📚 Exam Cheat Sheet: Mathematical Concepts for Machine Learning

Instructor: Reza Moslemi, Ph.D., P.Eng.

📌 1. Linear Algebra & Vector Space Analysis

✅ Scalars, Vectors, Matrices, Tensors

- **Scalar**: Single numerical value. (e.g., $5, -3, \pi$)
- **Vector**: 1D array ($n \times 1$) (e.g., $\begin{bmatrix} 2 \\ -8.3 \\ 0 \end{bmatrix}$)
- **Matrix**: 2D array ($n \times m$)
- **Tensor**: 3D+ generalization of matrices.

✅ Vector Operations

- **Addition**: $A + B = [a_1 + b_1, a_2 + b_2, \ldots]$
- **Dot Product**: $A \cdot B = |A||B| \cos\theta$
- **Cross Product**: Results in a perpendicular vector.

✅ Matrix Operations

- **Multiplication**: $AB \neq BA$ in general.
- **Identity Matrix** I_n
- **Inverse Matrix** A^{-1}, only for square matrices.
- **Determinant** (important for invertibility).

✅ Covariance & Correlation

- **Covariance**: Measures how two variables change together.
- **Correlation**: Normalized covariance, always between -1 and 1.
  - $\rho_{X,Y} = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y}$

📌 2. NLP Mathematics

✅ Bag of Words (BoW)

- Represent text as a vector. Example:
  - "College Student" → (1,1,0)
  - "College Professor" → (1,0,1)

✅ TF-IDF (Term Frequency - Inverse Document Frequency)

- **TF**: $\frac{\text{Word Count in Document}}{\text{Total Words in Document}}$
- **IDF**: $\log \frac{N}{df}$ (inverse document frequency)
- **TF-IDF Score**: $\text{TF} \times \text{IDF}$

📌 3. Regression Algorithms

✅ Linear Regression

- **Equation**: $Y = mX + b$
- **Least Squares Loss Function**:
  - $\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$
- R^2 **Score**: $1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2}$

✅ Logistic Regression (for Classification)

- **Sigmoid Function**: $p = \frac{1}{1+e^{-z}}$
- **Confusion Matrix**: Evaluates model performance.

📌 4. Decision Trees

✅ Tree Elements

- **Root Node**: First split.
- **Leaf Nodes**: Final predictions.
- **Entropy**: $H(S) = -\sum p_i \log_2 p_i$

✅ Random Forest

- Multiple trees → Reduces overfitting.

📌 5. Gradient Descent

✅ **Gradient**: Direction of steepest increase.

- **Gradient Descent**: Move in the **negative** gradient direction to minimize loss.
- **Learning Rate** α: Controls step size.

✅ Types of Gradient Descent

1. **Batch Gradient Descent**: Uses all data points.
2. **Stochastic Gradient Descent (SGD)**: Uses one sample at a time.
3. **Mini-Batch Gradient Descent**: Uses small batches.

📌 6. Support Vector Machines (SVMs)

✅ **Support Vectors**: Data points closest to the decision boundary.
✅ **Kernel Trick**: Projects data into higher dimensions for separability.

📌 7. Clustering Algorithms

✅ K-Means Clustering

- Groups data into k clusters.
- **Objective**: Minimize intra-cluster variance.

✅ Types of Clustering

- **Centroid-based** (K-Means)
- **Density-based** (DBSCAN)
- **Distribution-based**

📌 1. Linear Algebra & Vector Space Analysis

🔷 Explanation

Linear algebra is essential in machine learning for handling data in vector and matrix forms. Here are the key concepts:

Scalars, Vectors, Matrices, and Tensors

- **Scalar**: A single numerical value. Example: $5, -3, \pi$.
- **Vector**: A one-dimensional array. Example:

$$v = \begin{bmatrix} 2 \\ -8.3 \\ 0 \end{bmatrix}$$

- **Matrix**: A two-dimensional array. Example:

$$M = \begin{bmatrix} 4 & 6 & 75 \\ -8 & 5 & 6 \\ 0 & 0 & 42 \end{bmatrix}$$

- **Tensor**: A generalization of vectors and matrices into multiple dimensions (3D or higher).

