**Table of Contents**

Table of Contents

[Question 1: Choose a topic from the course 1](#_Toc180744814)

[Question 2: Give a real-world AI problem which 7](#_Toc180744815)

[Question 3: Discuss your understanding on algorithmic fairness and sketch 8](#_Toc180744816)

[Question 4: Discuss one practical challenge in modern AI 9](#_Toc180744817)

[Question 4: Let X be a random variable. Suppose that E[X] = 1. What are the practical implications? 10](#_Toc180744818)

[Question 5: An outlier is a data point that behaves abnormally, 11](#_Toc180744819)

[Question 6: Give one example and one counterexample to each of the following statements, 13](#_Toc180744820)

[Question 7: Given an image, suppose that an expert will present 14](#_Toc180744821)

[Question 8: Consider that there is such a learning algorithm A that by loading n labeled samples into memory, 17](#_Toc180744822)

[Question 9: Both principal component analysis (PCA) and random 18](#_Toc180744823)

[Question 10: The classic PAC learning model of Valiant’84 made two fundamental assumptions: 20](#_Toc180744824)

[Question 11:Random projection (RP) is a widely used tool for dimension reduction 22](#_Toc180744825)

[Question 12: State the main idea of collaborative filtering. What are the possible drawbacks 24](#_Toc180744826)

[Question 13: Consider two functions F1(w) and F2(w): both of them are strongly convex, 25](#_Toc180744827)

[Question 14: Many machine learning problems boil down to solving the following optimization program 26](#_Toc180744828)

[Question 15: Suppose we have the following data from n patients: 29](#_Toc180744829)

[Question 16: Suppose we have the following data from n patients: 31](#_Toc180744830)

[Question 17: We learned from the class that if a function is convex and L-smooth 32](#_Toc180744831)

[Question 18: Stochastic gradient descent (SGD) bears the benefits 35](#_Toc180744832)

[Question 19: Suppose that you have a network of many computers 36](#_Toc180744833)

[Question 20: Consider the following data points (represented by circles) in 2-dimensional space. 37](#_Toc180744834)

[Question 20: In real-world applications the positive and negative classes 38](#_Toc180744835)

[Question 21: The error type of false positive is defined as follows: 39](#_Toc180744836)

# Question 1: Choose a topic from the course

that you are most interested in, and talk about your understanding.

Interesting topic is from Machine learning

1. Supervised learning  
   2. Unsupervised learning  
   3. Semi-supervised learning

**1.Supervised Learning:**

**Definition:** Supervised learning is a type of machine learning where a model is trained on a labeled dataset. This means that each training example is paired with an output label. The goal of supervised learning is for the model to learn a mapping from inputs (features) to outputs (labels) so that it can predict the output for new, unseen data.

**How It Works:**

**Data Collection**: Gather a dataset consisting of input-output pairs. Each input example is typically represented as a vector of features, and each output is the corresponding label.

**Model Selection**: Choose an appropriate algorithm for the task. Common algorithms include:

Linear regression

Logistic regression

Decision trees

Support vector machines (SVM)

Neural networks

**Training**: Use the labeled data to train the model. The training process involves adjusting the model’s parameters to minimize the difference between the predicted outputs and the actual labels (often measured using a loss function).

**Validation and Testing**: Split the dataset into training, validation, and test sets. The validation set helps tune model parameters, while the test set evaluates the model's performance on unseen data.

**Prediction**: Once the model is trained and validated, it can make predictions on new data.

**Evaluation**: Assess the model's performance using metrics like accuracy, precision, recall, F1-score, or mean squared error (depending on the task).

**Example of Supervised Learning: Email Classification**

**Problem Statement:** Imagine you want to build a model that classifies emails as either "spam" or "not spam."

**Step 1: Data Collection** You collect a dataset of emails that have been labeled as "spam" or "not spam." Each email is represented by features extracted from its content, such as the presence of certain keywords, the length of the email, and the number of hyperlinks.

| **Email Content** | **Label** |
| --- | --- |
| "Congratulations! You've won..." | Spam |
| "Meeting tomorrow at 10 AM." | Not Spam |
| "Special offer just for you!" | Spam |
| "Let’s catch up soon." | Not Spam |

**Step 2: Model Selection** You decide to use a logistic regression model for this binary classification problem.

**Step 3: Training** Using the labeled dataset, you train the logistic regression model. The model learns to identify patterns that differentiate spam from non-spam emails based on the input features.

**Step 4: Validation and Testing** You split your data into training (80%) and testing (20%) sets. You validate the model’s performance on the training set and fine-tune any parameters.

**Step 5: Prediction** After training, you can use the model to predict whether new, unseen emails are spam or not based on their features.

**Step 6: Evaluation** You evaluate the model's performance on the test set. You might find that the model achieves an accuracy of 90%, meaning it correctly classifies 90% of the test emails.

**Conclusion**

Supervised learning is powerful for tasks where historical labeled data is available. Its effectiveness depends on the quality and quantity of the data, as well as the choice of algorithm and model tuning. This approach is widely used in various applications, including classification tasks like email filtering, sentiment analysis, and medical diagnosis.

**Supervised learning** can be further divided into several different types, each with its own unique characteristics and applications. Here are some of the most common types of supervised learning algorithms:

[**Linear Regression**](https://www.geeksforgeeks.org/ml-linear-regression/): Linear regression is a type of supervised learning regression algorithm that is used to predict a continuous output value. It is one of the simplest and most widely used algorithms in supervised learning.

[**Logistic Regression**](https://www.geeksforgeeks.org/understanding-logistic-regression/): Logistic regression is a type of supervised learning classification algorithm that is used to predict a binary output variable.

[**Decision Trees**](https://www.geeksforgeeks.org/decision-tree/): Decision tree is a tree-like structure that is used to model decisions and their possible consequences. Each internal node in the tree represents a decision, while each leaf node represents a possible outcome.

[**Random Forests**](https://www.geeksforgeeks.org/random-forest-regression-in-python/): Random forests again are made up of multiple decision trees that work together to make predictions. Each tree in the forest is trained on a different subset of the input features and data. The final prediction is made by aggregating the predictions of all the trees in the forest.

[**Support Vector Machine(SVM)**](https://www.geeksforgeeks.org/support-vector-machine-algorithm/): The SVM algorithm creates a hyperplane to segregate n-dimensional space into classes and identify the correct category of new data points. The extreme cases that help create the hyperplane are called support vectors, hence the name Support Vector Machine.

[**K-Nearest Neighbors**](https://www.geeksforgeeks.org/k-nearest-neighbours/)**(KNN) :**KNN works by finding k training examples closest to a given input and then predicts the class or value based on the majority class or average value of these neighbors. The performance of KNN can be influenced by the choice of k and the distance metric used to measure proximity.

[**Gradient Boosting**](https://www.geeksforgeeks.org/ml-gradient-boosting/): Gradient Boosting combines weak learners, like [decision trees](https://www.geeksforgeeks.org/decision-tree/), to create a strong model. It iteratively builds new models that correct errors made by previous ones.

[**Naive Bayes Algorithm**](https://www.geeksforgeeks.org/naive-bayes-classifiers/): The Naive Bayes algorithm is a supervised machine learning algorithm based on applying [Bayes’ Theorem](https://www.geeksforgeeks.org/bayes-theorem/) with the “naive” assumption that features are independent of each other given the class label.

**Advantages of Supervised Learning**

The power of **supervised learning** lies in its ability to accurately predict patterns and make data-driven decisions across a variety of applications. Here are some advantages of **supervised learning** listed below:

**Supervised learning** excels in accurately predicting patterns and making data-driven decisions.

**Labeled training data** is crucial for enabling **supervised learning models** to learn input-output relationships effectively.

**Supervised machine learning** encompasses tasks such as **supervised learning classification** and **supervised learning regression**.

Applications include complex problems like image recognition and natural language processing.

Established evaluation metrics (accuracy, precision, recall, F1-score) are essential for assessing **supervised learning model** performance.

Advantages of **supervised learning** include creating complex models for accurate predictions on new data.

**Supervised learning** requires substantial labeled training data, and its effectiveness hinges on data quality and representativeness.

**Disadvantages of Supervised Learning**

Despite the benefits of **supervised learning methods**, there are notable **disadvantages of supervised learning**:

[**Overfitting**](https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/): Models can overfit training data, leading to poor performance on new data due to capturing noise in **supervised machine learning**.

[**Feature Engineering**](https://www.geeksforgeeks.org/what-is-feature-engineering/): Extracting relevant features is crucial but can be time-consuming and requires domain expertise in **supervised learning applications**.

**Bias in Models:**Bias in the training data may result in unfair predictions in **supervised learning algorithms**.

**Dependence on Labeled Data**: **Supervised learning** relies heavily on labeled training data, which can be costly and time-consuming to obtain, posing a challenge for **supervised learning techniques**.

**2.Unsupervised Learning:**

**Definition:** Unsupervised learning is a type of machine learning where models are trained on data without labeled outputs. The goal is to identify patterns, groupings, or structures in the data without any prior knowledge of what those patterns might be.

Unsupervised machine learning is a type of machine learning where algorithms learn from data that has **no pre-defined labels or categories**. In contrast to supervised learning where the training data is labeled (think "cat" pictures and "dog" pictures), unsupervised learning algorithms are tasked with finding hidden patterns or structures within the data itself.

**How It Works:**

**Data Collection**: Gather a dataset that contains input features but no associated output labels.

**Model Selection**: Choose an appropriate algorithm based on the desired outcome. Common algorithms include:

K-means clustering

Hierarchical clustering

Principal Component Analysis (PCA)

Autoencoders

t-Distributed Stochastic Neighbor Embedding (t-SNE)

**Training**: The model analyzes the input data to find hidden patterns, group similar data points, or reduce dimensionality.

**Evaluation**: Since there are no labeled outputs, evaluating the model often relies on intrinsic measures, such as silhouette scores for clustering or explained variance for dimensionality reduction.

**Application**: Use the insights gained from the unsupervised learning process for tasks such as data exploration, anomaly detection, or feature extraction.

**Example of Unsupervised Learning: Customer Segmentation**

**Problem Statement:** A retail company wants to segment its customers into different groups based on their purchasing behavior to tailor marketing strategies.

