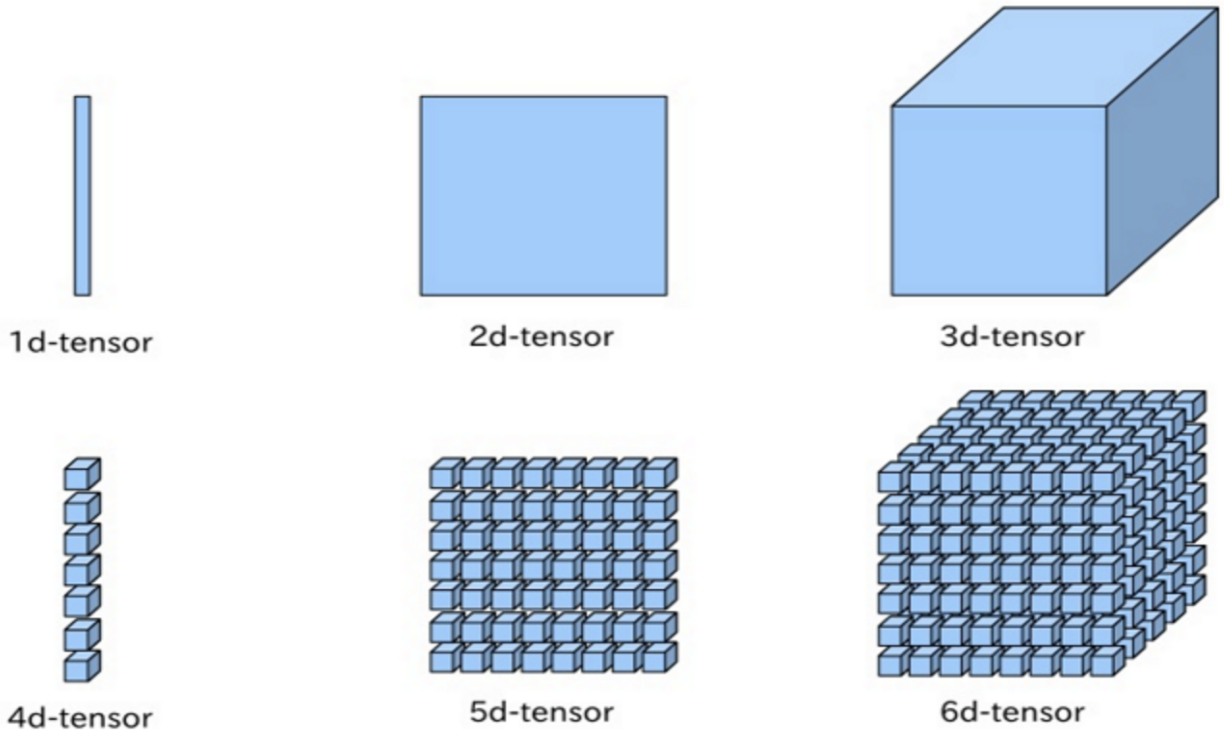
  - If each element is in $\mathbb{R}$ then $\mathbf{x}$ is in $\mathbb{R}^n$.
  - We can think of vectors as points in space where each element gives coordinate along an axis.

Standard Definitions

- Matrices: 2-D array of numbers
  - So each element identified by two indices.
  - E.g.,
$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}$$
  - where $A_{i:}$ is i^{th} row of A, $A_{:j}$ is j^{th} column of A.
  - If A has shape of height m and width n with real-values then $A \in \mathbb{R}^{m \times n}$.

- Tensor: A tensor is an array of numbers arranged on a regular grid with variable number of axes.
  - Sometimes we need an array with more than two axes.
  - E.g., an RGB color image has three axes.
  - Element (i, j, k) of tensor denoted by $A_{i,j,k}$.

Shapes of Tensors



Matrix Addition

- We can add matrices to each other if they have the same shape, by adding corresponding elements
  - If A and B have same shape (height m, width n)

$$C = A + B \implies C_{i,j} = A_{i,j} + B_{i,j}$$

- A scalar can be added to a matrix or multiplied by a scalar

$$D = aB + c \implies D_{i,j} = aB_{i,j} + c$$

- Less conventional notation used in ML:
  - Vector added to matrix

$$C = A + b \implies C_{i,j} = A_{i,j} + b_j$$

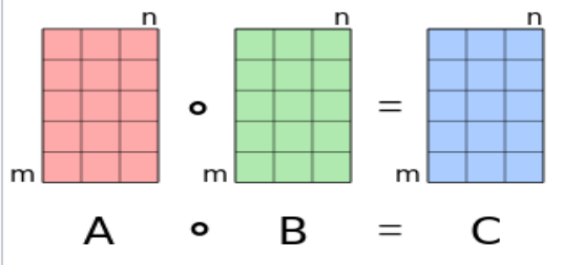
    - Called broadcasting since vector b added to each row of A

Multiplying Matrices

- For product $C = AB$ to be defined, A has to have the same no. of columns as the no. of rows of B
  - If A is of shape mxn and B is of shape nxp then matrix product C is of shape mxp

$$C = AB \implies C_{i,j} = \sum_k A_{i,k} B_{k,j}$$

- Note that the standard product of two matrices is not just the product of two individual elements.
- Such a product does exist and is called the element-wise product or the Hadamard product $A \odot B$.



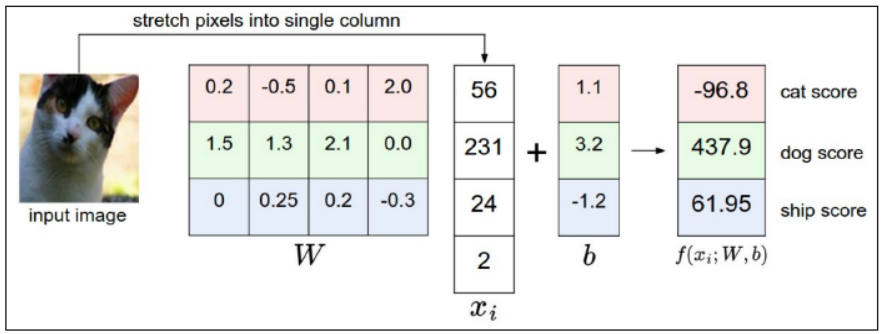
The Hadamard product operates on identically shaped matrices and produces a third matrix of the same dimensions.

Wikipedia

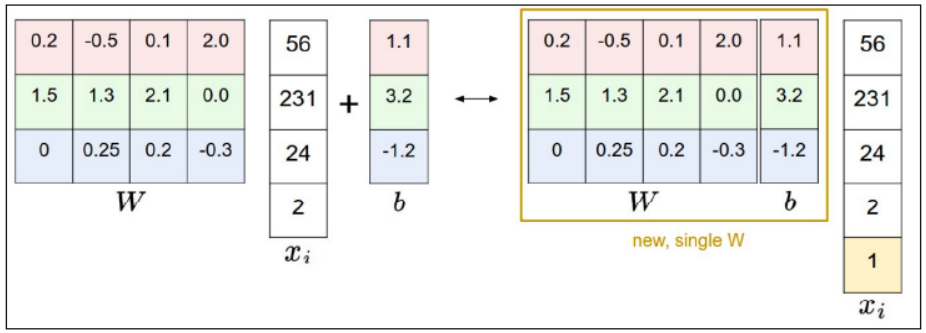
Tensors and Linear Classifier

Vector x is converted into vector y by multiplying x by a matrix W

A linear classifier $y = Wx^{\mathrm{T}} + b$



A linear classifier with bias eliminated $y = Wx^{\mathrm{T}}$



Slide Credit: Sargur N. Srihari (Hari): https://cedar.buffalo.edu/~srihari/

Linear Transformation

- Ax=b
  - where $A \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$
  - More explicitly (n equations in n unknowns)

$$A_{1,1}x_1 + A_{1,2}x_2 + \ldots + A_{1,n}x_n = b_1$$
$$A_{2,1}x_1 + A_{2,2}x_2 + \ldots + A_{2,n}x_n = b_2$$

$$A_{n,1}x_1 + A_{n,2}x_2 + \ldots + A_{n,n}x_n = b_n$$

- Consider the following:

$$A = \begin{bmatrix} A_{1,1} & \ldots & A_{1,n} \\ \vdots & \vdots & \vdots \\ A_{n,1} & \ldots & A_{n,n} \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

  Can view A as a linear transformation of vector x to vector b

- Sometimes we wish to solve for the unknowns $x = (x_1, \ldots, x_n)$ when A and b provide constraints.