Vector Operations

- **Addition**: Element-wise sum.

$$\begin{bmatrix} 8 \\ 5 \\ 4 \end{bmatrix} + \begin{bmatrix} 0 \\ -2 \\ 14 \end{bmatrix} = \begin{bmatrix} 8 \\ 3 \\ 18 \end{bmatrix}$$

- **Scalar Multiplication**: Multiply each element by a scalar.

$$3 \times \begin{bmatrix} 8 \\ 5 \\ 4 \end{bmatrix} = \begin{bmatrix} 24 \\ 15 \\ 12 \end{bmatrix}$$

- **Dot Product**:

$$A \cdot B = |A||B| \cos\theta$$

Example:

$$\begin{bmatrix} 8 \\ 5 \\ 4 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ -2 \\ 14 \end{bmatrix} = (8 \times 0) + (5 \times -2) + (4 \times 14) = 46$$

- **Cross Product**: Results in a perpendicu ↓ vector.

Matrix Operations

- **Matrix Multiplication**:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{bmatrix}$$

- **Identity Matrix**: I_n has ones on the diagonal, zeros elsewhere.
- **Inverse of a Matrix** (for square matrices):

$$A^{-1}A = I$$

- **Determinant**: Helps in checking if a matrix is invertible.

Covariance & Correlation

- **Covariance**: Measures how two variables change together.
- **Correlation**: Normalized covariance, always between -1 and 1.

$$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y}$$

  - $\rho = 1$ → Perfect correlation
  - $\rho = 0$ → No correlation
  - $\rho = -1$ → Perfect inverse correlation

📜 Cheat Sheet (Linear Algebra & Vector Space)

✅ **Scalars, Vectors, Matrices, and Tensors**
✅ **Vector Operations** (Addition, Dot Product, Cross Product)
✅ **Matrix Operations** (Multiplication, Inverse, Determinant)
✅ **Covariance & Correlation**

📝 Possible Exam Questions & Solutions

1. Compute the dot product of $A = [3, 4]$ and $B = [5, 2]$.
Solution:

$$(3 \times 5) + (4 \times 2) = 15 + 8 = 23$$

2. Given $A = \begin{bmatrix} 3 & 4 \\ 2 & 1 \end{bmatrix}$, find A^{-1}.

Solution:

$$A^{-1} = \frac{1}{\det(A)} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

$$\det(A) = (3 \times 1) - (4 \times 2) = 3 - 8 = -5$$

$$A^{-1} = \frac{1}{-5} \begin{bmatrix} 1 & -4 \\ -2 & 3 \end{bmatrix} = \begin{bmatrix} -1/5 & 4/5 \\ 2/5 & -3/5 \end{bmatrix}$$

📌 2. Regression Algorithms

🔷 Explanation

Regression algorithms are supervised learning models that predict numerical values.

Linear Regression

$$Y = mX + b$$

- **Loss Function** (Least Squares):

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- R^2 **Score**:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

Logistic Regression

- Used for **classification**.
- Uses the **sigmoid function**:

$$p = \frac{1}{1 + e^{-z}}$$

- **Confusion Matrix:** Measures model performance.

📜 Cheat Sheet (Regression Algorithms)

✅ **Linear Regression Equation**
✅ **Loss Function**
✅ R^2 **Score**
✅ **Logistic Regression & Sigmoid Function**
✅ **Confusion Matrix**

📝 Possible Exam Questions & Solutions

1. Given $Y = 3X + 2$, predict Y when $X = 5$.
Solution:

$$Y = (3 \times 5) + 2 = 15 + 2 = 17$$

2. Compute R^2 given: Actual $y = [2, 4, 5, 6]$, Predicted $\hat{y} = [2.5, 3.8, 4.9, 6.2]$.
Solution:

$$\sum(y_i - \hat{y}_i)^2 = (2 - 2.5)^2 + (4 - 3.8)^2 + (5 - 4.9)^2 + (6 - 6.2)^2$$

$$= 0.25 + 0.04 + 0.01 + 0.04 = 0.34$$

$$\sum(y_i - \bar{y})^2 = (2 - 4.25)^2 + (4 - 4.25)^2 + (5 - 4.25)^2 + (6 - 4.25)^2$$

$$= 5.06 + 0.06 + 0.56 + 3.06 = 8.74$$

$$R^2 = 1 - \frac{0.34}{8.74} = 1 - 0.039 = 0.961$$

I'll continue with **Decision Trees, Gradient Descent, SVMs, and Clustering** in the next response. Let me know if you want me to modify anything so far! 🚀

📌 3. Decision Trees

🔷 Explanation

A **Decision Tree (DT)** is a supervised learning model that predicts an outcome based on decision rules inferred from data.