**Step 1: Data Collection** The company collects data on customer transactions, including features such as:

Age

Annual income

Spending score (a score assigned by the company based on purchasing behavior)

| **Customer ID** | **Age** | **Annual Income** | **Spending Score** |
| --- | --- | --- | --- |
| 1 | 25 | 50,000 | 60 |
| 2 | 45 | 120,000 | 80 |
| 3 | 35 | 90,000 | 40 |
| 4 | 50 | 60,000 | 70 |
| 5 | 23 | 30,000 | 30 |

**Step 2: Model Selection** The company decides to use the K-means clustering algorithm to segment the customers based on the features.

**Step 3: Training** Using the K-means algorithm, the company sets a predetermined number of clusters (e.g., 3). The algorithm iteratively assigns customers to clusters based on their similarities in age, income, and spending score.

**Step 4: Evaluation** After clustering, the company evaluates the cohesion and separation of the clusters using metrics like the silhouette score, which measures how similar an object is to its own cluster compared to other clusters.

**Step 5: Application** The results might show three distinct customer segments:

**Segment 1**: Young, lower income, low spending (e.g., age 20-30, income < $50,000)

**Segment 2**: Middle-aged, high income, moderate spending (e.g., age 35-50, income $80,000-$120,000)

**Segment 3**: Older, diverse income, high spending (e.g., age 50+, varying income)

The company can now tailor marketing strategies for each segment, offering discounts to Segment 1 to increase engagement, while providing premium products to Segment 3.

**Conclusion**

Unsupervised learning is a powerful technique for discovering hidden structures in data when no labels are available. It is widely used for clustering, dimensionality reduction, and anomaly detection across various fields, including marketing, finance, and healthcare. By understanding the underlying patterns in data, organizations can make more informed decisions and drive strategic initiatives.

**Challenges of Unsupervised Learning**

• Exploratory data analysis — goal is not always clearly defined

• Difficult to assess performance — “right answer” unknown

• Working with high-dimensional data

**Types of Unsupervised Learning - Two approaches:**

• Cluster analysis - For identifying homogenous subgroups of samples

• Dimensionality reduction - For finding a low-dimensional representation to characterize and visualize the data

**Clustering** A set of methods for finding subgroups within the dataset.

• Observations should share common characteristics within the same group, but differ across groups.

• Groupings are determined from attributes of the data itself — differs from classification.

A diagram of a class

Description automatically generated with medium confidenceA diagram of different shapes and colors

Description automatically generated

Types of Clustering

• **Centroid-based clustering**

**• Hierarchical clustering**

**• Model-based clustering**

- Each cluster is represented by a parametric distribution

- Dataset is a mixture of distributions

• **Hard vs. soft/fuzzy clustering**

- Hard: observations divided into distinct clusters

- Soft: observations may belong to more than one cluster

**What is K-Means Clustering?**

K-means clustering is a popular unsupervised [**machine learning algorithm**](https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/) used for partitioning a dataset into a pre-defined number of clusters. The goal is to group similar data points together and discover underlying patterns or structures within the data.

Recall the first property of clusters – it states that the points within a cluster should be similar to each other. So, our aim here is to minimize the distance between the points within a cluster.

There is an algorithm that tries to minimize the distance of the points in a cluster with their centroid – the k-means clustering technique.

K-means is a centroid-based algorithm or a distance-based algorithm, where we calculate the distances to assign a point to a cluster. In K-Means, each cluster is associated with a centroid.

**The main objective of the K-Means algorithm is to minimize the sum of distances between the points and their respective cluster centroid.**

Optimization plays a crucial role in the k-means clustering algorithm. The goal of the optimization process is to find the best set of centroids that minimizes the sum of squared distances between each data point and its closest centroid

**Here’s how it works:**

**Initialization**: Start by randomly selecting K points from the dataset. These points will act as the initial cluster centroids.

**Assignment**: For each data point in the dataset, calculate the distance between that point and each of the K centroids. Assign the data point to the cluster whose centroid is closest to it. This step effectively forms K clusters.

**Update centroids**: Once all data points have been assigned to clusters, recalculate the centroids of the clusters by taking the mean of all data points assigned to each cluster.

**Repeat**: Repeat steps 2 and 3 until convergence. Convergence occurs when the centroids no longer change significantly or when a specified number of iterations is reached.

**Final Result**: Once convergence is achieved, the algorithm outputs the final cluster centroids and the assignment of each data point to a cluster.

The main objective of k-means clustering is to partition your data into a specific number (k) of groups, where data points within each group are similar and dissimilar to points in other groups. It achieves this by minimizing the distance between data points and their assigned cluster’s center, called the centroid.

Here’s an objective:

**Grouping similar data points:** K-means aims to identify patterns in your data by grouping data points that share similar characteristics together. This allows you to discover underlying structures within the data.

**Minimizing within-cluster distance:** The algorithm strives to make sure data points within a cluster are as close as possible to each other, as measured by a distance metric (usually Euclidean distance). This ensures tight-knit clusters with high cohesiveness.

**Maximizing between-cluster distance:** Conversely, k-means also tries to maximize the separation between clusters. Ideally, data points from different clusters should be far apart, making the clusters distinct from each other.

**What is Clustering?** Cluster analysis is a technique in data mining and machine learning that groups similar objects into clusters. K-means clustering, a popular method, aims to divide a set of objects into K clusters, minimizing the sum of squared distances between the objects and their respective cluster centers.

Hierarchical clustering and k-means clustering are two popular techniques in the field of unsupervised learning used for clustering data points into distinct groups. While k-means clustering divides data into a predefined number of clusters, hierarchical clustering creates a hierarchical tree-like structure to represent the relationships between the clusters.

**3.Semi-Supervised Learning**

**Definition:** Semi-supervised learning is a machine learning approach that combines both labeled and unlabeled data for training. This method is particularly useful when acquiring labeled data is expensive or time-consuming, while unlabeled data is abundant.

Self-supervised learning (SSL) is a paradigm in machine learning where a model is trained on a task using the data itself to generate supervisory signals, rather than relying on external labels provided by humans.

**Steps to Perform Semi-Supervised Learning**

**Data Collection**:

Gather a dataset that contains a small amount of labeled data and a large amount of unlabeled data.

**Data Preprocessing**:

Clean and preprocess the data (e.g., normalization, handling missing values).

Extract relevant features from both labeled and unlabeled datasets.

**Model Selection**:

Choose an appropriate learning model. Common algorithms for semi-supervised learning include:

Self-training

Co-training

Graph-based methods

Generative models

**Initial Training**:

Train the model on the labeled dataset to create an initial classifier.

**Label Propagation**:

Use the trained model to predict labels for the unlabeled data. This can be done iteratively, where the model's confidence is used to assign labels to the unlabeled instances.

**Retraining**:

Combine the labeled and newly labeled data to retrain the model, improving its performance.

**Evaluation**:

Evaluate the model using a test set (often containing labeled data) to assess its performance and generalization.

**Advantages of Semi-Supervised Learning**

**Efficiency**:

Reduces the amount of labeled data needed, lowering costs and time for labeling.

**Improved Performance**:

Leverages unlabeled data to improve the model’s generalization, often leading to better performance than purely supervised learning with limited labeled data.

**Scalability**:

Can scale well with large datasets, where labeling all data is impractical.

**Disadvantages of Semi-Supervised Learning**

**Assumption Dependency**:

Many methods rely on the assumption that the labeled and unlabeled data are drawn from the same distribution, which may not always hold.

**Potential for Error Propagation**:

If the initial model is not accurate, incorrect labels assigned to unlabeled data can lead to a decrease in model performance.

**Complexity**:

The implementation can be more complex compared to fully supervised learning due to the integration of unlabeled data.

**Example of Semi-Supervised Learning**

**Problem Statement**: Consider a text classification problem where you have a small set of labeled documents (e.g., emails labeled as "spam" or "not spam") and a large collection of unlabeled emails.

**Step 1: Data Collection**

Labeled dataset: 200 emails (100 spam, 100 not spam)

Unlabeled dataset: 10,000 emails

**Step 2: Data Preprocessing**

Clean the text data (remove punctuation, lowercasing, etc.).

Extract features using methods like TF-IDF or word embeddings.

**Step 3: Model Selection**

Choose a model, e.g., a Support Vector Machine (SVM) or a neural network.

**Step 4: Initial Training**

Train the model on the 200 labeled emails.

**Step 5: Label Propagation**

Use the trained model to predict labels for the 10,000 unlabeled emails. For example, if the model is confident that certain emails are spam, assign those labels.

**Step 6: Retraining**

Combine the labeled dataset (200 emails) with the newly labeled emails (from the 10,000) to retrain the model.

**Step 7: Evaluation**

Test the final model on a separate labeled test set to assess accuracy, precision, recall, etc.

**Methods Used in Semi-Supervised Learning**

**Self-training**:

A model is trained on the labeled data and then used to label the unlabeled data. This labeled data is then added to the training set for further training.

**Co-training**:

Two or more models are trained on different views or features of the data. Each model labels the unlabeled data for the other model to improve performance.

**Graph-based Methods**:

Represent the data as a graph where nodes are data points and edges represent similarities. Algorithms like Label Propagation use the graph structure to propagate labels from labeled to unlabeled points.

**Generative Models**:

Models that can generate data points, like Variational Autoencoders (VAEs), can be used to exploit the structure in the unlabeled data.

**Conclusion**

Semi-supervised learning is a powerful approach that strikes a balance between the benefits of supervised and unsupervised learning. It is particularly useful in scenarios where acquiring labeled data is challenging, allowing practitioners to make the most of both labeled and unlabeled datasets for improved model performance.

­­Question 2: Give a real-world AI problem which,

from your point of view, cannot be solved in five years. If you are going to work on it, what is your first step to approach the main challenges?

One real-world AI problem that is likely to remain unsolved in the next five years is **general artificial intelligence (AGI)**—creating machines that possess human-like understanding, reasoning, and adaptability across a wide range of tasks and domains.

**Challenges of AGI**

**Complexity of Human Cognition**: Human intelligence involves not just logic and reasoning but also emotional understanding, social skills, and common sense knowledge, which are difficult to replicate in machines.