Identity and Inverse Matrices

- Matrix inversion is a powerful tool to analytically solve $Ax = b$
- We can now solve $Ax = b$ as follows:

$$\begin{aligned} Ax &= b \\ A^{-1}Ax &= A^{-1}b \\ I_n x &= A^{-1}b \\ x &= A^{-1}b \end{aligned}$$

- If A^{-1} exists there are several methods for finding it.
- Two closed-form solutions
  1. Matrix inversion $x = A^{-1}b$
  2. Gaussian elimination

Performing Gaussian Elimination

Solve for the system of linear equations using the Gaussian Elimination method:

$$
\begin{aligned}
2x + y - z &= 8 \\
-3x - y + 2z &= -11 \\
-2x + y + 2z &= -3
\end{aligned}
$$

Disadvantage of closed-form solutions

- If A^{-1} exists, the same A^{-1} can be used for any given b
  - But A^{-1} cannot be represented with sufficient precision
  - It is not used in practice
- Gaussian elimination also has disadvantages
  - numerical instability (division by small no.)
  - $O(n^3)$ for $n \times n$ matrix
- Software solutions use value of b in finding x
  - E.g., difference (derivative) between b and output is used iteratively
- Least squares solutions of a $m \times n$ system

$$\underset{m \times n}{A}\, x = b \implies (A^T A)x = A^T b \implies x = (A^T A)^{-1} A^T b = A^+ b.$$

Use of a Vector in Regression

- A design matrix
  - N samples, D features

	# hours studied	# hours playing games	# classes missed	Grade
Student #1	10	3	0	87
Student #2	8	20	2	75
Student #3	5	1	5	63

- Feature vector has three dimensions
- This is a standard regression problem.

Norms

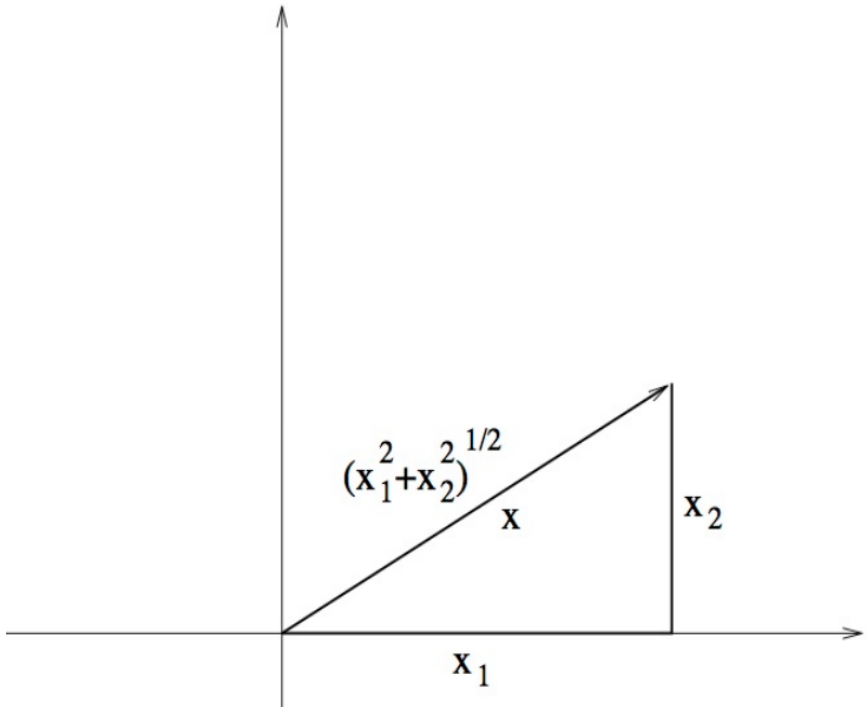
- Used for measuring the size of a vector
- Norms map vectors to non-negative values
- Norm of vector $x = [x_1, \ldots, x_n]^T$ is distance from origin to x

It is any function f that satisfies:

$$f(x) = 0 \implies x = 0$$

$$f(x+y) \leq f(x) + f(y) \quad \text{(Triangle Inequality)}$$

$$\forall\, \alpha \in \mathbb{R},\ \ f(\alpha x) = |\alpha| f(x)$$



L^P Norm

- Definition:

$$\|x\|_p = \left(\sum_i |x_i|^p \right)^{1/p}$$

  - L^2 Norm
    - Called Euclidean norm
    - Simply the Euclidean distance between the origin and the point x
    - written simply as $\|x\|$
    - Squared Euclidean norm is same as $x^T x$
  - L^1 Norm
    - Useful when 0 and non-zero have to be distinguished
    - Note that L^2 increases slowly near origin, e.g., $0.1^2 = 0.01$
  - L^∞ Norm

$$\|x\|_\infty = \max_i |x_i| \quad \text{(called max norm)}$$

Use of norm in Regression

- Linear Regression with x : a vector, w : weight vector

$$y(x,w) = w_0 + w_1 x_1 + \ldots + w_d x_d = w^T x$$

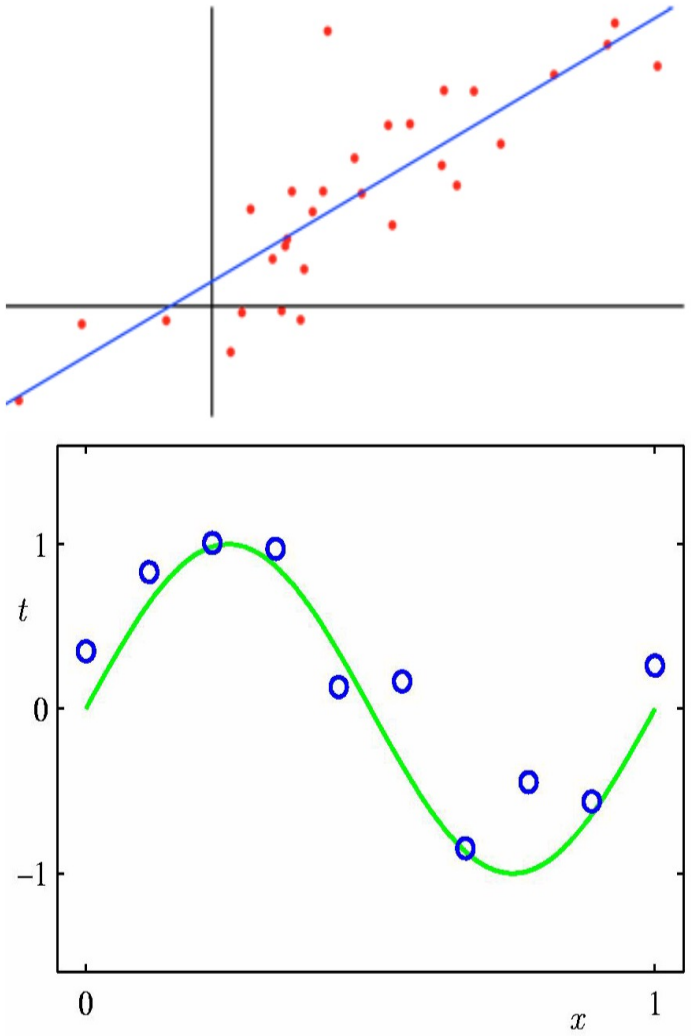
With non-linear basis functions ϕ_j

$$y(x,w) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(x)$$

- Loss Function

$$L(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,w) - t_n\}^2 + \frac{\lambda}{2}\|w^2\|$$

Second term is a weighted norm called a regularizer (to prevent overfitting)



Distance and Frobenius Norm

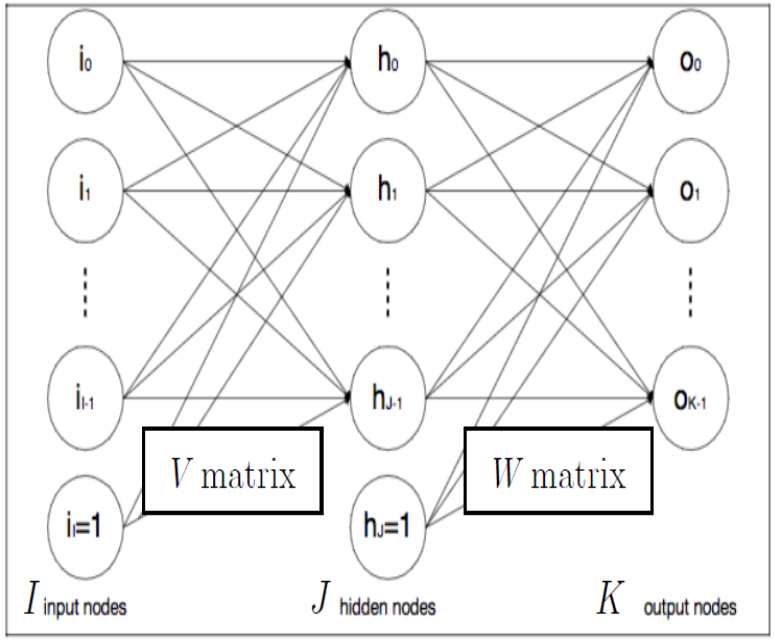
- Norm is the length of a vector
- Distance between two vectors (v, w)
  - $dist(v, w) = \|v - w\| = \sqrt{(v_1 - w_1)^2 + \ldots + (v_n - w_n)^2}$
    Distance to origin would be sqrt of sum of squares
- Similar to L^2 norm

$$\|A\|_F = \left(\sum_{i,j} A_{i,j}^2 \right)^{\frac{1}{2}}$$

- Frobenius in ML
  - Layers of neural network involve matrix multiplication
  - Regularization:
    minimize Frobenius of weight matrices $\|W_{(i)}\|$ over L layers

$$J_R = J + \lambda \sum_{i=1}^{L} \|W^{(i)}\|_F$$

Frobenius in ML



$$I_{1\times(I+1)} \times V_{(I+1)\times J} = \text{net}_J$$

$$h_j = f(\text{net}_j) \qquad f(x) = 1/(1+e^{-x})$$

Fig: Matrix Multiplication in Neural Net.