Tree Components

- **Root Node**: The starting point of the tree.
- **Decision Nodes**: Intermediate nodes where data is split.
- **Leaves**: Terminal nodes that predict outcomes.
- **Edges**: Connect nodes and represent decisions.

How Decision Trees Work

- At each step, the model finds the attribute that minimizes **entropy** (uncertainty) and maximizes **information gain**.
- The **best split** is the one that results in the most **pure** subgroups.

Entropy (H)

- Measures disorder in the data:

$$H(S) = -\sum p_i \log_2 p_i$$

- Lower entropy means **better classification**.

Overfitting & Improvement

- Decision trees can overfit, meaning they memorize training data rather than generalizing.
- Solutions:
  - **Pruning**: Remove unnecessary branches.
  - **Random Forest**: Train multiple decision trees and take a majority vote.

📜 Cheat Sheet (Decision Trees)

✅ **Root, Nodes, Leaves, Edges**
✅ **Entropy & Information Gain**
✅ **Overfitting Solutions (Pruning, Random Forests)**

📝 Possible Exam Questions & Solutions

1. Given a dataset with 80% "Yes" and 20% "No", compute the entropy.
Solution:

$$H = -(0.8 \log_2 0.8 + 0.2 \log_2 0.2)$$

$$= -(0.8 \times -0.32 + 0.2 \times -2.32) = 0.72$$

2. How can you reduce overfitting in a decision tree?
✅ **Pruning** (removing weak branches).
✅ **Random Forests** (using multiple trees).

📌 4. Gradient Descent

🔹 Explanation

Gradient Descent is an **optimization algorithm** used to minimize the **loss function**.

Key Concepts

- **Gradient**: Direction and rate of **fastest increase**.
- **Learning Rate (α)**: Determines the step size.
- **Local vs. Global Minima**: Small learning rates avoid getting stuck in local minima.

Types of Gradient Descent

1. **Batch Gradient Descent**: Uses all data points per update.
2. **Stochastic Gradient Descent (SGD)**: Updates using one data point at a time (faster, more noise).
3. **Mini-batch Gradient Descent**: Uses small groups of data points (a balance between batch and SGD).

📜 Cheat Sheet (Gradient Descent)

✅ **Minimizes Loss Function**
✅ **Learning Rate (α)**
✅ **Batch, SGD, Mini-batch Methods**

📝 Possible Exam Questions & Solutions

1. Why does a large learning rate (α) cause problems in gradient descent?
✅ It may **overshoot** the minimum and fail to converge.

2. What is the difference between Batch and Stochastic Gradient Descent?
✅ **Batch GD**: Uses all data points (slower, accurate).
✅ **SGD**: Uses one data point per step (faster, noisier).

📌 5. Support Vector Machines (SVMs)

🔹 Explanation

SVMs are **supervised learning models** that separate classes using a **hyperplane**.

Key Concepts

- **Support Vectors**: Data points closest to the decision boundary.
- **Hyperplane**: The boundary separating different classes.
- **Margin**: The distance between the hyperplane and closest points.

Kernel Trick

- **Linear SVM**: Works well if data is linearly separable.
- **Non-Linear SVM**: Uses a **kernel function** (e.g., RBF, polynomial) to map data into higher dimensions.

📜 Cheat Sheet (SVMs)

✅ **Support Vectors, Hyperplanes, Margin**
✅ **Kernel Trick for Non-Linear Data**

📜 Cheat Sheet (SVMs)

✅ **Support Vectors, Hyperplanes, Margin**
✅ **Kernel Trick for Non-Linear Data**

📝 Possible Exam Questions & Solutions

1. What is the purpose of the kernel trick in SVM?
✅ It transforms data into a **higher dimension** to make it separable.

2. What is the main objective of an SVM?
✅ To find a **maximum margin hyperplane** that best separates classes.

📌 6. Clustering Algorithms

🔷 Explanation

Clustering is an **unsupervised learning** method that groups similar data points together.

Types of Clustering

1. **K-Means Clustering**
   - Uses **K centroids** and assigns data to the nearest centroid.
   - Iteratively updates centroids until convergence.
2. **Density-Based Clustering (DBSCAN)**
   - Finds dense regions of points.
3. **Hierarchical Clustering**
   - Builds a tree-like structure of clusters.

K-Means Algorithm Steps

1. Choose **K** clusters.
2. Assign each data point to the **nearest centroid**.
3. Recalculate centroids.
4. Repeat until centroids **stop moving**.