**Integration of Knowledge**: AGI requires the ability to integrate knowledge from diverse domains and apply it flexibly in novel situations.

**Learning Efficiency**: Humans can learn effectively from very few examples, while current AI models typically require vast amounts of data.

**Safety and Ethics**: Ensuring that AGI behaves in a safe and ethical manner poses significant challenges, including addressing biases and making ethical decisions.

**First Steps to Approach the Challenges**

If I were to work on this problem, my first step would be to **focus on developing a better understanding of human cognition** and learning processes. Here’s how I would approach it:­­­

**Interdisciplinary Research**:

Collaborate with neuroscientists, psychologists, and cognitive scientists to study how humans learn, reason, and understand the world.

Investigate how humans acquire common sense knowledge and develop emotional intelligence.

**Prototype Development**:

Build simplified models that mimic specific aspects of human cognition (e.g., common sense reasoning, emotional responses).

Use these models to explore how they can be integrated into existing AI systems.

**Data Efficiency**:

Research techniques for few-shot and zero-shot learning, allowing models to learn from minimal data, mimicking human learning capabilities.

**Ethical Frameworks**:

Develop ethical frameworks and guidelines to ensure that any advancements toward AGI prioritize safety, fairness, and alignment with human values.

**Community Engagement**:

Engage with the AI ethics community to discuss the implications of AGI and build consensus on best practices for its development.

By focusing on understanding human intelligence and learning processes, I believe we can make incremental advancements that may eventually contribute to the broader goal of AGI, while addressing its associated challenges.

# Question 3: Discuss your understanding on algorithmic fairness and sketch

**your own approach to fortify an AI algorithm with fairness.**

**Definition**: Algorithmic fairness refers to the principle that AI systems should make decisions that are unbiased and equitable across different demographic groups, ensuring that no group is unfairly disadvantaged or discriminated against. It encompasses various aspects, including fairness in data representation, model behavior, and the impacts of algorithmic decisions.

**Key Concepts in Algorithmic Fairness**

Types of Fairness:

Demographic Parity: Ensures that outcomes are distributed equally among different groups (e.g., gender, race).

Equal Opportunity: Ensures that individuals who qualify for a positive outcome (e.g., being hired) have equal chances of being selected, regardless of group membership.

Calibration: Ensures that predicted probabilities reflect true outcomes across different groups.

Sources of Bias:

Data Bias: Arises when training data reflects historical biases or imbalances.

Model Bias: Results from the algorithmic choices made during model development.

Feedback Loops: Can occur when biased decisions reinforce existing disparities over time.

Approach to Fortifying an AI Algorithm with Fairness

Data Audit and Preprocessing:

Conduct a Bias Audit: Analyze the training data for imbalances or biases related to protected attributes (e.g., race, gender).

Data Augmentation: If underrepresented groups are identified, augment the dataset to achieve a more balanced representation.

Feature Selection: Carefully choose features to ensure that irrelevant or biased features do not influence the model.

Fairness-Conscious Model Training:

Fairness Constraints: Implement constraints during the training process that enforce fairness criteria (e.g., demographic parity).

Adversarial Debiasing: Use adversarial training techniques where a secondary model predicts group membership from the main model’s predictions, incentivizing the main model to reduce biases.

Evaluation Metrics:

Diverse Fairness Metrics: Use multiple fairness metrics to assess model performance across different groups. This may include demographic parity, equal opportunity, and disparate impact ratios.

Regular Performance Monitoring: Continuously evaluate the model’s predictions in real-world scenarios to ensure sustained fairness.

Post-Hoc Adjustments:

Calibration Techniques: Use calibration methods to adjust the predicted probabilities to ensure they are reflective of true outcomes across groups.

Fairness-Aware Algorithms: Apply algorithms designed for fairness adjustment after the model has been trained, such as re-weighting or re-sampling techniques.

Stakeholder Engagement:

Inclusive Design: Involve diverse stakeholders in the design process to gather insights and perspectives on fairness.

Transparency and Accountability: Clearly communicate the model’s decision-making process and establish accountability measures to address potential fairness issues.

Iterative Improvement:

Feedback Mechanisms: Implement mechanisms to gather feedback from users and affected communities, allowing for continuous improvement of the model.

Adaptation to Changes: Stay responsive to societal changes that might impact fairness considerations, adjusting the model and its inputs accordingly.

Conclusion

Fortifying an AI algorithm with fairness requires a holistic approach that addresses data, model training, evaluation, and stakeholder engagement. By embedding fairness principles into the entire lifecycle of the AI system—from data collection to deployment—organizations can better ensure that their algorithms promote equity and justice in their outcomes.

Question 4: Discuss one practical challenge in modern AI applications, and propose three potential approaches.

**Practical Challenge in Modern AI Applications: Data Privacy**

Challenge Overview: In the era of big data, AI systems often require large amounts of personal data to train and perform effectively. This raises significant concerns regarding data privacy, as individuals are increasingly wary of how their data is collected, stored, and used. Regulatory frameworks like GDPR and CCPA impose strict guidelines on data handling, making it challenging for organizations to develop AI applications while ensuring compliance and maintaining user trust.

**Proposed Approaches to Address Data Privacy**

**Differential Privacy:**

Concept: Implement differential privacy techniques to allow AI models to learn from data without exposing individual data points. This approach adds random noise to the data or to the model outputs, ensuring that the risk of re-identifying individuals is minimized.

Implementation: Organizations can apply differential privacy during data aggregation and model training. For instance, while training a model on user data, they can add noise to the gradients, ensuring that the model's output does not reveal information about any single user.

Benefits: This technique allows organizations to utilize valuable data insights while protecting individual privacy, fostering user trust.

**Federated Learning:**

Concept: Federated learning is a decentralized approach where the model is trained across multiple devices without the need to centralize the data. Instead of sending raw data to a central server, each device computes updates to the model locally and only shares the aggregated updates.

Implementation: Organizations can deploy federated learning frameworks, allowing users' devices to participate in training. For example, a health app could train a model on user data locally, contributing to a central model without sharing sensitive health information.

Benefits: This approach preserves user privacy by keeping sensitive data on devices, while still enabling the development of robust AI models.

**Data Anonymization and Minimization:**

Concept: Implement robust data anonymization techniques to remove personally identifiable information (PII) from datasets. Coupled with data minimization principles, this ensures that only the necessary data is collected and retained for AI training.

Implementation: Before using datasets for training, organizations can apply techniques such as k-anonymity, where data is modified so that individuals cannot be distinguished from others in a group. Additionally, they should review data retention policies to ensure that only essential data is kept.

Benefits: Anonymizing data reduces the risk of privacy breaches and complies with regulatory requirements while allowing organizations to extract insights from data without compromising individual privacy.

**Conclusion**

Data privacy remains a significant challenge in modern AI applications, but implementing approaches like differential privacy, federated learning, and data anonymization can help organizations navigate this issue effectively. By prioritizing user privacy, organizations can build trust and ensure compliance with regulations, ultimately fostering a more ethical and responsible AI landscape.

# Question 4: Let X be a random variable. Suppose that E[X] = 1. What are the practical implications?

**If we further know that Var[X] = 10, what are the practical implications?**

**Practical Implications of E[X]=1E[X] = 1E[X]=1**

The expected value E[X]=1E[X] = 1E[X]=1 indicates that, on average, the random variable XXX takes a value of 1. Here are some practical implications:

**Average Outcome**: If XXX represents a quantity of interest (e.g., sales revenue, customer satisfaction score, etc.), an expected value of 1 suggests that, across many observations, the average result is 1. This provides a baseline for evaluating performance.

**Decision-Making**: If decisions are made based on the expected outcomes (e.g., financial forecasting or resource allocation), knowing that E[X]=1E[X] = 1E[X]=1 helps set targets and benchmarks.

**Risk Assessment**: The expected value alone doesn’t provide insights into variability. Hence, stakeholders should consider other metrics (like variance) for a complete understanding of the situation.

**Practical Implications of Var[X]=10Var[X] = 10Var[X]=10**

The variance Var[X]=10Var[X] = 10Var[X]=10 measures the spread of the values of XXX around the mean. A variance of 10 implies the following:

**High Variability**: A variance of 10 indicates that the values of XXX can vary significantly from the expected value of 1. This high variability suggests that while the average outcome is 1, individual observations can be much lower or higher.

**Uncertainty**: The presence of a high variance indicates a level of uncertainty associated with predictions or estimates involving XXX. For example, if XXX is used in risk management or financial projections, stakeholders must be cautious due to the potential for outcomes far from the mean.

**Impact on Strategy**: In a business context, understanding that XXX can fluctuate widely might lead to more conservative planning. Companies might set contingency measures in place to accommodate the potential for outcomes that deviate significantly from the expected value.

**Combined Implications of E[X]=1E[X] = 1E[X]=1 and Var[X]=10Var[X] = 10Var[X]=10**

When considered together, knowing that E[X]=1E[X] = 1E[X]=1 and Var[X]=10Var[X] = 10Var[X]=10 provides a clearer picture:

**Risk vs. Reward**: Stakeholders need to balance the average expected outcome with the inherent risk (variability) associated with it. Strategies might involve preparing for scenarios where XXX could be far from 1, while still aiming to achieve that average.

**Use of Confidence Intervals**: To understand the range of possible outcomes, stakeholders could construct confidence intervals around the expected value. Given the standard deviation 10≈3.16\sqrt{10} \approx 3.1610​≈3.16, a significant portion of values (about 68% if normally distributed) would fall within approximately 1 ± 3.16.

**Informed Decisions**: Decision-makers can use these statistics to inform policies or actions. For instance, knowing the average helps in setting targets, while the variance prompts a deeper analysis into why the variability exists and how it can be managed.

In summary, E[X]=1E[X] = 1E[X]=1 provides a foundational expectation, while Var[X]=10Var[X] = 10Var[X]=10 emphasizes the need for caution and deeper analysis due to the significant variability of outcomes.