Matrix Decomposition

- Matrices can be decomposed into factors to learn universal properties, just like integers:
  - Properties not discernible from their representation
  - Decomposition of an integer into prime factors
  - From $12 = 2 \times 2 \times 3$ we can discern that 12 is not divisible by 5 or any multiple of 12 is divisible by 3, but representations of 12 in binary or decimal are different.
- Decomposition of Matrix A as $A = V \operatorname{diag}(\lambda) V^{-1}$, where V is formed of eigenvectors and λ are eigenvalues.

What are Eigen Vectors

- An eigenvector of a square matrix $\boldsymbol{A}$ is a non-zero vector $\boldsymbol{v}$ such that multiplication by $\boldsymbol{A}$ only changes the scale of $\boldsymbol{v}$

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

- The scalar λ is known as eigenvalue
- If $\boldsymbol{v}$ is an eigenvector of $\boldsymbol{A}$, so is any rescaled vector $s\boldsymbol{v}$. Moreover, $s\boldsymbol{v}$ still has the same eigenvalue

What are eigenvalues?

- Given a matrix, A, x is the eigenvector and λ is the corresponding eigenvalue if $Ax = \lambda x$.
  - A must be a square matrix, the determinant of $A - \lambda I$ must be equal to zero.

$$Ax - \lambda x = 0 \implies (A - \lambda I)x = 0$$

    - Trivial solution is if $x = 0$.
    - The non trivial solution occurs when $det(A - \lambda I) = 0$.
- Are eigenvectors unique?
  - If x is an eigenvector, then βx is also an eigenvector and $\beta\lambda$ is an eigenvalue

$$A(\beta x) = \beta(Ax) = \beta(\lambda x) = \lambda(\beta x)$$

Calculating the Eigenvectors/ values

- Expand the $det(A - \lambda I) = 0$ for a 2×2 matrix

$$det(A - \lambda I) = det\left(\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) = 0$$

$$det\begin{bmatrix} a_{11} - \lambda & a_{12} \\ a_{21} & a_{22} - \lambda \end{bmatrix} = 0 \implies (a_{11} - \lambda)(a_{22} - \lambda) - a_{12}a_{21} = 0$$

$$\lambda^2 - \lambda(a_{11} + a_{22}) + (a_{11}a_{22} - a_{12}a_{21}) = 0$$

- For a 2×2 matrix, this is a simple quadratic equation with two solutions (maybe complex).
- The "characteristic equation" can be used to solve for x.

Eigenvalue example

- Consider,

$$A = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \implies \begin{cases} \lambda^2 - (a_{11} + a_{22})\lambda + (a_{11}a_{22} - a_{12}a_{21}) = 0 \\ \lambda^2 - (1+4)\lambda + (1*4 - 2*2) = 0 \\ \lambda^2 = (1+4)\lambda \implies \lambda = 0, \lambda = 5 \end{cases}$$

- The corresponding eigenvectors can be computed as

$$\lambda = 0 \implies \left[\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \right] \cdot \begin{bmatrix} x \\ y \end{bmatrix} = 0 \implies \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1x + 2y \\ 2x + 4y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

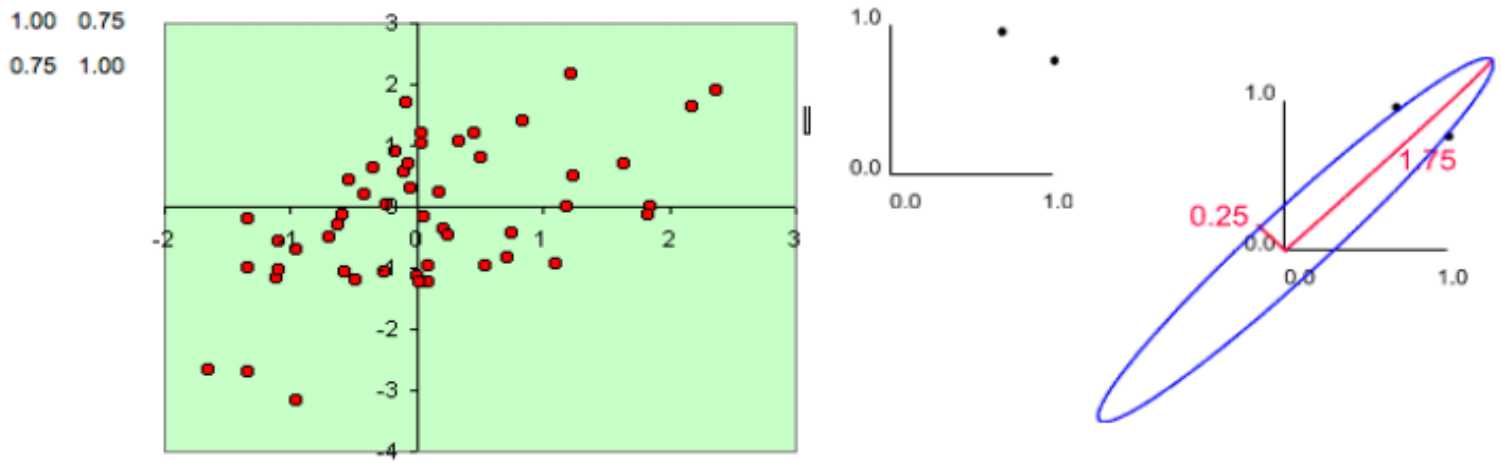
$$\lambda = 5 \implies \left[\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} - \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} \right] \cdot \begin{bmatrix} x \\ y \end{bmatrix} = 0 \implies \begin{bmatrix} -4 & 2 \\ 2 & -1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -4x + 2y \\ 2x - 1y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- For $\lambda = 0$, one possible solution is $x = (2, -1)$
- For $\lambda = 5$, one possible solution is $x = (1, 2)$

Physical Interpretation

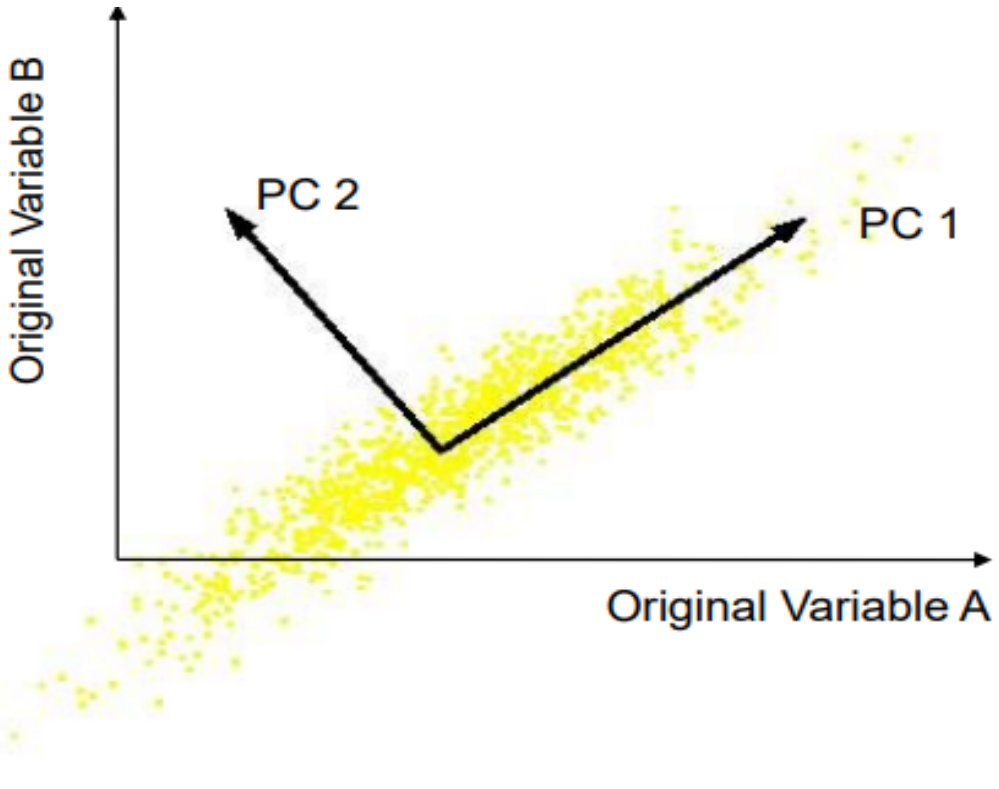
- Consider a correlation matrix, A

$$A = \begin{bmatrix} 1 & 0.75 \\ 0.75 & 1 \end{bmatrix} \implies \lambda_1 = 1.75, \lambda_2 = 0.25$$



- Error ellipse with the major axis as the larger eigenvalue and the minor axis as the smaller eigenvalue.

Physical Interpretation



- Orthogonal directions of greatest variance in data.
- Projections along PC1 (Principal Component) discriminate the data most along any one axis.

Physical Interpretation

- First principal component is the direction of greatest variability (covariance) in the data.
- Second is the next orthogonal (uncorrelated) direction of greatest variability.
    - So first remove all the variability along the first component, and then find the next direction of greatest variability.
- And so on …
- Thus each eigenvectors provides the directions of data variances in decreasing order of eigenvalues.