📜 Cheat Sheet (Clustering)

✅ **K-Means, DBSCAN, Hierarchical Clustering**
✅ **Centroids & Cluster Assignments**

📝 Possible Exam Questions & Solutions

1. What happens if K is too large in K-Means?
✅ Overfitting: Too many small clusters.

2. Given 10 data points, apply 2-means clustering.
Solution:

1. Choose **K = 2** random centroids.
2. Assign each point to the closest centroid.
3. Recalculate centroids.
4. Repeat until convergence.

---

1. Matrix Factorization & Singular Value Decomposition (SVD)

Key Concepts:

- Matrix Factorization: Expressing a matrix as the product of multiple matrices.
- Singular Value Decomposition (SVD):

$$M = U\Sigma V^T$$

  - U: Orthogonal matrix (left singular vectors)
  - Σ: Diagonal matrix (singular values)
  - V^T: Orthogonal matrix (right singular vectors)
- Application in **recommender systems**:
  - A user-item rating matrix R can be factorized into:

$$R \approx P \cdot Q^T$$

  - P represents users, Q represents items.
  - Missing values in R can be estimated using an optimization approach.

Mathematical Application:

- Gradient Descent to minimize loss function:

$$\min_{P,Q} \sum (r_{ij} - p_i q_j^T)^2 + \lambda(||P||^2 + ||Q||^2)$$

  - λ: Regularization parameter.

---

2. Digital Signal Processing (DSP)

Key Concepts:

- **Sampling & Quantization**: Converting analog signals into digital signals.
- **Basic Operations**:
  - **Time Reversal**: $x(-n)$
  - **Time Shifting**: $x(n-k)$ shifts right if $k > 0$.
  - **Time Scaling**: Compression or expansion of signals.
  - **Decimation** (Downsampling) & **Interpolation** (Upsampling).

Mathematical Application:

- Given an analog function $x(t) = 0.5t^2 + \sin(t)$, compute sampled values $x(n)$.

---

3. Fourier Transforms (1D & 2D)

Key Concepts:

- **Fourier Series**: Representing periodic functions using sinusoids.
- **Fourier Transform**:
  - Continuous Fourier Transform:
  $$F(u) = \int_{-\infty}^{\infty} f(t)e^{-j2\pi ut}dt$$
  - Discrete Fourier Transform (DFT):
  $$F(u) = \frac{1}{M}\sum_{t=0}^{M-1} f(t)e^{-j\frac{2\pi ut}{M}}$$
  - **Magnitude Spectrum**: $|F(u)| = \sqrt{R^2 + I^2}$.
  - **Phase Spectrum**: $\phi(u) = \tan^{-1}(I/R)$.
- **Filtering in the frequency domain**:
  - **Low-pass filter**: Removes high-frequency noise.
  - **High-pass filter**: Emphasizes edges.

Mathematical Application:

- Compute the Fourier Transform of a box function (rectangular pulse).

4. Image Processing & Transformations

Key Concepts:

- **Convolution**:
  - Applying a **filter (kernel)** to an image to extract features.
  - Example: **Edge detection** using Sobel/Kirsch operators.
  - Formula:
  $$g(x, y) = \sum_{s,t} w(s, t) f(x + s, y + t)$$
- **Geometric Transformations**:
  - **Affine Transformations**: Preserve collinearity & ratios of distances.
  - **Rotation**:
  $$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$
  - **Translation**: $u = x + c_1, v = y + c_2$.
  - **Scaling**: $u = s_x x, v = s_y y$.

Mathematical Application:

- Apply an affine transformation (rotation + scaling) to an image.

Cheat Sheet

Topic	Formula/Concept
SVD	$M = U\Sigma V^T$
Gradient Descent	$P \leftarrow P - \eta\nabla P, \quad Q \leftarrow Q - \eta\nabla Q$
Sampling	$x(n) = x(nT)$
Fourier Transform	$F(u) = \int_{-\infty}^{\infty} f(t)e^{-j2\pi ut}dt$
Discrete Fourier Transform	$F(u) = \frac{1}{M}\sum f(t)e^{-j2\pi ut/M}$
Convolution	$g(x, y) = \sum w(s, t) f(x + s, y + t)$
Translation	$u = x + c, v = y + c$
Rotation	$u = x\cos\theta - y\sin\theta, v = x\sin\theta + y\cos\theta$
Scaling	$u = s_x x, v = s_y y$

1. Matrix Factorization & SVD

Q1: Compute the Singular Value Decomposition (SVD) of a Matrix

Given:

$$M = \begin{bmatrix} 4 & 0 \\ 3 & -5 \end{bmatrix}$$

Find $M = U\Sigma V^T$.

