Mathematical Foundations of AI

Fall 25

Introduction to the Class

Google Classroom Code: mgzmh4jv

Retake Policy

- Retake of missed assessment items (other than sessional/ final exam) will not be held. Student who misses an assessment item (other than sessional / final exam) is awarded zero marks in that assessment item i.e. late submission will not be accepted.
- For missed sessional/ final exam, exam retake/ pretake application along with necessary evidence are required to be submitted to the department secretary. The examination assessment and retake committee decides the exam retake/ pretake cases.

Plagiarism Policy

Plagiarism in project or sessional/ final exam will result in F grade in the course.

Plagiarism in an assignment will result in zero marks in the whole assignments category.

Course Description

This course is an introduction to key mathematical concepts at the heart of machine learning. The focus is on matrix methods and statistical models and features real-world applications

Mathematical topics covered include linear equations, regression, regularization, the singular value decomposition, iterative optimization algorithms, and probabilistic models.

Machine learning topics include the LASSO, support vector machines, kernel methods, clustering, dictionary learning, neural networks, and deep learning.

What is Artificial Intelligence?

Subfields

- Supervised learning,
- Unsupervised learning,
- Reinforcement learning.

Supervised Learning

- **Definition:**
  Learn a mapping from inputs x to outputs y using *labeled data*.
- **Mathematics Behind It:**
  - Function approximation: $y \approx f(x; \theta)$.
  - Optimization: minimize loss function $L(f(x), y)$.
- **Examples:**
  - **Regression:** Predict house prices from features.
  - **Classification:** Identify whether an email is spam or not.
- **AI Applications:**
  - Speech recognition, medical diagnosis, stock price prediction.

Unsupervised Learning

- **Definition:**
  Find *hidden structure* in **unlabeled data** (no outputs given).
- **Mathematics Behind It:**
  - Clustering → partition data points (k-means, Gaussian Mixture Models).
  - Dimensionality reduction → map high-dimensional data to lower dimensions (PCA).
- **Examples:**
  - Grouping customers into segments based on buying habits.
  - Discovering topics in large collections of documents.
- **AI Applications:**
  - Market segmentation, recommendation systems, anomaly detection.

Reinforcement Learning (RL)

- **Definition:**
  Learn by interacting with an **environment** → agent chooses actions, receives rewards, improves over time.
- **Mathematics Behind It:**
  - Markov Decision Processes (MDPs).
  - Policy $\pi(a \mid s)$: probability of action given state.
  - Goal: maximize cumulative reward.
- **Examples:**
  - Training a robot to walk.
  - An AI agent playing chess or Go.
- **AI Applications:**
  - AlphaGo (DeepMind), self-driving cars, robotics.

Which Learning type?

- If you have lots of labeled medical scans → which method?
- If you have social media data with no labels → which method?
- If you want a robot to learn by trial & error → which method?

Which Learning type?

- If you have lots of labeled medical scans → which method? (Supervised)
- If you have social media data with no labels → which method? (Unsupervised)
- If you want a robot to learn by trial & error → which method? (Reinforcement Learning)

Why Mathematics Matters:

- Linear algebra → representations, embeddings.
- Calculus → optimization, backpropagation.
- Probability/statistics → uncertainty & decision making.
- Optimization → training models.
- Information theory → measuring learning.

Math → Algorithm → Application

- Vectors & Norms → Cosine Similarity → Search Engines
- Eigenvalues & SVD → PCA → Facial Recognition
- Probability & Bayes → Naive Bayes → Spam Detection

Vectors & Norms → Cosine Similarity → Search Engines

- **Math:**
  - Vector norms ($\|x\|_2$ = length of vector).
  - Inner product ($\langle x, y \rangle$).
  - Cosine similarity = $(x \cdot y) / (\|x\| \|y\|)$.
- **AI Algorithm:**
  - Cosine similarity used in **k-NN, document similarity, embeddings**.
- **Application:**
  - Google search ranks results by vector similarity between query & documents.