Spectral Decomposition Theorem

- If A is symmetric and positive definite $k \times k$ matrix $(x^T Ax > 0)$ with λ_i $(\lambda_i > 0)$ and $e_i, i = 1, 2, \ldots, k$ being the k eigenvalues and eigenvectors pairs, then

$$\underset{(k\times k)}{A} = \lambda_1 \underset{(k\times 1)}{e_1} \underset{(1\times k)}{e_1^T} + \lambda_2 \underset{(k\times 1)}{e_2} \underset{(1\times k)}{e_2^T} + \ldots + \lambda_k \underset{(k\times 1)}{e_k} \underset{(1\times k)}{e_k^T} \implies A = \sum_{i=1}^{k} \lambda_i \underset{(k\times 1)}{e_i} \underset{(1\times k)}{e_i^T} = P\Lambda P^T$$

$$\underset{k\times k}{P} = [e_1, e_2, \ldots, e_k]; \quad \underset{k\times k}{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \lambda_k \end{bmatrix}$$

  - This is also called the eigen decomposition theorem.
- Any symmetric matrix can be reconstructed using its eigenvalues and eigenvectors.

Example of spectral decomposition

- Let A be a symmetric, positive definite matrix

$$A = \begin{bmatrix} 2.2 & 0.4 \\ 0.4 & 2.8 \end{bmatrix} \implies det(A - \lambda I) = 0$$

$$\implies \lambda^2 - 5\lambda + (6.16 - 0.16) = (\lambda - 3)(\lambda - 2) = 0$$

- The eigenvectors for the corresponding eigenvalues are $e_1^T = [\frac{1}{\sqrt{5}}, \frac{2}{\sqrt{5}}], e_2^T = [\frac{2}{\sqrt{5}}, \frac{-1}{\sqrt{5}}]$.
- Consequently,

$$A = \begin{bmatrix} 2.2 & 0.4 \\ 0.4 & 2.8 \end{bmatrix} = 3 \begin{bmatrix} \frac{1}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \end{bmatrix} + 2 \begin{bmatrix} \frac{2}{\sqrt{5}} \\ \frac{-1}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} \frac{2}{\sqrt{5}} & \frac{-1}{\sqrt{5}} \end{bmatrix}$$

$$= \begin{bmatrix} 0.6 & 1.2 \\ 1.2 & 2.4 \end{bmatrix} + \begin{bmatrix} 1.6 & -0.8 \\ -0.8 & 0.4 \end{bmatrix}$$

Eigendecomposition is not unique

- Eigendecomposition is $A = P\Lambda P^T$
  - where P is an orthogonal matrix composed of eigenvectors of A
- Decomposition is not unique when two eigenvalues are the same.
- Eigenvalues really work for square matrices. Consider a regression problem, we have input and output matrix and m and k have different dimensions. Then, this method fails. Solution: SVD.
- By convention order entries of Λ in descending order:
  - Under this convention, eigendecomposition is unique if all eigenvalues are unique

Singular Value Decomposition

- If A is rectangular $m \times k$ matrix of real numbers, then there exists an $m \times m$ orthogonal matrix U and a $k \times k$ orthogonal matrix V such that

$$\underset{(m\times k)}{A} = \underset{(m\times m)}{U}\,\underset{(m\times k)}{\Lambda}\,\underset{(k\times k)}{V^T} \qquad UU^T = VV^T = I$$

  - Λ is an $m \times k$ matrix where the $(i,j)^{th}$ entry $\lambda_i, i = j = 1, 2, \ldots, min(m,k)$ and the other entries are zero. Physically, A equals rotation$\times$stretching$\times$rotation.
    - The positive constants λ_i are the singular values of A.
- If A has rank r, then there exists r positive constants $\lambda_1, \lambda_2, \ldots, \lambda_r$; r orthogonal $m \times 1$ unit vectors $u_1, u_2, \ldots, u_r$ and r orthogonal $k \times 1$ unit vectors $v_1, v_2, \ldots, v_r$ such that

$$A = \sum_{i=1}^{r} \lambda_i u_i v_i^T$$

Singular Value Decomposition (contd.)

- If A is symmetric and positive definite then
  - SVD = Eigen decomposition
    - EIG(λ_i) = SVD(λ_i^2)
- Here AA^T has an eigenvalue-eigenvector pair (λ_i^2, u_i)

$$\begin{aligned} AA^T &= (U\Lambda V^T)(U\Lambda V^T)^T \\ &= U\Lambda V^T V\Lambda U^T \\ &= U\Lambda^2 U^T \end{aligned}$$

- Alternatively, the v_i are the eigenvectors of A^TA with same non zero eigenvalue λ_i^2

$$A^TA = V\Lambda^2 V^T$$

Example for SVD

- Let A be a symmetric, positive definite matrix
  - U can be computed as

$$A = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} \implies AA^T = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 11 & 1 \\ 1 & 11 \end{bmatrix}$$

$$det(AA^T - \gamma I) = 0 \implies \gamma_1 = 12, \gamma_2 = 10 \implies u_1^T = \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right], u_2^T = \left[\frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}\right]$$

  - V can be computed as

$$A = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} \implies A^TA = \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} = \begin{bmatrix} 10 & 0 & 2 \\ 0 & 10 & 4 \\ 2 & 4 & 2 \end{bmatrix}$$

$$det(A^TA - \gamma I) = 0 \implies \gamma_1 = 12, \gamma_2 = 10, \gamma_3 = 0$$

$$\implies v_1^T = \left[\frac{1}{\sqrt{6}}, \frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}}\right], v_2^T = \left[\frac{2}{\sqrt{5}}, \frac{-1}{\sqrt{5}}, 0\right], v_3^T = \left[\frac{1}{\sqrt{30}}, \frac{2}{\sqrt{30}}, \frac{-5}{\sqrt{30}}\right]$$

Example for SVD

- Taking $\lambda_1^2 = 12$ and $\lambda_2^2 = 10$, the singular value decomposition of A is

$$A = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix}$$
$$= \sqrt{12} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{6}}, \frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}} \end{bmatrix} + \sqrt{10} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{2}{\sqrt{5}}, \frac{-1}{\sqrt{5}}, 0 \end{bmatrix}$$

- Thus, the U, V and Λ are computed by performing eigendecomposition of AA^T and A^TA.
- Any matrix has a singular value decomposition but only symmetric, positive definite matrices have an eigen decomposition.

Use of SVD?

- SVD can be used to compute optimal low-rank approximations of arbitrary matrices.
- Face recognition
  - Represent the face images as eigenfaces and compute distance between the query face image in the principal component space.
- Data mining
  - Latent Semantic Indexing for document extraction.
- Image Compression

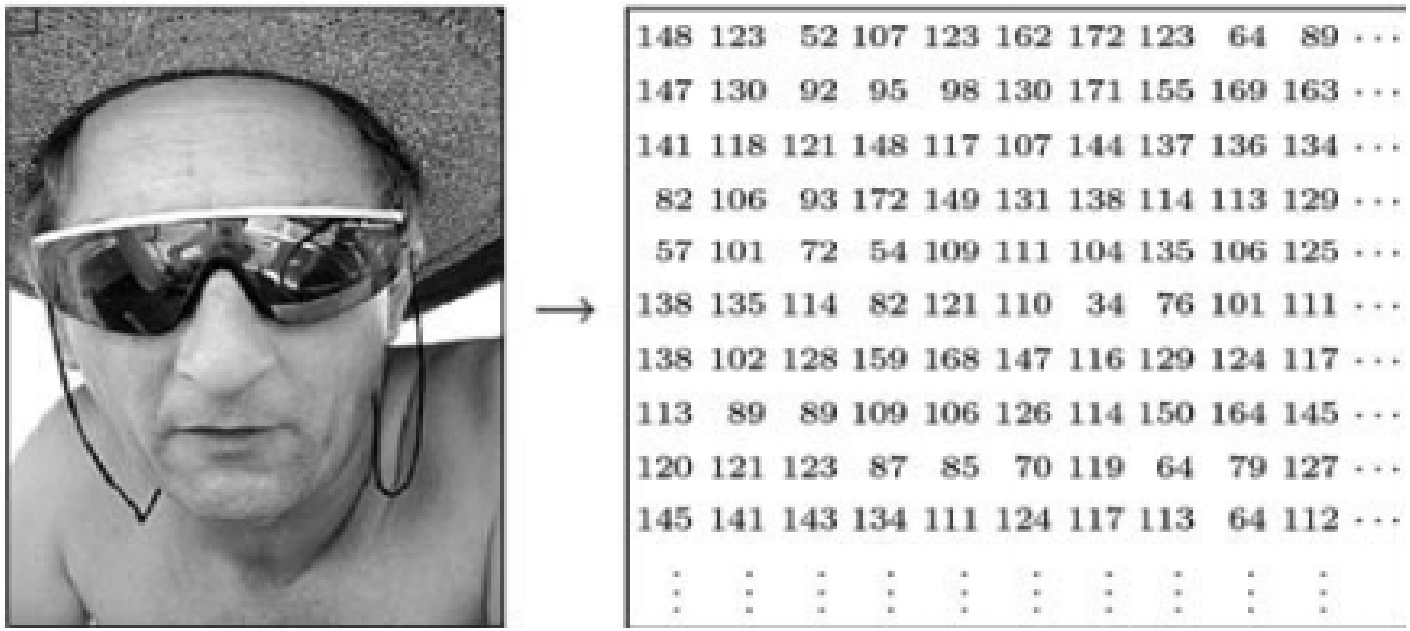
Linear Algebra to Image Processing

Introduction

- Image processing can be defined as the processing of images using mathematical operations.
- Digital images are obtained by the process of digitalization or directly using any digital device. The use of computers to perform image processing on digital images is called digital image processing.
- Digital Image processing is not just limited to retouching or resizing images captured by the camera; it is widely used nowadays. Some major fields are medicine, remote sensing, data transmission and encoding, robotics, computer vision, pattern recognition, the film industry, microscope imaging, and image sharpening and restoration.