Solution:

Step 1: Compute $M^T M$

$$M^T M = \begin{bmatrix} 4 & 3 \\ 0 & -5 \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 3 & -5 \end{bmatrix} = \begin{bmatrix} 16+9 & 0+(-15) \\ 0+(-15) & 0+25 \end{bmatrix}$$

$$M^T M = \begin{bmatrix} 25 & -15 \\ -15 & 25 \end{bmatrix}$$

Step 2: Find Eigenvalues of $M^T M$
Solve $\det(M^T M - \lambda I) = 0$.

$$\begin{vmatrix} 25-\lambda & -15 \\ -15 & 25-\lambda \end{vmatrix} = 0$$

$$(25-\lambda)(25-\lambda) - (-15)(-15) = 0$$

$$\lambda^2 - 50\lambda + 625 - 225 = 0$$

$$\lambda^2 - 50\lambda + 400 = 0$$

$$(\lambda - 40)(\lambda - 10) = 0$$

$$\lambda_1 = 40, \quad \lambda_2 = 10$$

Step 3: Compute Singular Values

$$\sigma_1 = \sqrt{40} = 2\sqrt{10}, \quad \sigma_2 = \sqrt{10}$$

Thus,

$$\Sigma = \begin{bmatrix} 2\sqrt{10} & 0 \\ 0 & \sqrt{10} \end{bmatrix}$$

Step 4: Compute U and V
Eigenvectors of $M^T M$ give V, and compute $U = MV\Sigma^{-1}$ gives the final SVD.

2. Digital Signal Processing

Q2: Compute Sampled Values of a Function

Given $x(t) = 2t + \sin(t)$, compute sampled values at $t = 0, 1, 2, 3, 4$.

Solution:

$$x(0) = 2(0) + \sin(0) = 0$$

$$x(1) = 2(1) + \sin(1) \approx 2 + 0.8415 = 2.8415$$

$$x(2) = 2(2) + \sin(2) \approx 4 + 0.9093 = 4.9093$$

$$x(3) = 2(3) + \sin(3) \approx 6 + 0.1411 = 6.1411$$

$$x(4) = 2(4) + \sin(4) \approx 8 - 0.7568 = 7.2432$$

Thus, sampled values are:

$$\{0, 2.8415, 4.9093, 6.1411, 7.2432\}$$

---

3. Fourier Transform

Q3: Compute the Fourier Transform of $f(t) = e^{-at}$ for $t \geq 0$

Solution:

The Fourier Transform formula is:

$$F(u) = \int_0^\infty e^{-at} e^{-j2\pi ut} dt$$

$$F(u) = \int_0^\infty e^{-(a+j2\pi u)t} dt$$

Using the integral:

$$\int e^{-bt} dt = \frac{e^{-bt}}{-b}$$

$$F(u) = \frac{1}{a + j2\pi u}, \quad \text{for } a > 0$$

Thus, the Fourier Transform is:

$$F(u) = \frac{1}{a + j2\pi u}$$

---

4. Image Processing (Convolution)

Q4: Apply a 3×3 Edge Detection Filter to an Image

Given an image matrix:

$$I = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Apply the **Sobel edge detection filter**:

$$K = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Solution:

1. **Select center pixel (5) and its neighbors:**

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

2. **Multiply each pixel by filter values:**

$$(-1)(1) + (0)(2) + (1)(3) + (-2)(4) + (0)(5) + (2)(6) + (-1)(7) + (0)(8) + (1)(9)$$

$$= -1 + 0 + 3 - 8 + 0 + 12 - 7 + 0 + 9$$

$$= 8$$

Thus, the result at the center pixel is **8**.

📌 Complex Numbers & Cartesian System

🔷 Explanation

A **complex number** is written as:

$$z = a + bi$$

where:

- a = **Real part**
- b = **Imaginary part** (multiplied by i, where $i^2 = -1$)

Complex Plane (Argand Diagram)

A complex number is plotted on the **Cartesian coordinate system** where:

- The **x-axis** represents the **real part** (a).
- The **y-axis** represents the **imaginary part** (b).