Question 5: An outlier is a data point that behaves abnormally, that its magnitude is out of control**.**

**For example, consider the age of a patient in his transcript: it is an outlier if it shows age = 2000, or if age = 25.5. Even if we observe a “normal” value of 85 is may also be an outlier provided that his actual age is 8. Outlier may incur for a variety of reasons, e.g. human mistakes. In this case, we have to take such dirty data into algorithmic design. Consider the simple regression model: yi =⟨ai,wtrue⟩+ei, ∥wtrue∥0 ≤k, i=1,...,n. (0.1) In the above expression, ai ∈ Rd is the feature vector, wtrue is the groundtruth model we aim to estimate, ei ∈ R is the possible outlier for the ith sample and yi ∈ R is the corrupted response. • Given {ai,yi}ni=1 for sufficiently large n, say n → ∞, is it possible to learn the true model without any prior knowledge/condition on {ei}ni=1? • Now suppose that only n1 samples are corrupted by outliers where n1 ≪ n. In other words, among {ei}ni=1, n − n1 of them are zeros (but we do not know which of them have zero values). Give a proper formulation under which it is possible to simultaneously recover wtrue and detect the outliers.**

A white text on a white background

Description automatically generated

**Learning the True Model in the Presence of Outliers:**

**A screenshot of a paper

Description automatically generated**

**A math problem with numbers and equations

Description automatically generated with medium confidence**

A close-up of black text

Description automatically generated

# Question 6: Give one example and one counterexample to each of the following statements,

**where X1 and X2 are random variables.**

A white background with black text

Description automatically generated

**Understanding the Statements**

Before we provide examples and counterexamples, let's clarify the statements:

**E[X1X2] = E[X1]E[X2]:** This statement is true if X1 and X2 are **independent** random variables. If they are not independent, the equality may not hold.

**Var[X1X2] = Var[X1]Var[X2]:** This statement is **generally not true**, even for independent random variables. The variance of the product of two random variables is more complex and depends on their specific distributions.

**Var[X1+X2] = Var[X1] + Var[X2]:** This statement is true if X1 and X2 are **independent** random variables. If they are not independent, the equality may not hold, and there will be an additional term accounting for the covariance between X1 and X2.

**Examples and Counterexamples**

**Statement 1: E[X1X2] = E[X1]E[X2]**

**Example (Independence):**

**X1:** The result of a coin flip (1 for heads, 0 for tails).

**X2:** The result of another independent coin flip.

**Independence:** The outcome of one flip does not affect the outcome of the other.

**Equality:** E[X1X2] = 0.25 = E[X1]E[X2].

**Counterexample (Dependence):**

**X1:** The number of heads in two coin flips.

**X2:** The number of tails in the same two coin flips.

**Dependence:** X1 and X2 are not independent, as their values are related.

**Inequality:** E[X1X2] = 0.5 ≠ E[X1]E[X2] = 0.25.

**Statement 2: Var[X1X2] = Var[X1]Var[X2]**

**Example (Independence):**

**X1:** A standard normal random variable.

**X2:** Another independent standard normal random variable.

**Independence:** The two variables are uncorrelated.

**Inequality:** Var[X1X2] is not equal to Var[X1]Var[X2] in general, even for independent standard normal random variables. The exact value depends on the specific properties of the joint distribution.

**Counterexample (Dependence):**

**X1:** A uniform random variable on [0, 1].

**X2:** 1 - X1.

**Dependence:** X1 and X2 are perfectly negatively correlated.

**Inequality:** Var[X1X2] = 0 ≠ Var[X1]Var[X2] = 1/12.

**Statement 3: Var[X1+X2] = Var[X1] + Var[X2]**

**Example (Independence):**

**X1:** A standard normal random variable.

**X2:** Another independent standard normal random variable.

**Independence:** The two variables are uncorrelated.

**Equality:** Var[X1+X2] = 2 = Var[X1] + Var[X2].

**Counterexample (Dependence):**

**X1:** A uniform random variable on [0, 1].

**X2:** 1 - X1.

**Dependence:** X1 and X2 are perfectly negatively correlated.

**Inequality:** Var[X1+X2] = 0 ≠ Var[X1] + Var[X2] = 1/6.

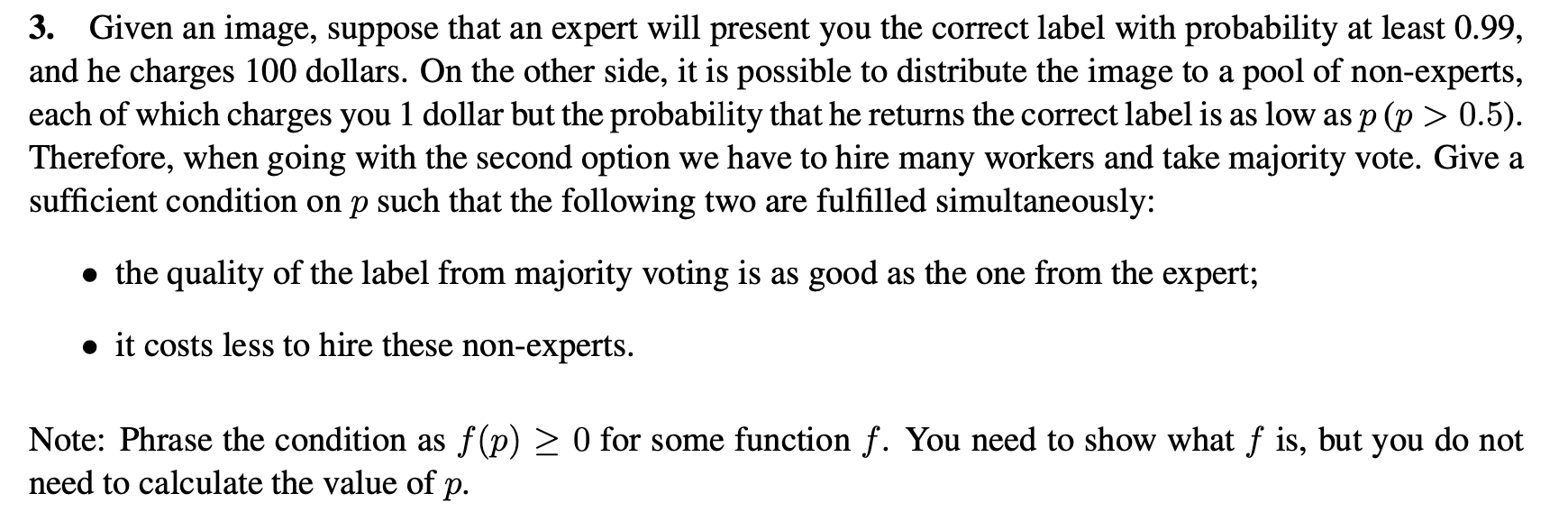
Question 7: Given an image, suppose that an expert will present you the correct label with probability at least 0.99,

**and he charges 100 dollars. On the other side, it is possible to distribute the image to a pool of non-experts, each of which charges you 1 dollar but the probability that he returns the correct label is as low as p (p > 0.5). Therefore, when going with the second option we have to hire many workers and take majority vote. Give a sufficient condition on p such that the following two are fulfilled simultaneously:**

**• the quality of the label from majority voting is as good as the one from the expert;**

**• it costs less to hire these non-experts.**

**Note: Phrase the condition as f(p) ≥ 0 for some function f. You need to show what f is, but you do not need to calculate the value of p.**



**Answer:**

A paper with mathematical equations

Description automatically generated

A page of a math problem

Description automatically generated

**Another solution:** We have two options:

Expert: 100 dollars, correct label with probability at least 0.99.

Non-experts: 1 dollar each, correct label with probability p > 0.5. We need to hire multiple non-experts and take a majority vote.

We want to find a condition on p such that:

The quality of the label from majority voting is as good as the one from the expert.

It costs less to hire non-experts.

**Formalizing the Conditions**

A page of a math problem

Description automatically generated

# Question 8: Consider that there is such a learning algorithm A that by loading n labeled samples into memory,

**it eturns a hypothesis hn with error rate less than 1/ sqrt (n)  
• Assume we hope to learn a hypothesis with error rate ε = 2−200. Find the minimum sample size n.**

**• Suppose that the samples lie in Rd with d = 220 and we use double-precision floating-point format (64 bits) to represent a real. What is the memory cost to store the samples as you calculated above.**

**• Is it realistic to learn such a model on a single computer? If not, propose an alternative approach.**

A white paper with black text

Description automatically generated

Answer:

A screenshot of a math test

Description automatically generated A screenshot of a text

Description automatically generated

**Another Method:**

A white paper with black text

Description automatically generated A screenshot of a computer

Description automatically generated

Question 9: Both principal component analysis (PCA) and random projection (RP) are widely used tools for dimension

**reduction. From a unified perspective, the only difference between them lies on the projection matrix. Suppose the data matrix Y ∈ Rd×n. PCA projects Y onto U⊤1:rY where U1:r consists of the first r columns of U which is produced by singular value decomposition. In contrast, RP transforms the data into AY where A ∈ Rr×d is a random matrix (typically Gaussian).**

**• Give an example where PCA might be a better choice than RP;  
• Give another example where RP is preferred;  
• What are the main drawbacks of both methods, and what is your solution?**

A white paper with black text

Description automatically generated

Answer:  
**When to Choose PCA**

**Example:** Image compression.

**Reason:** PCA is particularly effective when there's a strong underlying linear structure in the data. In images, this often manifests as correlations between pixel values. PCA can identify the principal components (directions of maximum variance) and project the data onto these components, capturing most of the important information with fewer dimensions.

**When to Choose RP**

**Example:** High-dimensional data with no clear linear structure.

**Reason:** RP is computationally efficient and can be applied to very large datasets. It's a good choice when the data is high-dimensional and there's no clear linear relationship between features. RP can randomly project the data onto a lower-dimensional space, preserving the pairwise distances between points reasonably well.

**Drawbacks and Solutions**

**PCA:**

**Sensitive to outliers:** Outliers can significantly influence the principal components, leading to suboptimal results.

**Assumes linearity:** PCA assumes a linear relationship between features. If the data is nonlinear, PCA may not capture the underlying structure effectively.

**Solution:** Consider robust PCA variants or nonlinear dimensionality reduction techniques like t-SNE or UMAP.

**RP:**

**Loss of information:** RP can introduce some distortion in the data. The amount of information preserved depends on the number of dimensions projected onto.