Digital Images

A digital image is a representation of a real image as a set of numbers that can be stored and handled by digital computers.



The $(i,j)^{th}$ entry of the matrix comprises of $(i,j)^{th}$ pixel value, which determines the intensity of light of the image.

Image Attributes

- **Image Size:** Number of rows * Number of columns.

- **Image Resolution:** Area covered by per pixel.

- **Number of Color Planes:** Grayscale image(1), RGB image(3).

Image Attributes

- **Bit Depth:** Color information stored in each pixel of the image.

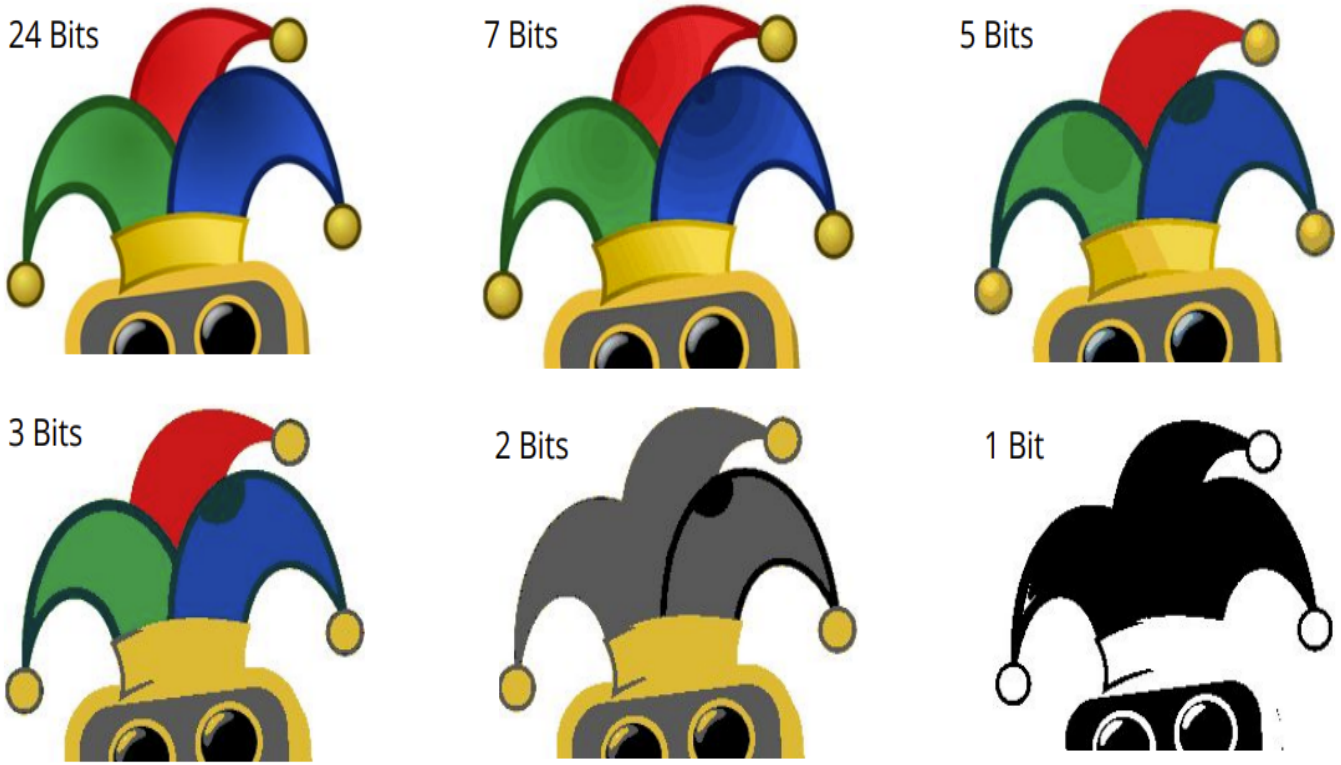


Image Attributes

- Pixels in an image carry the weight of color at that point. Let's consider a 8-bit gray-scale image i.e., each pixel can take any of the 2^8. Black is represented by the minimum value 0 and white by the maximum value $2^8 - 1 = 255$. All intermediate values represent different shades of grey.
- Matrix representation of color images depends on the color system used, RGB (the most popular one), specifies the amount of Red (R), Green (G) and Blue (B), and each color can vary from 0 to 255 for a 8-bit image. Thus, in the RGB, a pixel can be represented as a tri-dimensional vector (r, g, b) where r, g and b are integer numbers from 0 to 255. Each color matrix is called a 'channel'. Different combination of RGB produce different colors.

Image filters through matrix operations

- **Addition:** Similar to matrix addition and subtraction, two images of same size can be added or subtracted. The resulting added image will have pixel values as sum of its components, capped to 255 for a 8-bit image.

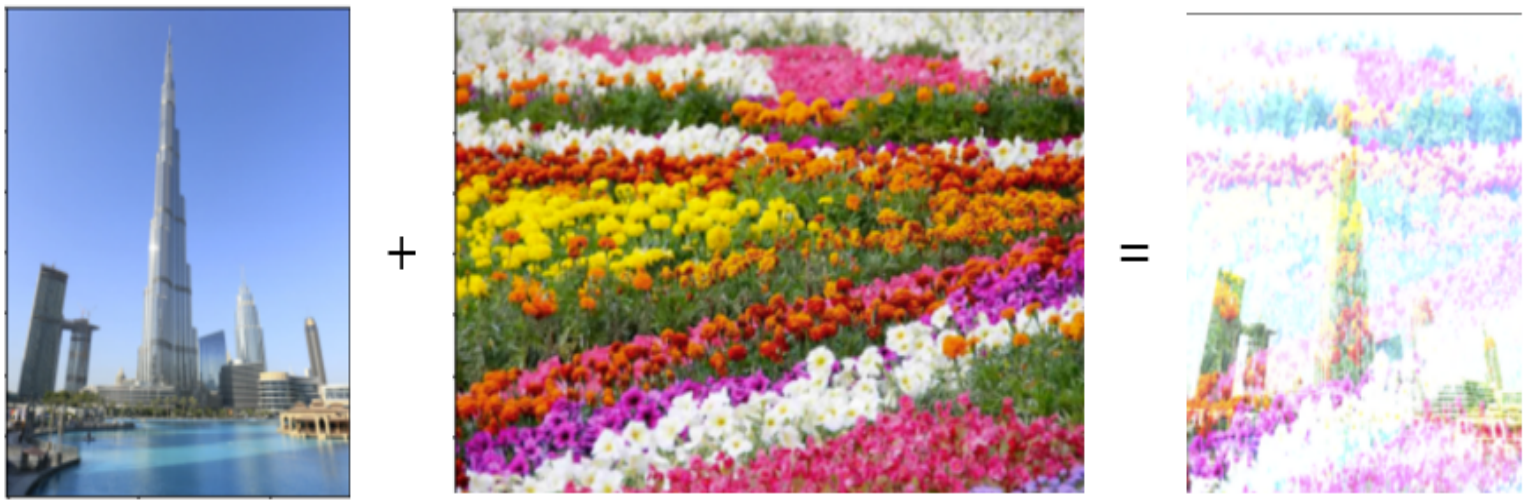


Image filters through matrix operations

- **Subtraction:** Subtracting two images is similar to adding the negative of the second image (i.e., flower image) to the first image (i.e., Burj Khalifa).

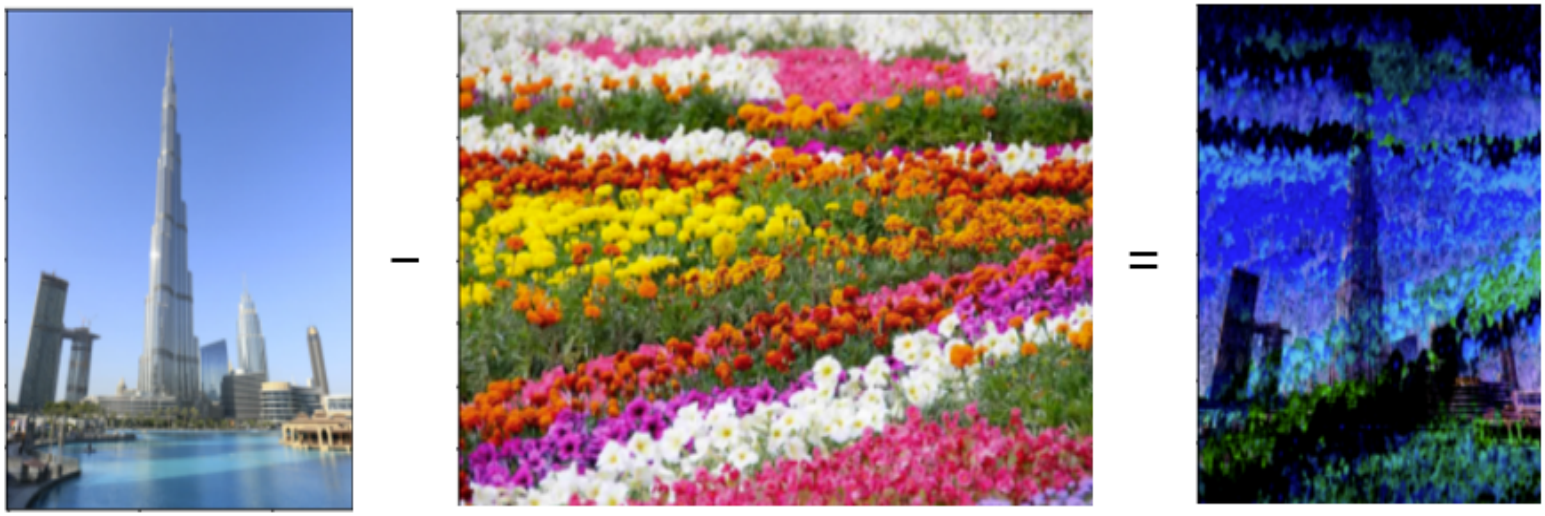


Image filters through matrix operations

- **Matrix Multiplication:** To create the mirror reflection of an image we multiply the image with a left diagonal matrix with entries 1 as follows:

$$\begin{bmatrix} 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & \dots & 1 & 0 \\ . & . & \dots & . & . \\ . & . & \dots & . & . \\ . & . & \dots & . & . \\ 1 & 0 & \dots & 0 & 0 \end{bmatrix}$$

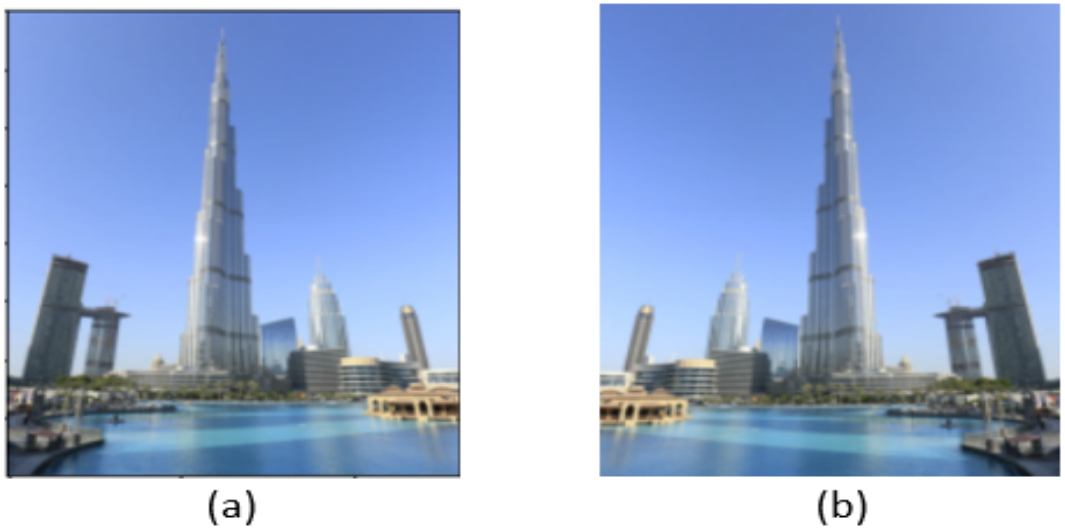


Figure: (a) Original image, (b) Mirror reflection

Image filters through matrix operations

- **Linear Transformation:** RGB images comprises of three color channels, hence can be represented as tri-dimensional vector) where each element represents a color channel of the image. In the simplest case the function multiplied can be a linear transformation, that transforms a tri-dimensional vector (pixel) into another tri-dimensional vector. When it's a linear transformation, the transformation can be represented as a 3x3 matrix T.

Image filters through matrix operations

- **Grey Scale Conversion:** The RGB image is transformed to its grey scale by applying an 3×3 average filter reprsented by:

$$T = \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

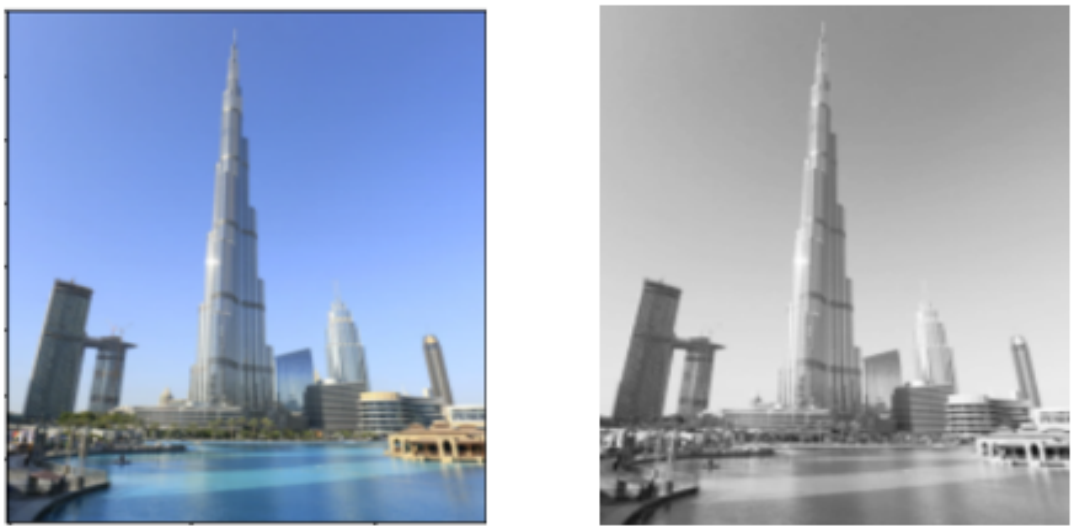


Figure: (Left) Original image, (Right) Grey scale conversion

Image filters through matrix operations

- **Sepia Effect:** The sepia effect is a popular tonal editing method that imparts a mellow tone to an image, giving it an archival or vintage appearance. It may be applied to any image by using the filter as

$$T = \begin{bmatrix} 0.393 & 0.769 & 0.189 \\ 0.349 & 0.686 & 0.168 \\ 0.272 & 0.534 & 0.131 \end{bmatrix}$$

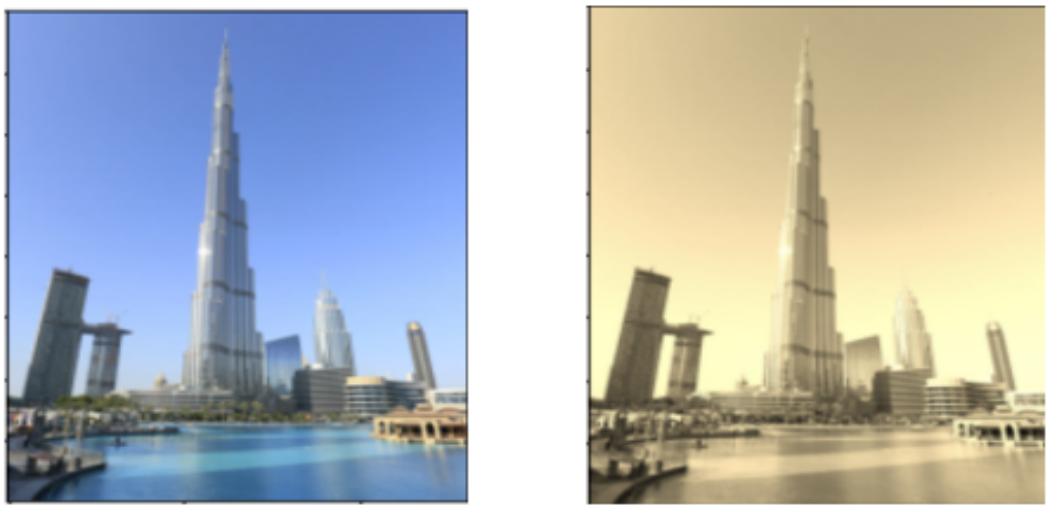


Figure: (Left) Original image, (Right) Sepia tone applied

Image filters through matrix operations

- **Channel Extraction:** The color channels of any image can be extracted separately by appropriate transformation. In case of the RGB image, T= [1 0 0], generates Red channel of the image, T= [0 1 0] provides Green channel, and T= [0 0 1] extracts Blue channel.

Figure: (Left) Red channel, (Middle) Green channel, (Right) Blue channel

Edge detection using gradients

- Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting discontinuities in brightness.
- Edge detection is used for image segmentation and data extraction in areas such as image processing, computer vision, face recognition, medical imaging, and fingerprint identification.