Quadrants and Sign of Complex Numbers

Quadrant	Real Part (a)	Imaginary Part (b)
I (Top-Right)	$+$	$+$
II (Top-Left)	$-$	$+$
III (Bottom-Left)	$-$	$-$
IV (Bottom-Right)	$+$	$-$

Example: Identify the Signs

- $z_1 = 3 + 4i$ → **Quadrant I** → $(+, +)$
- $z_2 = -2 + 5i$ → **Quadrant II** → $(-, +)$
- $z_3 = -1 - 3i$ → **Quadrant III** → $(-, -)$
- $z_4 = 4 - 2i$ → **Quadrant IV** → $(+, -)$

🔷 Finding the Original Matrix R

Given P **and** Q, reconstruct R using **matrix multiplication**:

$$R = P \times Q$$

Example

Given:

$$P = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}, \quad Q = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Find R:

$$R = P \times Q = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} \times \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$= \begin{bmatrix} (2 \times 1 + 3 \times 3) & (2 \times 2 + 3 \times 4) \\ (4 \times 1 + 5 \times 3) & (4 \times 2 + 5 \times 4) \end{bmatrix}$$

$$= \begin{bmatrix} 2 + 9 & 4 + 12 \\ 4 + 15 & 8 + 20 \end{bmatrix} = \begin{bmatrix} 11 & 16 \\ 19 & 28 \end{bmatrix}$$

Thus, $R = \begin{bmatrix} 11 & 16 \\ 19 & 28 \end{bmatrix}$.

📜 Cheat Sheet (Recommendation Systems & Matrix Factorization)

✅ **Matrix Factorization**: $R \approx P \times Q$
✅ **Reconstruct Original Matrix**: Multiply P and Q
✅ **Each entry**: $R_{ij} = P_i \cdot Q_j$

TFIDF example

📌 TF-IDF Example Question & Solution

🔷 Question:

Given the following three documents:

- **Doc 1**: "Machine learning is fun"
- **Doc 2**: "Deep learning is a subset of machine learning"
- **Doc 3**: "Machine learning and deep learning are popular"

1. Construct the **TF matrix** for the words: **machine, learning, deep, fun**.
2. Compute the **IDF** values for each word.
3. Calculate the **TF-IDF** scores for each word in **Doc 1**.

🔷 Solution:

Step 1: Count Words in Each Document

We focus on **machine, learning, deep, fun** and count their occurrences in each document.

Word	Doc 1	Doc 2	Doc 3
machine	1	1	1
learning	1	2	1
deep	0	1	1
fun	1	0	0

Step 2: Compute Term Frequency (TF)

TF formula:

$$TF = \frac{\text{word count in document}}{\text{total words in document}}$$

Total words in each document:

- **Doc 1:** 4 words
- **Doc 2:** 7 words
- **Doc 3:** 6 words

Applying the formula:

Word	Doc 1 (4 words)	Doc 2 (7 words)	Doc 3 (6 words)
machine	$\frac{1}{4} = 0.25$	$\frac{1}{7} = 0.14$	$\frac{1}{6} = 0.17$
learning	$\frac{1}{4} = 0.25$	$\frac{2}{7} = 0.29$	$\frac{1}{6} = 0.17$
deep	$\frac{0}{4} = 0$	$\frac{1}{7} = 0.14$	$\frac{1}{6} = 0.17$
fun	$\frac{1}{4} = 0.25$	$\frac{0}{7} = 0$	$\frac{0}{6} = 0$

Step 3: Compute Inverse Document Frequency (IDF)

IDF formula:

$$IDF = \log \frac{N}{df}$$

where:

- $N = 3$ (total documents)
- df (document frequency) is the number of documents containing the word.

Word	df (Docs Containing Word)	IDF Calculation	IDF Value
machine	3	$\log \frac{3}{3} = \log 1$	0
learning	3	$\log \frac{3}{3} = \log 1$	0
deep	2	$\log \frac{3}{2} = 0.18$	0.18
fun	1	$\log \frac{3}{1} = 0.48$	0.48

Step 4: Compute TF-IDF for Doc 1

$$TF - IDF = TF \times IDF$$

Word	TF (Doc 1)	IDF	TF-IDF (Doc 1)
machine	0.25	0	$0.25 \times 0 = 0$
learning	0.25	0	$0.25 \times 0 = 0$
deep	0	0.18	$0 \times 0.18 = 0$
fun	0.25	0.48	$0.25 \times 0.48 = 0.12$

Thus, TF-IDF scores for Doc 1:

$$(\text{machine: } 0, \quad \text{learning: } 0, \quad \text{deep: } 0, \quad \text{fun: } 0.12)$$

📜 Cheat Sheet (TF-IDF Calculation)

✅ TF formula: $\frac{\text{word count in document}}{\text{total words in document}}$
✅ IDF formula: $\log \frac{N}{df}$
✅ TF-IDF formula: $TF \times IDF$
✅ Higher TF-IDF = More important word

📝 Another Possible Exam Question

Question:
Compute the **TF-IDF score** for the word **"learning"** in **Doc 2**, using the given document set.