Mini-Demo: Embedding + Cosine Similarity

Example Sentences: *"The cat sits on the mat."* and *"A kitten rests on the rug."*

- **Step-by-Step:**
- **Tokenize**: Split into words (ignore case/punctuation).
  - Sentence 1: ["the", "cat", "sits", "on", "the", "mat"]
  - Sentence 2: ["a", "kitten", "rests", "on", "the", "rug"]
- **Vocabulary**: Unique words from both sentences → ["the", "cat", "sits", "on", "mat", "a", "kitten", "rests", "rug"]
- **Vectors** (Bag-of-Words):
  - Sentence 1: [2, 1, 1, 1, 1, 0, 0, 0, 0] and Sentence 2: [1, 0, 0, 1, 0, 1, 1, 1, 1]
- **Cosine Similarity Formula**:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\| \mathbf{A} \| \| \mathbf{B} \|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

- **Result**: Raw BoW similarity ≈ 0.236
- **Key Takeaway**:
  Even simple math (cosine similarity) captures *semantic closeness* in AI
- Higher value → More similar meaning (e.g., "cat" ≈ "kitten", "mat" ≈ "rug").
- Foundation for advanced embeddings (Word2Vec, BERT, etc.).

Python + NumPy Quickstart

- Import NumPy.
- Create vectors and matrices.
- Compute dot product, norms, cosine similarity.
- Show how this can classify two short documents (toy NLP example).

```python
import numpy as np

doc1 = np.array([1, 2, 3])
doc2 = np.array([2, 1, 0])

cosine_similarity = np.dot(doc1, doc2) / (np.linalg.norm(doc1) * np.linalg.norm(doc2))
print("Cosine Similarity:", cosine_similarity)
```

Eigenvalues & SVD → PCA → Dimensionality Reduction

- **Math:**
  - Eigen decomposition: $Av = \lambda v$.
  - SVD decomposes matrix into $U\Sigma V^T$.
- **AI Algorithm:**
  - PCA projects high-dimensional data onto fewer dimensions (principal components).
- **Application:**
  - Facial recognition: compress images while preserving most variance.
  - Data preprocessing in almost every ML pipeline.
- *SVD approximates a matrix (image) by keeping only the most important singular values/vectors—dramatically reducing size while preserving features.*

Represent Image as a Matrix

- Grayscale image = 2D matrix (e.g., 512x512 pixels).
- Color image = 3 matrices (R, G, B channels).

Step 2: Apply SVD

- For each channel matrix **A** (size m×n):

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

- **U**: Left singular vectors (orthogonal, m×m).
- **Σ**: Diagonal matrix of singular values (sorted in descending order, m×n).
- **V^T**: Right singular vectors (orthogonal, n×n).