Edge detection using gradients

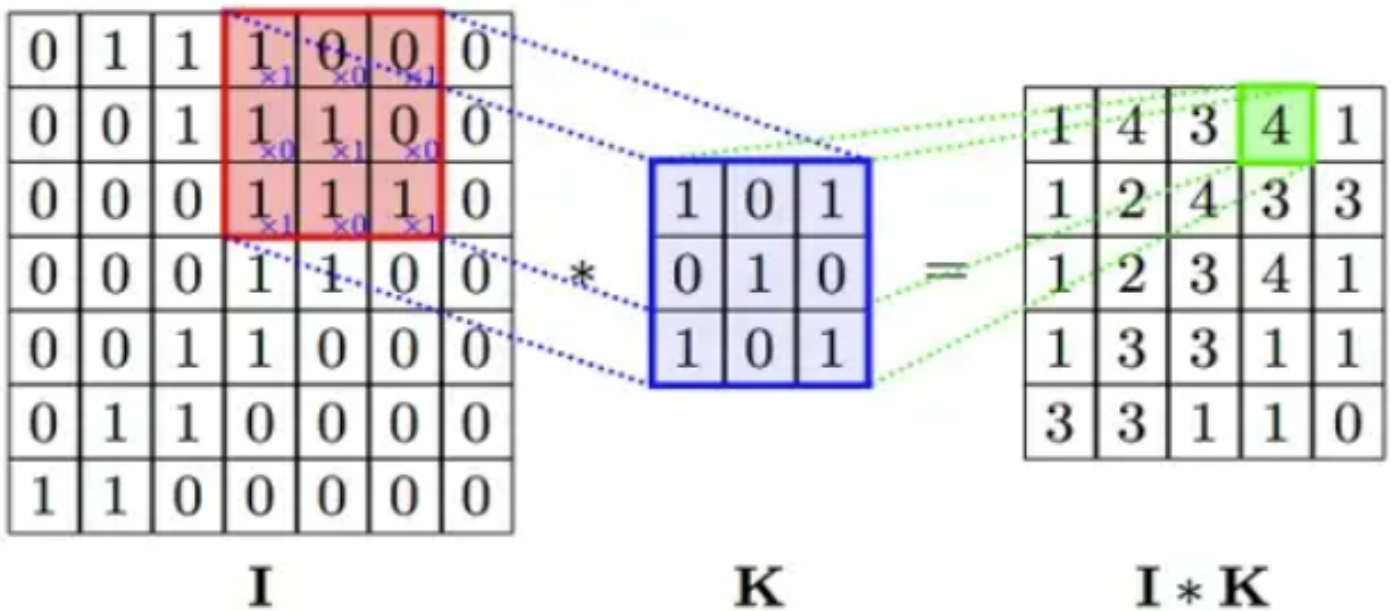
- Edges are significant local changes of intensity in a digital image. An edge can be defined as a set of connected pixels that forms a boundary between two disjoint regions. There are three types of edges:
  - Horizontal edges
  - Vertical edges
  - Diagonal edges
- Edges of an image represents region with a sharp contrast i.e., rapid change of intensity, hence the edges can be identified by computing gradients of the image.

Convolution

- Convolution is a simple mathematical operation which provides a way of 'multiplying together' two arrays of numbers, generally of different sizes, but of the same dimensionality, to produce a third array of numbers of the same dimensionality.
- Convolution is an essential operator in processing digital images and it allows to add elements of an image to its local neighbors, weighted by the kernel.
- Although represented by $*$, it differs from traditional matrix multiplication.
- For detecting edges of an image, the original image matrix is convoluted with relevant kernel matrix.

Convolution example

For two matrices I and K, where I represents an image and K represents a kernel, convolution is the process of multiplying locally similar (corresponding) entries and summing. So, the operation that we perform can also be called: Cross Correlation. The element at coordinates [1, 4] of the resulting image would be a weighted combination of all the entries of the image matrix, with weights given by the kernel. This process is visualized in the figure below.



Edge detection using gradients

Edge detection operators are of two types:

- **Gaussian** – based operator which computes second-order derivations in a digital image like, Canny edge detector
- **Gradient** – based operator which computes first-order derivations in a digital image like, Sobel operator, Prewitt operator

Canny edge detector

Canny operator is based on the advanced algorithm derived from of Laplacian of Gaussian operator. It is widely used as an optimal edge detection technique. It detects edges based on three criteria:

- Low error rate.
- Edge points must be accurately localized.
- There should be just one single edge response.

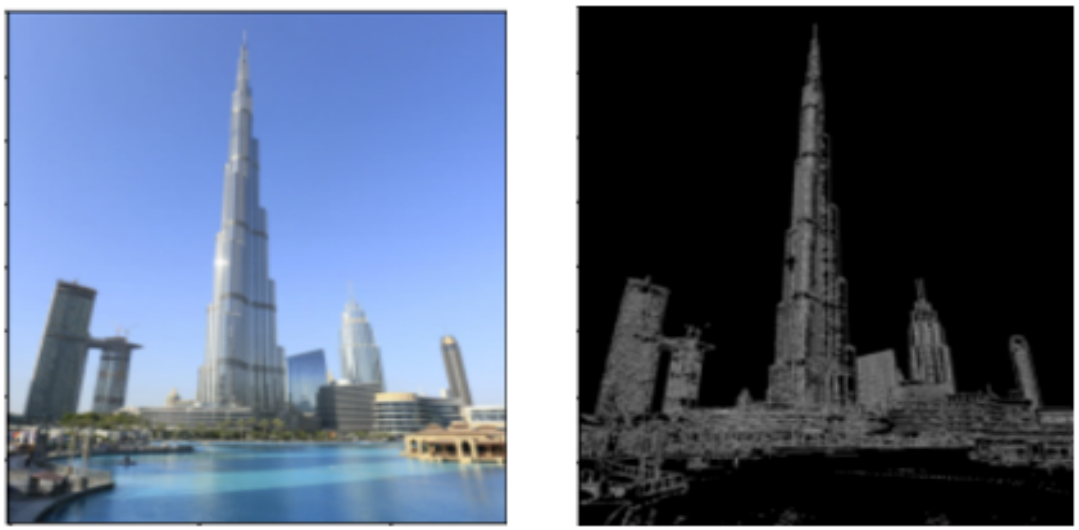


Figure: (Left) Original image (Right) Edges detected using Canny algorithm

Canny edge detector

- Advantages:
  - It has good localization
  - It extract image features without altering the features
  - Less Sensitive to noise
- Limitations:
  - There is false zero crossing
  - Complex computation and time consuming

Sobel edge detector

Sobel operator is a discrete differentiation operator. It computes the gradient approximation of image intensity function for image edge detection. At the pixels of an image, the Sobel operator produces either the normal to a vector or the corresponding gradient vector. It uses two 3×3 kernels or masks which are convolved with the input image to calculate the vertical and horizontal derivative approximations respectively –

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

These kernels are designed to respond maximally to edges running vertically and horizontally relative to the pixel grid, one kernel for each of the two perpendicular orientations. The kernels can be applied separately to the input image, to produce separate measurements of the gradient component in each orientation (call these Gx and Gy). These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient. The gradient magnitude is given by: $|G| = \sqrt{G_x^2 + G_y^2}$ Typically, an approximate magnitude is computed using: $|G| = |G_x| + |G_y|$

Sobel edge detector

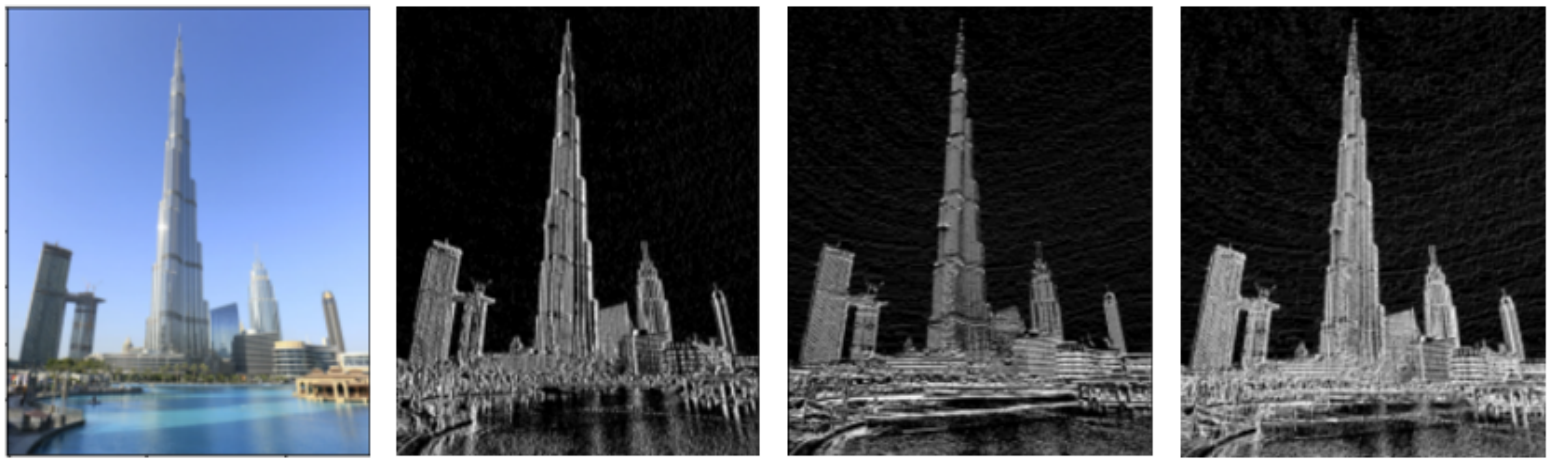


Figure: (Left to right) Original image, edges detected using Sobel vertical operator, horizontal operator, and combined operator

Sobel edge detector

- Advantages:
  - Simple and time efficient computation
  - Very easy at searching for smooth edges
- Limitations:
  - Diagonal direction points are not preserved always
  - Highly sensitive to noise
  - Not very accurate in edge detection
  - Detect with thick and rough edges does not give appropriate results