Solution:

1. **TF for "learning" in Doc 2** = $\frac{2}{7} = 0.29$
2. **IDF for "learning"** = $\log \frac{3}{3} = 0$
3. **TF-IDF** = $0.29 \times 0 = 0$

✅ **Final Answer:** TF-IDF = **0** (since "learning" appears in all documents).

Entropy

🔷 Example Question

A dataset has the following class distribution:

- **Class A**: 4 occurrences
- **Class B**: 6 occurrences

Find the entropy.

🔷 Solution

Step 1: Compute Class Probabilities

- Total samples = $4 + 6 = 10$
- $p_A = \frac{4}{10} = 0.4$
- $p_B = \frac{6}{10} = 0.6$

Step 2: Apply Entropy Formula

$$H(S) = -(p_A \log_2 p_A + p_B \log_2 p_B)$$

$$H(S) = -(0.4 \log_2 0.4 + 0.6 \log_2 0.6)$$

Using log base 2 values:

- $\log_2 0.4 \approx -1.32$
- $\log_2 0.6 \approx -0.74$

$$H(S) = -(0.4 \times -1.32 + 0.6 \times -0.74)$$

$$H(S) = -(-0.528 - 0.444)$$

$$H(S) = 0.972$$

✅ **Final Answer: Entropy = 0.97** (approx.)

📝 Another Possible Exam Question

Question:

A dataset contains 3 classes:

- Class A: 3 samples
- Class B: 3 samples
- Class C: 4 samples

Find the entropy.

Solution:

1. **Total samples** = $3 + 3 + 4 = 10$
2. **Probabilities**:
   - $p_A = \frac{3}{10} = 0.3$
   - $p_B = \frac{3}{10} = 0.3$
   - $p_C = \frac{4}{10} = 0.4$
3. **Apply formula**:

$$H(S) = -(0.3 \log_2 0.3 + 0.3 \log_2 0.3 + 0.4 \log_2 0.4)$$

Using log values:

$$\log_2 0.3 \approx -1.74, \quad \log_2 0.4 \approx -1.32$$

$$H(S) = -(0.3 \times -1.74 + 0.3 \times -1.74 + 0.4 \times -1.32)$$

$$H(S) = -(-0.522 - 0.522 - 0.528)$$

$$H(S) = 1.572$$

✅ **Final Answer: Entropy = 1.57**

---

2. Translation (Time Shifting)

Definition:

Translation (or shifting) means moving a signal **left (advance)** or **right (delay)** in the discrete domain.

Mathematical Formulation:

$$x[n-k] \quad \text{(Right shift by } k\text{)}$$

$$x[n+k] \quad \text{(Left shift by } k\text{)}$$

- **If $k > 0$, the signal moves to the right (delay).**
- **If $k < 0$, the signal moves to the left (advance).**

Example:

Given a discrete signal:

$$x[n] = \{1, 2, 3, 4, 5\}$$

Right Shift by 2 ($x[n-2]$)

$$x[n-2] = \{0, 0, 1, 2, 3, 4, 5\}$$

(The first two positions are filled with zeros.)

Left Shift by 2 ($x[n+2]$)

$$x[n+2] = \{3, 4, 5, 0, 0\}$$

(The last two positions become zeros.)

3. Shrinking (Compression) and Expansion (Stretching)

Definition:

- **Shrinking (Compression)** makes the signal **faster**, reducing the number of values.
- **Expansion (Stretching)** makes the signal **slower**, increasing the number of values.

Mathematical Formulation:

$$x[c \cdot n] \quad \text{(Shrink/Compress by factor } c\text{)}$$

$$x[n/c] \quad \text{(Expand/Stretch by factor } c\text{)}$$

- **If $c > 1$, the signal shrinks (compresses).**
- **If $0 < c < 1$, the signal expands (stretches).**

Example:

Given a discrete signal:

$$x[n] = \{1, 2, 3, 4, 5, 6, 7, 8\}$$

Shrink by 2 ($x[2n]$)

$$x[2n] = \{1, 3, 5, 7\}$$

(Every second value is taken, compressing the signal.)