**Randomness:** The results of RP can vary slightly due to the random nature of the projection matrix.

**Solution:** Repeat RP multiple times and take the average of the results to reduce the impact of randomness.

**Unified Perspective and Solution:**

Both PCA and RP aim to reduce dimensionality while preserving essential information. The choice between them depends on the specific characteristics of the data and the desired trade-off between accuracy and computational efficiency. In some cases, a hybrid approach combining elements of both PCA and RP might be beneficial, such as using PCA to identify the most important directions and then applying RP to further reduce dimensionality.

**Another Method:**

**1. When PCA is a Better Choice than RP**

Example: Data with Correlated Features

If the data has features that are highly correlated, PCA is often the better choice because it identifies and projects the data onto the principal components that maximize variance.

Reasoning:

PCA will effectively reduce dimensionality while preserving as much variance as possible. By capturing the directions of maximum variance, PCA can retain the structure of the data and reduce noise, which is particularly beneficial in cases where relationships among features are strong.

For instance, in a dataset containing measurements of physical attributes (e.g., height, weight, and body mass index), where features are correlated, PCA will yield meaningful components that summarize the key variations in the data.

**2. When RP is Preferred**

Example: Large Scale Data Processing

When working with very large datasets where speed and computational efficiency are critical, Random Projection (RP) can be a better choice.

Reasoning:

RP is computationally efficient and easy to implement. It requires less computational power and memory compared to PCA, especially because it does not require the computation of the covariance matrix or eigenvalue decomposition.

For example, in streaming data applications or real-time systems where data is continuously generated (like sensor data in IoT), using RP allows for fast dimensionality reduction without the overhead of performing SVD or PCA. This is particularly useful when dimensionality reduction needs to be performed quickly to enable further analysis or machine learning.

Main Drawbacks of Both Methods

PCA Drawbacks

Computational Complexity: PCA requires the computation of the covariance matrix and its eigenvalue decomposition, which can be computationally expensive for very large datasets.

Sensitivity to Outliers: PCA is sensitive to outliers, which can significantly affect the resulting principal components and lead to suboptimal projections.

RP Drawbacks

Lack of Interpretability: The random projections may not preserve the geometric properties of the data, leading to a loss of meaningful structure and interpretability.

Guaranteed Distortion: While RP can theoretically provide good approximations, there is no guarantee that it will preserve distances as well as PCA does, especially in lower dimensions.

Proposed Solutions

For PCA:

Incremental PCA: Use algorithms like Incremental PCA or Online PCA that allow for the processing of data in batches, making it feasible for large datasets. This can help mitigate the computational burden.

Outlier Detection and Removal: Implement pre-processing steps to identify and remove outliers before performing PCA to improve robustness.

For RP:

Combination with Other Techniques: Use RP in conjunction with other dimensionality reduction techniques. For example, you could first apply a simple filtering method to reduce noise, followed by RP for faster computation.

Evaluate Dimensionality Preservation: Use techniques such as distance preservation metrics to evaluate the effectiveness of RP and ensure that it meets the specific requirements of the application.

By understanding the strengths and weaknesses of PCA and RP, one can choose the appropriate method based on the data characteristics and the specific requirements of the analysis or application.

# Question 10: The classic PAC learning model of Valiant’84 made two fundamental assumptions:

**1) the distributions of the training data and testing data are the same; and 2) all instances are labeled correctly. The goal of PAC learning is to find a hypothesis whose error rate, i.e. the probability that it misclassifies a new sample, is upper bounded by ε ∈ (0, 1). Give two respective examples to illustrate that if either assumption is violated, PAC learning becomes impossible.**

**A black text on a white background

Description automatically generated**

**Answer:**

**Analyzing the PAC Learning Assumptions**

The classic PAC (Probably Approximately Correct) learning model by Valiant '84 makes two fundamental assumptions:

Identical Distributions: The training data and testing data are drawn from the same distribution.

Correct Labels: All instances in the training data are labeled correctly.

If either of these assumptions is violated, PAC learning can become impossible. Let's illustrate this with examples:

Violation of Assumption 1: Identical Distributions

Example:

Scenario: A machine learning model is trained on historical financial data to predict stock prices.

Violation: The distribution of the training data (historical data) may differ from the distribution of the testing data (future data) due to changing market conditions, economic events, or unforeseen circumstances.

Impossibility of PAC Learning: If the testing data comes from a significantly different distribution than the training data, the model may perform poorly, and it becomes impossible to guarantee a low error rate on unseen data.

Violation of Assumption 2: Correct Labels

Example:

Scenario: A dataset is used to train a model to classify images as cats or dogs. However, some images are incorrectly labeled due to human error or data corruption.

Violation: The training data contains mislabeled instances.

Impossibility of PAC Learning: If the model learns from mislabeled data, it may develop a biased hypothesis that performs poorly on correctly labeled data. This can make it impossible to achieve a low error rate on new, correctly labeled samples.

In both examples, the violation of the PAC learning assumptions leads to a mismatch between the training and testing environments, making it difficult or impossible to guarantee a low error rate on unseen data. This highlights the importance of ensuring that the training data is representative of the testing data and is free from errors.

**Another Solution**

The PAC (Probably Approximately Correct) learning model relies heavily on the assumptions of consistent distributions and correct labeling. Here are two examples that illustrate how violating each assumption can lead to the impossibility of PAC learning.

**1. Violating the Distribution Assumption**

**Example**: **Training Data from a Different Distribution**

Imagine a scenario where we want to train a model to classify images of cats and dogs. We collect a training dataset consisting solely of images of dogs taken in sunny outdoor environments. However, our testing dataset includes images of cats and dogs in various settings, including indoors and during nighttime.

**Explanation**:

**Training Distribution**: The training data has a distribution that only reflects sunny outdoor dog images.

**Testing Distribution**: The testing data encompasses a broader distribution that includes various scenarios and both cats and dogs.

When the distributions differ, the model trained on the training data will likely perform poorly on the testing data because it has not learned to recognize images from the broader distribution. Consequently, the model’s error rate may be significantly higher than the desired ϵ\epsilonϵ, leading to the conclusion that PAC learning is impossible under this violation. The model cannot generalize effectively to unseen data that does not match the training distribution.

**2. Violating the Correct Labeling Assumption**

**Example**: **Noisy Labels**

Consider a scenario where a researcher is training a spam filter. The training dataset consists of emails that have been labeled as "spam" or "not spam." However, due to errors in labeling, some legitimate emails are incorrectly marked as spam, and some spam emails are marked as legitimate.

**Explanation**:

**Labeling Errors**: Some instances in the training set are mislabeled due to human error or automated labeling inaccuracies. For example, a valid email from a bank might be labeled as spam, while a promotional email might be misclassified as legitimate.

When the model learns from this incorrectly labeled data, it may adopt incorrect patterns based on the noise introduced by the mislabels. As a result, the learned hypothesis will have an intrinsic error rate that exceeds ϵ\epsilonϵ, even if the model is capable of classifying correctly when given perfectly labeled instances. Thus, it cannot guarantee that the learned hypothesis will perform with a low error rate on new data, making PAC learning impossible under this assumption violation.

**Summary**

**First Example**: When the distribution of training data and testing data is different, the model trained on one distribution may not generalize to another, leading to poor performance and making PAC learning infeasible.

**Second Example**: When the training instances are not labeled correctly, the model may learn from incorrect data patterns, resulting in an error rate that cannot be controlled, thus violating the principles of PAC learning.

These examples highlight the critical importance of both assumptions in the PAC learning framework.

# Question 11:Random projection (RP) is a widely used tool for dimension reduction

**• What are the major advantages of RP?**

**• Give an example where RP fails to boost computational efficiency;  
• What do you think will happen if we use different types of random matrices, for example, discrete**

**matrix for random projection? What are possible benefits and what are potential issues?**

**Answer:**

**Major Advantages of Random Projection (RP)**

Computational Efficiency: RP is generally computationally less intensive than other dimensionality reduction techniques like PCA. It avoids the need for singular value decomposition (SVD) or eigenvalue decomposition, making it faster, especially for high-dimensional datasets.

Theoretical Guarantees: RP is grounded in the Johnson-Lindenstrauss lemma, which states that high-dimensional points can be embedded into a lower-dimensional space while preserving pairwise distances with high probability. This means RP can maintain the geometric structure of the data.

Simplicity: The implementation of RP is straightforward. Generating a random matrix and multiplying it with the data matrix is a simple process, which makes RP easy to integrate into existing workflows.

Scalability: RP scales well with large datasets, as it can handle incremental data addition without needing to recompute projections from scratch.

Example Where RP Fails to Boost Computational Efficiency

**Example: Highly Sparse Data**

In scenarios where the data is highly sparse, such as text data represented in a bag-of-words model or in high-dimensional sparse feature spaces, using RP might not yield computational efficiency.

Reasoning:

When projecting a sparse matrix onto a lower-dimensional space using a dense random matrix, the resulting matrix can become dense. This could lead to an increase in the number of non-zero entries, thereby negating the benefits of sparsity and increasing memory usage and computational costs in subsequent processing steps.

In such cases, other techniques that preserve sparsity, like feature selection or specialized sparse dimensionality reduction methods, might be more efficient.

Effects of Using Different Types of Random Matrices

1. Discrete Random Matrices

Possible Benefits:

Simplicity and Interpretability: Using discrete values (e.g., +1 or -1) can simplify computations, as matrix multiplication can be performed using bitwise operations, which are computationally cheaper.

Memory Efficiency: Discrete matrices may require less storage than continuous ones, especially in situations where binary representation can be effectively utilized.

Potential Issues:

Loss of Information: Discrete random projections can introduce quantization errors, leading to a loss of information, especially if the continuous data distribution is not well-represented by the discrete values.

Distortion of Distances: The use of discrete values may not preserve distances as effectively as continuous values, potentially leading to poorer performance in distance-based tasks, like clustering or nearest neighbor searches.

2. Other Random Matrix Types (e.g., Sparse or Structured Matrices)

Possible Benefits:

Preservation of Sparsity: Using sparse random matrices can help maintain the sparsity of the input data, which is beneficial in terms of memory and computational efficiency for large, sparse datasets.