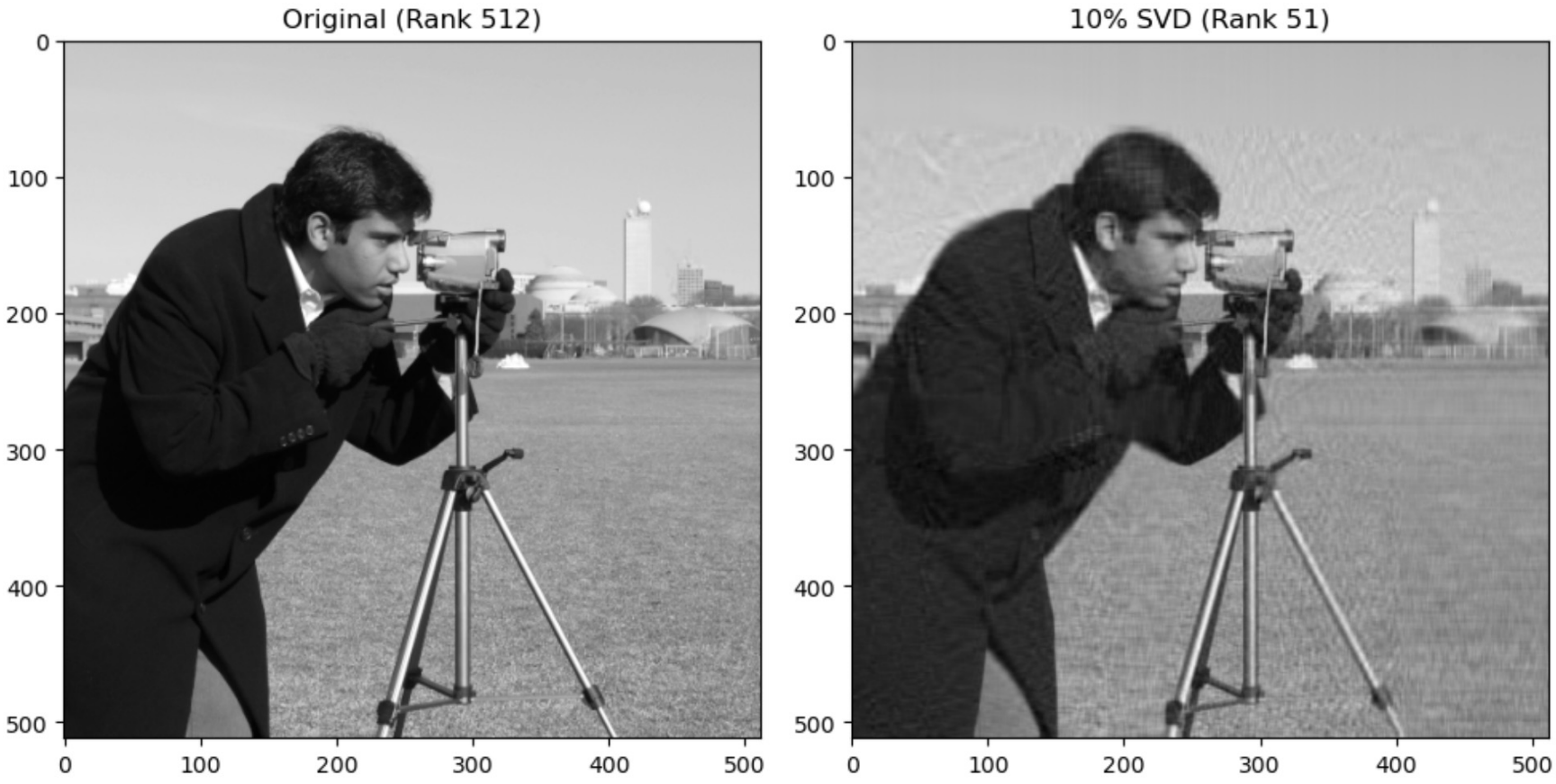
Step 3: Truncate to 10% Components

- Original rank = min(m, n) (e.g., 512).
- Keep only top **k = 10% × 512 ≈ 51** singular values (set the rest to 0).
- Approximate **A** with rank-k:

$$A_{compressed} = U_{\{:,1:k\}} \cdot \Sigma_{\{1:k,1:k\}} \cdot V_{\{1:k,:T\}}$$

Step 4: Reconstruct & Compare

- **Storage Saved**: Original: m×n vs. Compressed: k×(m + n + 1).
  - For 512×512 → Original: 262,144 values → Compressed: 51×(512 + 512 + 1) = 52,275 (**80% smaller**).


Original (Rank 512)
10% SVD (Rank 51)
0
100
200
300
400
500

Probability & Bayes → Naive Bayes → Spam Detection

- **Math:**
  - Bayes' Theorem:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

  - Random variables, conditional probability.
- **AI Algorithm:**
  - Naive Bayes classifier: assumes features are conditionally independent given the label.
- **Application:**
  - Gmail spam filter decides *spam vs not spam* using probabilities.

Spam Detection

A tiny dataset to compute P(word|spam) manually and classify emails, perfect for teaching the fundamentals of Naive Bayes:

1. Toy Dataset (5 Emails)

Email	Label (Spam/Ham)
"Win money fast!"	Spam
"Meeting today at 5pm"	Ham
"Lottery prize claim now"	Spam
"Project update attached”	Ham
"Free vacation click here"	Spam

Spam Detection

2. Preprocessing
   - Tokenize (lowercase, ignore punctuation):
   - Spam: ["win", "money", "fast", "lottery", "prize", "claim", "now", "free", "vacation", "click", "here"]
   - Ham: ["meeting", "today", "at", "5pm", "project", "update", "attached"]
3. Vocabulary: All unique words (18 total): ["win", "money", "fast", "lottery", "prize", "claim", "now", "free", "vacation", "click", "here", "meeting", "today", "at", "5pm", "project", "update", "attached"]
   - Compute P(word|Spam) & P(word|Ham)
   - Spam: "free"=1, "win"=1, "money"=1, "click"=1, ...
   - Ham: "meeting"=1, "today"=1, "project"=1, ...

Assignments (Ungraded)

- **Short Essay (1 page):** Pick one AI application (e.g., recommendation systems, self-driving cars, fraud detection) and explain which areas of math are most critical for it.
- **Coding Task:** Implement cosine similarity for 3–5 small text vectors and rank similarity.