Expand by 2 ($x[n/2]$)

$$x[n/2] = \{1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8\}$$

(Each value is repeated, stretching the signal.)

4. Flipping (Time Reversal)

Definition:

Flipping reverses the order of the signal.

Mathematical Formulation:

$$x[-n]$$

Example:

Given:

$$x[n] = \{1, 2, 3, 4, 5\}$$

Flipped Signal ($x[-n]$)

$$x[-n] = \{5, 4, 3, 2, 1\}$$

5. Combined Transformations

Example:

Given:

$$x[n] = \{1, 2, 3, 4, 5\}$$

1. Shift Right by 2 and Flip

1. Right shift by 2:

$$x[n-2] = \{0, 0, 1, 2, 3, 4, 5\}$$

2. Flip:

$$x[-(n-2)] = \{5, 4, 3, 2, 1, 0, 0\}$$

2. Shrink by 2 and Shift Left by 1

1. Shrink:

$$x[2n] = \{1, 3, 5\}$$

2. Shift left by 1:

$$x[2n+1] = \{3, 5, 0\}$$

Image Transformation: Kernel Filter Calculation

Image Transformation: Kernel Filter Calculation

In **image processing**, **kernel filtering** (also called convolution) is used to modify an image by applying a small **matrix (kernel)** to it. This process is widely used for **blurring, sharpening, edge detection, and noise reduction**.

1. Convolution Operation

Convolution is performed by **sliding** a filter (kernel) over an image and computing a weighted sum of pixel values.

Mathematical Formula

Given an image $I(x, y)$ and a filter $K(s, t)$, the output $G(x, y)$ after convolution is:

$$G(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} K(s,t) \cdot I(x+s, y+t)$$

where:

- $K(s, t)$ is the **kernel (filter)**
- $I(x, y)$ is the **image pixel at (x, y)**
- $G(x, y)$ is the **filtered pixel value**
- a and b are the **kernel half sizes** (for a 3×3 kernel, $a = b = 1$)

2. Example: Applying a 3×3 Kernel

Consider this **image (5×5 matrix)**:

$$I = \begin{bmatrix} 10 & 20 & 30 & 40 & 50 \\ 20 & 30 & 40 & 50 & 60 \\ 30 & 40 & 50 & 60 & 70 \\ 40 & 50 & 60 & 70 & 80 \\ 50 & 60 & 70 & 80 & 90 \end{bmatrix}$$

We apply the **Sharpening Kernel**:

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

Step 1: Select a Pixel (Center Pixel)

We apply the kernel to the **center pixel (50 at position (2,2))**:

$$\begin{bmatrix} 30 & 40 & 50 \\ 40 & 50 & 60 \\ 50 & 60 & 70 \end{bmatrix}$$

Step 2: Multiply Each Pixel by Kernel Value

$$G(2,2) = (0 \cdot 30) + (-1 \cdot 40) + (0 \cdot 50) + (-1 \cdot 40) + (5 \cdot 50) + (-1 \cdot 60) + (0 \cdot 50) + (-1 \cdot 60) + (0 \cdot 70)$$

$$= 0 - 40 + 0 - 40 + 250 - 60 + 0 - 60 + 0$$

$$= 50$$

Step 3: Repeat for the Entire Image

Repeat this process for every pixel in the image **(excluding the edges)**.

3. Common Filters and Their Effects

Filter Type	Kernel Matrix	Effect
Identity (No Change)	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	No effect on image
Blur (Averaging)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	Smooths image, reduces noise
Gaussian Blur	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	Softer blurring effect
Edge Detection (Sobel X)	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	Detects vertical edges
Edge Detection (Sobel Y)	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$	Detects horizontal edges
Sharpening	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	Enhances edges

4. Edge Handling Techniques

Since convolution requires neighboring pixels, handling the **image borders** is important.

Methods:

1. **Zero Padding** (default): Add extra 0's around the image.
2. **Replication Padding**: Extend edge values to maintain size.
3. **Cropping**: Reduce output size to avoid undefined pixels.

5. Summary

1. **Kernel filtering** is a fundamental technique in image processing.
2. **Convolution** is the mathematical operation used for applying filters.
3. Different **kernels produce different effects** (blur, sharpen, edge detection, etc.).
4. **Border handling** methods prevent undefined pixels.