Specific Structural Properties: Structured matrices, like orthogonal matrices, may provide better preservation of certain geometric properties, leading to more reliable embeddings.

Potential Issues:

Complexity of Construction: Creating structured or sparse random matrices can be more complex than generating simple Gaussian matrices, potentially negating the simplicity advantage of RP.

Computational Overhead: Some structured matrix types may introduce additional computational overhead in their construction or application, reducing the overall efficiency benefits of RP.

Summary

Random projection offers significant advantages in terms of computational efficiency, theoretical grounding, simplicity, and scalability. However, it may not always be the best choice, particularly in highly sparse datasets. When exploring different types of random matrices, benefits may include computational simplicity and memory efficiency, but potential issues like information loss and increased complexity must also be considered.

**Another Solution**

Advantages of Random Projection (RP)

Computational Efficiency: RP is computationally efficient, especially for large datasets. It involves simple matrix multiplication, which can be parallelized easily.

Dimensionality Reduction: RP can significantly reduce the dimensionality of data while preserving key information. This can speed up subsequent computations and reduce storage requirements.

Preservation of Distances: RP approximately preserves pairwise distances between data points, which is crucial for many machine learning tasks.

Universality: RP works well for a wide range of data types and distributions.

Example of RP Failing to Boost Computational Efficiency

Sparse Data: For extremely sparse data, the computational cost of matrix multiplication in RP can become significant. In such cases, specialized techniques like Locality-Sensitive Hashing (LSH) might be more efficient.

Impact of Different Random Matrices

Using different types of random matrices for RP can have varying effects:

Benefits

Tailored Properties: Specific types of random matrices can be chosen to achieve desired properties, such as preserving certain statistical moments or ensuring specific distributions.

Computational Efficiency: Some types of random matrices might lead to faster computations or require less storage.

Potential Issues

Performance Degradation: If the chosen random matrix is not suitable for the data or the task, it can lead to a decrease in performance.

Theoretical Guarantees: It might be more difficult to derive theoretical guarantees for the performance of RP using non-standard random matrices.

Examples of Different Random Matrices:

Gaussian Random Matrices: Commonly used due to their simplicity and theoretical properties.

Rademacher Random Matrices: Entries are drawn from {-1, 1} with equal probability.

Sparse Random Matrices: Can be more efficient for sparse data, but might require careful design to preserve information.

Structured Random Matrices: Matrices with specific structures (e.g., Hadamard matrices) can offer computational advantages in certain scenarios.

In conclusion, RP is a versatile tool for dimensionality reduction, offering computational efficiency and good performance in many cases. The choice of random matrix can impact the performance and theoretical guarantees, so it's essential to consider the specific characteristics of the data and the desired properties when selecting a suitable random matrix.

# Question 12: State the main idea of collaborative filtering. What are the possible drawbacks

**of its current formulation, and how to improve them?**

Main Idea of Collaborative Filtering

Collaborative filtering is a technique used in recommendation systems that predicts user preferences based on the collective behaviors and preferences of many users. The main idea is that if two users have similar preferences in the past, they are likely to have similar preferences in the future. Collaborative filtering typically works in two main ways:

User-Based Collaborative Filtering: This approach recommends items by finding similar users. If User A is similar to User B and User A likes an item, then User B is likely to also like that item.

Item-Based Collaborative Filtering: This method recommends items by finding similar items based on user ratings. If an item is similar to one that a user has liked in the past, it will be recommended to that user.

Possible Drawbacks of Collaborative Filtering

Cold Start Problem: Collaborative filtering struggles with new users or items that have insufficient data (few ratings), making it difficult to provide meaningful recommendations.

Sparsity of Data: In many real-world applications, user-item interaction matrices are very sparse. This sparsity can hinder the algorithm’s ability to find meaningful patterns and make accurate predictions.

Scalability Issues: As the number of users and items increases, the computational complexity of collaborative filtering can grow significantly, making it difficult to maintain performance.

Popularity Bias: Collaborative filtering can favor popular items, leading to a "filter bubble" where niche or less popular items are not recommended, limiting diversity in recommendations.

User Privacy Concerns: Collaborative filtering relies on user data, which raises privacy issues, especially if sensitive information is being used.

Possible Improvements

Addressing the Cold Start Problem:

Hybrid Approaches: Combine collaborative filtering with content-based filtering to utilize item attributes (e.g., genre, description) when there is insufficient user data.

Incorporate Demographic Data: Use demographic information (age, location) to make initial recommendations until sufficient user interaction data is available.

Handling Sparsity:

Matrix Factorization Techniques: Employ techniques like Singular Value Decomposition (SVD) or Alternating Least Squares (ALS) to reduce dimensionality and extract latent factors from sparse data.

Use of Clustering: Group users or items into clusters based on similarity to increase the density of interactions within those clusters.

Improving Scalability:

Approximate Nearest Neighbors: Use algorithms that can quickly identify similar users or items without needing to compare every entry in the dataset, such as locality-sensitive hashing (LSH).

Distributed Computing: Implement collaborative filtering on distributed systems (e.g., Apache Spark) to handle large datasets more efficiently.

Mitigating Popularity Bias:

Diversity Promotion: Introduce algorithms that explicitly promote diversity in recommendations by balancing popular and niche items.

Personalized Ranking: Adjust recommendation algorithms to focus on personalizing the ranking of items rather than merely selecting the top-N popular items.

Enhancing Privacy:

Anonymization Techniques: Implement data anonymization techniques to protect user identities while still allowing for effective recommendations.

Differential Privacy: Use differential privacy mechanisms to ensure that individual user data cannot be inferred from the recommendations provided.

By addressing these drawbacks, collaborative filtering can become a more robust and effective approach for generating personalized recommendations in various applications.

# Question 13: Consider two functions F1(w) and F2(w): both of them are strongly convex,

**but F1 is smooth and F2 is non-smooth. Suppose we apply GD to optimize these two functions. The following figure shows two convergence curves: a solid line and a dashed line. One is for F1(wt) and another for F2(wt).**

**Explain which curve may correspond to F1.**

**Plot a possible convergence curve when applying stochastic GD to optimize F1 in the same figure.**

**Now recall that SGD will randomly select a sample to compute the stochastic gradient in each itera- tion. Use another figure to plot the convergence curves of SGD with 5 different trials, where one trial means we restart SGD with the same initial point and same step size.**

**Finally, recall that the convergence guarantee of SGD is phrased in terms of expectation, but why do we often run it for one trial and still observe certain convergence property?**

**A graph of a function

Description automatically generated**

**Answer:**

**Explanation of the Convergence Curves**

1. Identifying the Curves for F1 and F2

In optimization, the convergence curve for a smooth function typically shows more consistent and rapid convergence compared to a non-smooth function.

Curve for F1 (Smooth and Strongly Convex): The solid line likely corresponds to F1(w)F1(w)F1(w). This curve would demonstrate a smooth decrease in the objective function value as iterations progress, reflecting the advantages of smoothness, which allows for faster convergence rates (often linear or even faster).

Curve for F2 (Non-Smooth and Strongly Convex): The dashed line likely represents F2(w)F2(w)F2(w). Non-smooth functions typically exhibit slower convergence behavior, often characterized by erratic progress, with potential plateaus or abrupt changes in function values as the algorithm navigates through the non-smooth landscape.

2. Stochastic Gradient Descent (SGD) Convergence for F1

When applying stochastic gradient descent (SGD) to optimize F1(w)F1(w)F1(w), the convergence curve will be more erratic than the curve obtained from deterministic gradient descent due to the inherent randomness in SGD. However, over many iterations, we can expect the curve to trend downward, indicating progress towards convergence.

Possible Plot

The solid line represents the convergence of F1(w)F1(w)F1(w) using deterministic GD.

The dashed line represents the convergence of F2(w)F2(w)F2(w).

The new curve for SGD applied to F1(w)F1(w)F1(w) might be a wavy line that oscillates around the expected convergence but trends downward overall.

3. SGD Trials with Random Sample Selection

When we run multiple trials of SGD (each starting from the same initial point and using the same step size), we can observe variability in the convergence due to the randomness of sample selection.

Possible Plot for Multiple Trials

Each trial would be represented by a separate curve, showing different paths to convergence, but with all curves generally trending downward as they approach the minimum of F1(w)F1(w)F1(w).

There might be variations in the rate of convergence and the oscillation around the optimal value, with some trials converging faster than others.

4. Expectation in SGD and Observed Convergence Properties

The convergence guarantees of SGD are usually framed in terms of the expected behavior of the algorithm. This means that, theoretically, if we run SGD many times (averaging across random samples), we would observe convergence to the optimum.

Why Do We Often See Convergence in One Trial?

Low Variance in Estimates: If the dataset is large or the objective function has favorable properties, the stochastic gradients may closely approximate the true gradient, leading to reasonable convergence even in a single run.

Effective Learning Rates: If the learning rate is appropriately chosen, SGD can still make substantial progress towards the minimum, overcoming the variance introduced by randomness.

Strong Convexity: Since both F1F1F1 and F2F2F2 are strongly convex, this property helps ensure that even with stochastic updates, the iterates tend to converge towards the unique global minimum.

Sufficient Number of Iterations: With enough iterations, the average behavior of SGD can dominate the random fluctuations, leading to a convergence trend even in one trial.

In summary, while the convergence guarantee of SGD is framed around expectation, the characteristics of the function being optimized, the choice of learning rate, and the number of iterations can lead to favorable outcomes in a single trial.