Prewitt edge detector

Prewitt operator is almost similar to the Sobel operator. It also detects vertical and horizontal edges of an image. It is one of the best ways to detect the orientation and magnitude of an image. It uses the kernels or masks –

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad M_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

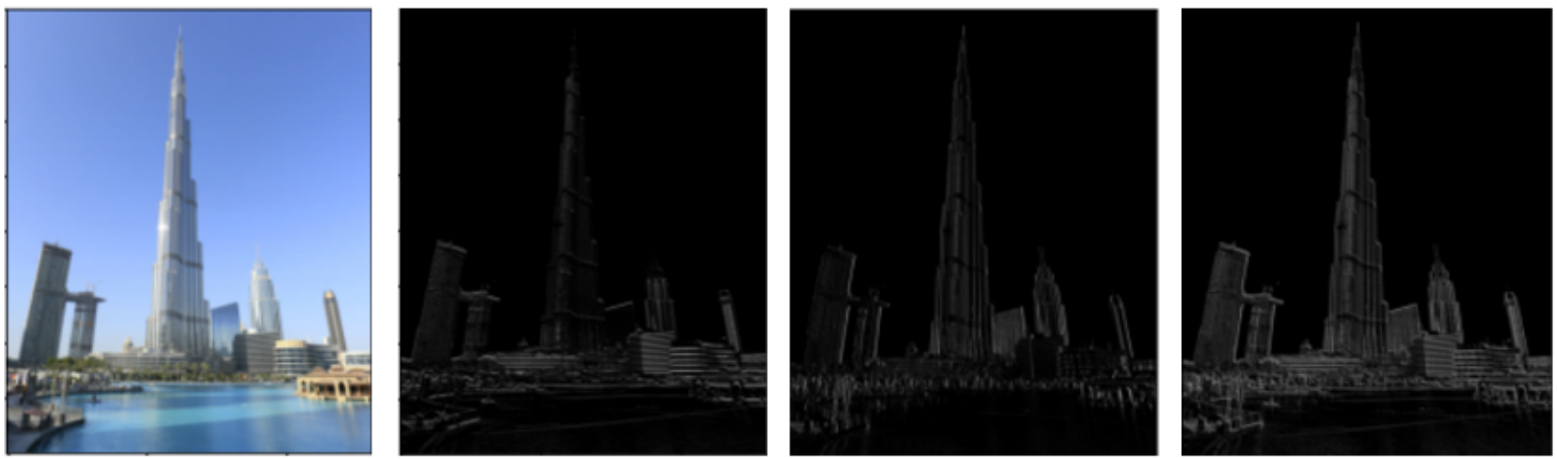


Figure: (Left to right) Original image, edges detected using Prewitt vertical operator, horizontal operator, and combined operator

Prewitt edge detector

- Advantages:
  - Good performance on detecting vertical and horizontal edges
  - Best operator to detect the orientation of an image
  - The magnitude of coefficient is fixed and cannot be changed
  - Diagonal direction points are not preserved always
- Limitations:
  - Diagonal direction points are not preserved always
  - Highly sensitive to noise
  - Not very accurate in edge detection
  - Detect with thick and rough edges does not give appropriate results

Image Compression using SVD

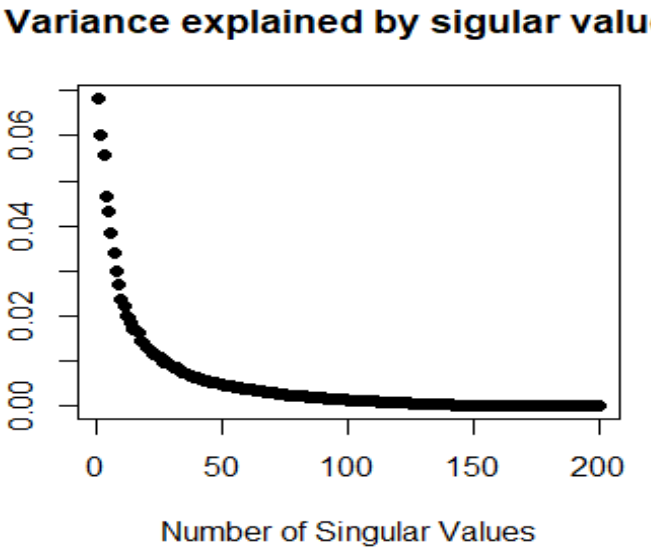
- An image is stored as a 200×200 matrix M with entries between 0 and 1. The matrix M has rank 200.
- Select $r > 200$ as an approximation to the original M.
  - As r is increased from 1 all the way to 200 the reconstruction of M would improve i.e. approximation error would reduce
- Advantage
  - To send the matrix M, need to send $200 \times 200 = 40000$ numbers.
  - To send an $r = 35$ approximation of M, need to send $35 + 35 * 200 + 35 * 200 = 14035$ numbers
    - 35 singular values.
    - 35 left vectors, each having 200 entries.
    - 35 right vectors, each having 200 entries.

Compression in digital images

Data and Code is available at https://github.com/mad-stat/SVD-Applications

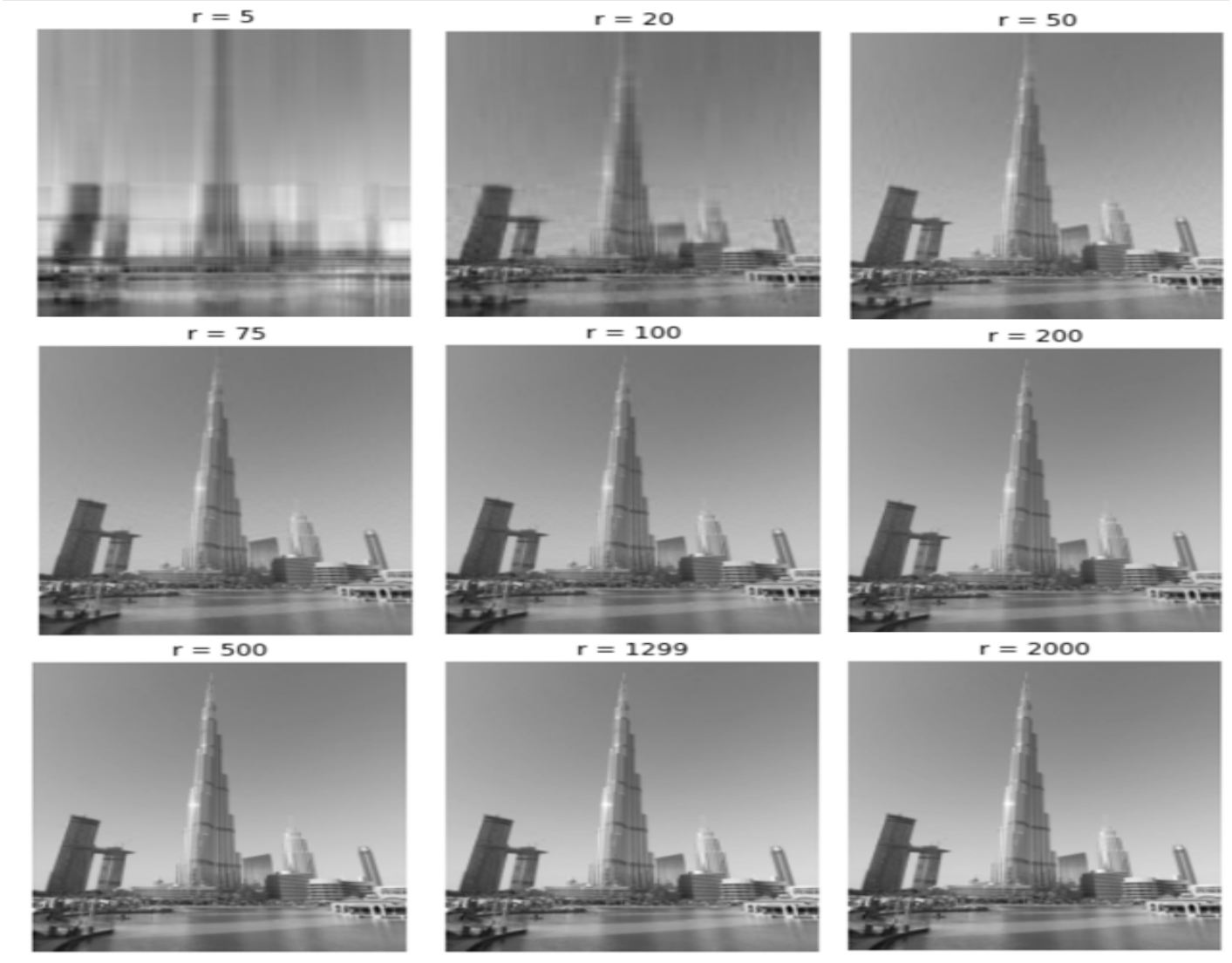
SVD of digital image

- Obtain the singular values (Λ) and the singular vectors (U, V) of the image using SVD.
- The amount of variance explained by the singular values can be obtained by plotting $\Lambda^2 / sum(\Lambda^2)$.



- Most of the variance is explained by first 35 singular vectors in this image.
- After 35 vectors the variance is very low and almost steady. After about 75 vectors it's miniscule. Hence, we can compress the image without losing much of the quality by retaining the first 75 vectors and discarding others.

Compressed image



Textbook and References