# Question 14: Many machine learning problems boil down to solving the following optimization program

**: minF(w), s.t.w∈Rd. (1) w Suppose that d = 2 and F (w) = 1/2 (w12 + (w1 + w2)2) where w1 and w2 are the first and second coordinates of w respectively. • Calculate the gradient and the Hessian matrix of F (w); • Show that F (w) is a strongly convex and smooth function, and calculate the tightest strong convexity parameter α and smoothness parameter L; • Consider that we run gradient descent (GD) to find the global optimum of F(w), starting from the initial iterate w0 = (1, 1) and proceed with learning rate η = 1/4. Calculate the iterates w1, w2, w3. • Suppose we are able to calculate more iterates w4,w5,...,wt,... with η = 1/4, and we plot the curve“ wt η = 2? Please plot them in the same figure and explain how you obtain these curves. 2 v.s.t”asbelow.IfwerunGDwithη=2/3,whatwillthecurvelikelybe?Whatabout 3 2.5 2 1.5 1 0.5 0 10 20 30 40 50 t ||wt || • Now consider minimizing the same function with stochastic GD, where the learning rate ηt = 1/t at the t-iteration. Draw “ wt 2 v.s. t” in the figure above.**

**A graph on a white background

Description automatically generated**

A white sheet with black text

Description automatically generated with medium confidence A group of math equations

Description automatically generated

A math equations and formulas on a white background

Description automatically generated

# Question 15: Suppose we have the following data from n patients**:**

A table of text with numbers and symbols

Description automatically generated with medium confidence

**Answer:**

To address the problem of estimating missing values in the "Blood Pressure" column based on the available patient data, we can formulate it as a machine learning problem, specifically as a regression problem. Here's how we can approach it:

**Problem Formulation**

**Data Representation**: We have a dataset where each patient is represented as a feature vector consisting of various attributes, such as Age, Weight, Height, Gender, and potentially some measurements related to health (e.g., Sharp Pain). The target variable we want to predict is Blood Pressure.

**A math equations and numbers

Description automatically generated**

**Solution Approach**

**Data Preprocessing**:

**Encoding**: If categorical variables (like Gender) are present, they should be converted into a numerical format using techniques like one-hot encoding.

**Normalization/Standardization**: Scale the continuous features to ensure they contribute equally to the model.

**Model Selection**: We can use regression techniques to estimate the missing Blood Pressure values. Some possible models include:

**Linear Regression**: A straightforward approach that assumes a linear relationship between the features and the target.

**Decision Trees**: Non-linear regression trees that can capture complex interactions between features.

**Random Forest**: An ensemble method that combines multiple decision trees for more robust predictions.

**K-Nearest Neighbors (KNN)**: A model that predicts values based on the average of the nearest neighbors' outcomes.

**Training**:

Use the rows of XXX and yyy where Blood Pressure values are known to train the model.

Split the data into training and validation sets to assess model performance.

**Prediction**:

Once the model is trained, use it to predict the missing values in yyy.

For patients with missing Blood Pressure values, pass their feature vectors from XXX into the trained model to obtain estimates.

**Evaluation**:

Evaluate the model’s performance using metrics such as Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE) on a validation set with known values.

**Imputation**:

Replace the missing entries in the Blood Pressure column with the predicted values.

**Summary**

In summary, we formulate the problem of estimating missing Blood Pressure values as a supervised regression task where we leverage other patient features to predict the target variable. By carefully selecting and training an appropriate model, we can effectively estimate the missing values and complete the dataset.

**Another Solution**

**Formulating the Problem as a Machine Learning Task**

**Problem Statement:** Given a dataset with missing values in a specific column (e.g., Blood Pressure), predict the missing values based on the observed values in the same and other columns.

**Machine Learning Task:** This is a **regression** problem. We're predicting a continuous numerical value (Blood Pressure) based on other features (Age, Weight, Height, Gender, etc.).

**Solution Approach: Multiple Imputation**

**Multiple Imputation** is a popular technique for handling missing data. It involves creating multiple completed datasets by filling in missing values with plausible values. These imputed values are based on the observed data and statistical models.

**Steps:**

**Imputation Model Selection:** Choose a suitable imputation model. Common choices include:

**Regression imputation:** Use regression models to predict missing values based on observed values.

**Hot deck imputation:** Replace missing values with values from similar cases.

**Stochastic imputation:** Use a probabilistic model to generate imputed values.

**Multiple Imputations:** Create multiple completed datasets by applying the chosen imputation model multiple times. This helps account for uncertainty in the imputation process.

**Analysis:** Analyze each completed dataset separately and combine the results to obtain a final estimate and measure the uncertainty associated with the imputation.

**Example Using Regression Imputation:**

**Split data:** Divide the dataset into two parts: one with complete cases and one with missing values.

**Build a regression model:** Use the complete cases to build a regression model predicting Blood Pressure based on Age, Weight, Height, Gender, and other observed features.

**Impute missing values:** Use the regression model to predict missing Blood Pressure values in the incomplete cases.

**Repeat:** Repeat steps 2 and 3 multiple times to create multiple imputed datasets.

**Advantages of Multiple Imputation:**

Accounts for uncertainty in imputation.

Preserves variability in the data.

Provides a more accurate estimate of standard errors.

**Additional Considerations:**

**Missing Data Pattern:** The pattern of missingness (e.g., missing at random, missing not at random) can influence the choice of imputation method.

**Imputation Model Complexity:** The complexity of the imputation model should be appropriate for the data and the missingness pattern.

**Sensitivity Analysis:** Conduct sensitivity analyses to assess the impact of different imputation methods and models on the results.

By following these steps and considering the factors mentioned above, you can effectively handle missing data in your dataset and obtain reliable estimates of the missing Blood Pressure values.

# Question 16: Suppose we have the following data from n patients:

**where for each column and each row there are some missing entries (represented by the symbol “?”).**

**• State how we can estimate all the missing values.**

**• Now consider that the above data matrix is fully observed, but some of its entries are corrupted by outliers. Give a proper formulation under which we are able to recover the clean data matrix.**

A screenshot of a test

Description automatically generated

To address the two parts of your question regarding handling missing values and outlier recovery in a data matrix, we can follow structured approaches.

**Part 1: Estimating Missing Values**

Given that the dataset has several missing entries in various columns and rows, we can use several methods to estimate the missing values. Here are some strategies:

**1. Data Imputation Techniques:**

**Mean/Median Imputation**: For each feature, replace missing values with the mean or median of the observed values for that feature.

**K-Nearest Neighbors (KNN) Imputation**: Use KNN to find the k nearest patients based on the available features and predict the missing value using the average of the k neighbors' values.

**Multivariate Imputation by Chained Equations (MICE)**: This approach models each feature with missing values as a function of other features in a round-robin fashion.

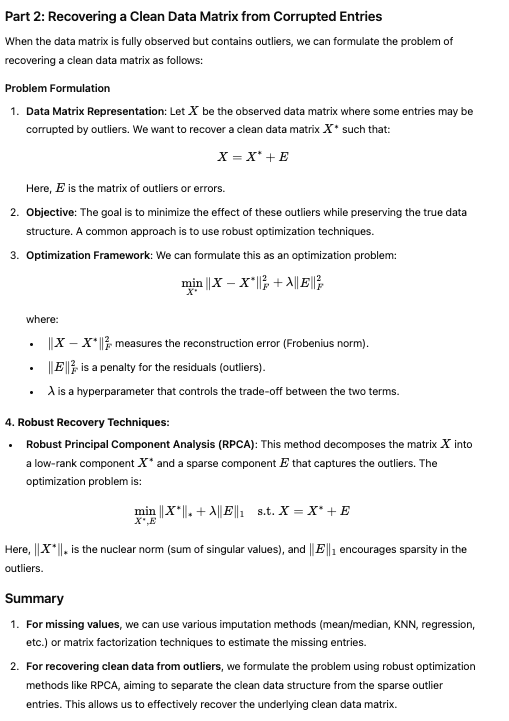
**Regression Imputation**: Train a regression model using the features that are completely observed to predict the missing values.

**2. Matrix Factorization:**

**Singular Value Decomposition (SVD)** or **Non-negative Matrix Factorization (NMF)** can be applied to approximate the complete data matrix. These methods can recover missing entries by finding a low-rank approximation of the matrix.

**3. Deep Learning Approaches:**

Use models like autoencoders to learn the underlying structure of the data and reconstruct the missing values.



Question 17: We learned from the class that if a function is convex and L-smooth, then the best learning rate for

gradient descent (GD) is L1. If a function is α-strongly convex and L-smooth, then the best learning rate for

GD

A math equations and formulas on a white sheet

Description automatically generated with medium confidence

A math equations and formulas on a white sheet

Description automatically generated with medium confidence

A white sheet of paper with black text

Description automatically generated

A text on a white background

Description automatically generated with medium confidence

Question 18: Stochastic gradient descent (SGD) bears the benefits of reducing overall computational complexity which makes it a prominent choice in large-scale machine learning. In order to apply SGD to obtain a convergent sequence, a key step is to (a) rewrite the objective function in the form F(w) = m1 Pmi fi(w) for some m and some function fi, and (b) ensure that for a uniformly chosen index i ∈ {1, 2, . . . , m}, Ei[∇fi(w)] = ∇F (w). This way SGD perform as GD in expectation which is a good news since we know GD reduces the objective value.

A math equations and formulas

Description automatically generated with medium confidence

Answer:

A screenshot of a math problem

Description automatically generatedA math equations and formulas

Description automatically generated with medium confidence

A white paper with black text

Description automatically generated

Question 19: Suppose that you have a network of many computers, each of which has limited storage space. State a strategy that allows you to learn a good classification model from a massive amount of data (“massive” means the whole data set cannot be stored in any single machine).

To learn a good classification model from a massive dataset that cannot fit on a single machine, you can employ a **distributed machine learning strategy**. Here’s a structured approach to tackle this challenge:

**1. Data Partitioning**

* **Divide the Dataset**: Split the massive dataset into smaller, manageable chunks and distribute these chunks across the available machines. Each machine will handle only a part of the data, ensuring that it fits within its storage constraints.

**2. Local Model Training**

* **Local Training**: Each machine trains a local model on its subset of the data. This could be done using algorithms that support incremental or mini-batch learning, allowing each machine to learn from its local data without needing to access the entire dataset.
* **Use of Simple Models**: Depending on the data size and complexity, use simpler models or algorithms that can converge quickly and efficiently on smaller datasets (e.g., logistic regression, decision trees, etc.).

**3. Model Aggregation**

* **Parameter Aggregation**: After local training, each machine shares its model parameters (weights) with a central server or designated aggregator. This can be done using:
  + **Federated Learning**: Each machine sends updates to the central model without sharing raw data, enhancing privacy and efficiency.
  + **Parameter Averaging**: Average the parameters received from each machine to create a global model.

**4. Iterative Refinement**

* **Iterative Process**: The global model is sent back to each machine. Each machine then uses this model as a starting point and continues training on its local data for several epochs.
* **Repeat**: This process of local training and parameter aggregation is repeated for several iterations until the model converges or meets a performance threshold.

**5. Handling Non-IID Data**

* **Data Distribution Considerations**: Ensure that the model accounts for potential non-IID (independently and identically distributed) data across different machines. Techniques such as weighted averaging or local updates can help mitigate biases introduced by uneven data distributions.

**6. Scalability and Fault Tolerance**

* **Distributed Frameworks**: Utilize distributed machine learning frameworks (e.g., TensorFlow, PyTorch, or Apache Spark MLlib) that provide built-in capabilities for handling large datasets and parallel processing.
* **Fault Tolerance**: Implement mechanisms to recover from machine failures, ensuring that the training process can continue smoothly even if some machines go offline.

**7. Model Evaluation**

* **Global Evaluation**: Once training is complete, evaluate the global model using a separate validation dataset. If possible, this dataset should be held on a central server or distributed across the machines to assess performance.

**Summary**

This distributed strategy allows you to leverage the computational power and storage of multiple machines to learn a good classification model from a massive dataset. By partitioning data, performing local training, and aggregating model parameters iteratively, you can efficiently manage resources and achieve scalability while maintaining model accuracy.

# Question 20: Consider the following data points (represented by circles) in 2-dimensional space.

• Illustrate the group structure discovered by sparse subspace clustering in the left panel;

A graph of a group of dots

Description automatically generated

Answer:

To illustrate the requested concepts—sparse subspace clustering and the clustering result of a single iteration of k-means with specific initial centers—I'll describe how you can visualize each method effectively.

**Left Panel: Sparse Subspace Clustering**

1. **Concept**: Sparse subspace clustering identifies clusters in high-dimensional data that reside in low-dimensional subspaces. This method discovers groups of points that lie within the same subspace, even if they are not necessarily close together in the overall feature space.
2. **Illustration**:
   * Create a 2D scatter plot with data points represented as circles.
   * Color the circles to represent different clusters identified by the sparse subspace clustering method. For instance, you could use different colors (e.g., red, blue, green) to show distinct groups.
   * If applicable, draw boundaries or lines to highlight the subspaces or regions in which each cluster resides.

**Right Panel: K-Means Clustering (Single Iteration)**

1. **Concept**: In k-means clustering, the algorithm iteratively assigns data points to the nearest cluster center and updates the cluster centers based on the assigned points.
2. **Initial Setup**:
   * You have 3 initial centers at (−10,0)(-10, 0)(−10,0), (0,−10)(0, -10)(0,−10), and (10,10)(10, 10)(10,10).
3. **Illustration**:
   * Create a 2D scatter plot of the same data points.
   * Mark the initial centers with distinct shapes (e.g., stars or crosses) and label them accordingly.
   * For the single iteration of k-means:
     + Assign each point to the nearest center based on Euclidean distance.
     + Change the color of each point according to the cluster to which it is assigned.
   * Optionally, draw lines connecting each point to its nearest center to show the assignment.

**K-means Clustering**

K-means clustering assigns each data point to the nearest centroid. In this case, with k = 3 and initial centers (-10, 0), (0, 10), and (10, 10), the clustering result would likely be:

* **Cluster 1:** Points close to (-10, 0)
* **Cluster 2:** Points close to (0, 10)
* **Cluster 3:** Points close to (10, 10)

**However, this result might not be optimal.** The initial centers can significantly influence the final clustering result. If the initial centers were chosen differently, the clustering could be quite different.

**Visualization**

**Left Panel (Sparse Subspace Clustering):**

* The two groups would likely be visually distinct, with points in each group forming a clear horizontal and vertical line.

**Right Panel (K-means Clustering):**

* The clustering result would likely show three distinct clusters, with points in each cluster being relatively close to their respective centroid. However, the boundaries between clusters might not be well-defined, especially if the data points are not well-separated.

**Note:** To get a more accurate visualization, it would be helpful to have the actual clustering results from both algorithms. This would allow for a direct comparison and a better understanding of the differences between the two methods.

**Summary of Visualization Steps**

* **Plot Creation**: Use a plotting library (like Matplotlib in Python) to generate the scatter plots.
* **Data Points**: Ensure data points are plotted as circles for both panels.
* **Coloring and Marking**: Use distinct colors for clusters and shapes for centers.
* **Annotations**: Label the clusters and centers for clarity.

This approach will help you visually convey the differences in clustering methodologies and their resulting group structures effectively.

# Question 20: In real-world applications the positive and negative classes

are often imbalanced. For example, there are a massive amount of data from airplanes that operate normally, while the number of aviation accidents is relatively few. Therefore, in order for a machine learning system to detect abnormal status, the algorithm must appreciate those negative examples (i.e. the data from flight recorder). Given a set of training samples {(xi, yi)}ni=1 ⊂ Rd × {+1, −1}, state how you will design a learning algorithm for such abnormality detection system.  
**Answer:**  
Designing a learning algorithm for an abnormality detection system in the context of imbalanced classes, such as detecting aviation accidents (rare positive examples) among normal flights (common negative examples), requires a strategic approach. Here’s a structured way to tackle this problem:  
**1. Data Understanding and Preprocessing**

**Exploratory Data Analysis**: Analyze the dataset to understand the distribution of classes, identify key features, and explore relationships between them.

**Feature Engineering**: Extract relevant features that could help differentiate between normal and abnormal statuses. This might include metrics from flight data, environmental conditions, and operational parameters.

**Handling Missing Values**: Address any missing or corrupted data points to ensure a clean dataset for training.

**2. Class Imbalance Handling**

**Resampling Techniques**:

**Oversampling**: Increase the number of positive examples (accident data) by duplicating existing examples or generating synthetic examples using methods like SMOTE (Synthetic Minority Over-sampling Technique).

**Undersampling**: Reduce the number of negative examples to balance the classes. This can be done randomly or by selecting representative samples.

**Cost-sensitive Learning**: Assign higher penalties (costs) to misclassifying the positive class. This encourages the model to focus more on correctly identifying rare events.

**3. Model Selection**

**Algorithm Choice**: Choose algorithms that are robust to class imbalance. Some suitable options include:

**Tree-based models** (e.g., Random Forest, Gradient Boosting) which can handle class imbalance well.

**Support Vector Machines (SVM)** with a custom kernel that gives more importance to minority class examples.

**Anomaly detection models** (e.g., Isolation Forest, One-Class SVM) specifically designed for identifying rare events.

**4. Model Training and Evaluation**

**Cross-validation**: Use stratified k-fold cross-validation to ensure that each fold maintains the same class distribution as the original dataset.

**Evaluation Metrics**: Since accuracy can be misleading in imbalanced scenarios, focus on metrics like:

**Precision, Recall, and F1-Score**: Particularly the F1-Score, which balances precision and recall.

**ROC-AUC (Receiver Operating Characteristic Area Under Curve)**: To assess the model's ability to discriminate between classes.

**Confusion Matrix**: To visualize the performance of the model in classifying both classes.

**5. Post-Processing and Calibration**

**Threshold Adjustment**: After model training, adjust the decision threshold based on the desired balance between precision and recall, especially if the costs of false positives and false negatives differ significantly.

**Ensemble Methods**: Consider using ensemble techniques like bagging or boosting to improve model robustness and generalization.

**6. Deployment and Monitoring**

**Real-time Monitoring**: Once deployed, continuously monitor the system's performance and adapt it to changing patterns in the data.

**Feedback Loop**: Incorporate a feedback mechanism to retrain the model with new data and improve detection as more incidents are logged.

**7. Documentation and Compliance**

**Regulatory Compliance**: Ensure that the model adheres to aviation industry regulations and standards, including safety and reporting requirements.

**Transparent Reporting**: Document the model's decisions, performance, and any biases to ensure transparency and facilitate trust among stakeholders.

**Summary**

This structured approach enables you to design a robust abnormality detection system tailored for the unique challenges of imbalanced datasets. By focusing on appropriate preprocessing, model selection, evaluation metrics, and continuous monitoring, you can effectively identify rare but critical events in aviation data.

Question 21: The error type of false positive is defined as follows: an algorithm outputs positive for a sample but in reality its label is negative. In some real-world applications, a learning algorithm should never incur the error of false positive. For example, the face recognition system of a laptop is designed such that it always denies unauthorized access to confidential data. Likewise, a self-driving car shall never recognize a red traffic light as green (but it is fine to classify green as red). Given a set of training samples {(xi, yi)}ni=1 ⊂ Rd × {+1, −1}, state a proper formulation which is capable of preventing false positive.

**Answer:**

To design a learning algorithm that prevents false positives—where the algorithm incorrectly classifies a negative instance as positive—you can adopt a strategy focused on strict control of the decision boundary and cost-sensitive learning. Here’s a structured formulation:

**A white sheet of paper with black text

Description automatically generated**

**4. Training Algorithm**

Use a suitable optimization algorithm (e.g., Gradient Descent, Adam) to minimize the overall loss function defined above. Ensure the model can handle the potential class imbalance and is trained with a strong emphasis on minimizing false positives.

**5. Post-Training Validation**

* **Threshold Adjustment**: After training, you may need to adjust the decision threshold for classifying samples as positive or negative. This can further help to ensure that false positives are minimized.
* **Validation Metrics**: Focus on metrics that emphasize the importance of avoiding false positives:
  + Precision: True Positives / (True Positives + False Positives).
  + Specificity: True Negatives / (True Negatives + False Positives).

**6. Deployment Considerations**

* **Continuous Monitoring**: Once deployed, continuously monitor the model's predictions and validate them against real-world outcomes to ensure no false positives occur.
* **User Feedback**: Implement a mechanism for user feedback to identify potential misclassifications and adapt the model accordingly.

**Summary**

This formulation creates a framework that prioritizes the avoidance of false positives through a custom loss function that penalizes false positives heavily, incorporates regularization, and ensures thorough validation and monitoring post-training. This is essential for applications where false positives can lead to significant consequences.