ADVANCED AI AND ML(21AI71)
Internal Assessment Test 2

1. Derive the gradient descent rule. Differentiate between gradient descent and stochastic gradient descent. (6+4)

$$\nabla E(\vec{w}) \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots, \frac{\partial E}{\partial w_n}\right] \tag{4.3}$$

$$\vec{w} \leftarrow \vec{w} + \Delta\vec{w}$$

where

$$\Delta\vec{w} = -\eta \nabla E(\vec{w}) \tag{4.4}$$

$$w_i \leftarrow w_i + \Delta w_i$$

where

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} \tag{4.5}$$

$$\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 \\
&= \frac{1}{2} \sum_{d \in D} \frac{\partial}{\partial w_i} (t_d - o_d)^2 \\
&= \frac{1}{2} \sum_{d \in D} 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) \\
&= \sum_{d \in D} (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - \vec{w} \cdot \vec{x}_d) \\
\frac{\partial E}{\partial w_i} &= \sum_{d \in D} (t_d - o_d)(-x_{id}) \qquad (4.6)
\end{aligned}$$

$$\Delta w_i = \eta \sum_{d \in D} (t_d - o_d)\, x_{id} \tag{4.7}$$

The key differences between standard gradient descent and stochastic gradient descent are:

- In standard gradient descent, the error is summed over all examples before updating weights, whereas in stochastic gradient descent weights are updated upon examining each training example.
- Summing over multiple examples in standard gradient descent requires more computation per weight update step. On the other hand, because it uses the true gradient, standard gradient descent is often used with a larger step size per weight update than stochastic gradient descent.
- In cases where there are multiple local minima with respect to E($, stochastic gradient descent can sometimes avoid falling into these local minima because it uses the various VEd(G) rather than VE(6) to guide its search.

2. Explain a genetic algorithm with a prototypical genetic algorithm. Explain the different genetic operators present in it. (6+4)

GA($Fitness$, $Fitness\_threshold$, p, r, m)

Fitness: A function that assigns an evaluation score, given a hypothesis.

Fitness_threshold: A threshold specifying the termination criterion.

p: The number of hypotheses to be included in the population.

r: The fraction of the population to be replaced by Crossover at each step.

m: The mutation rate.

- *Initialize population:* $P \leftarrow$ Generate p hypotheses at random
- *Evaluate:* For each h in P, compute $Fitness(h)$
- While $[\max_h Fitness(h)] < Fitness\_threshold$ do

  Create a new generation, P_S:

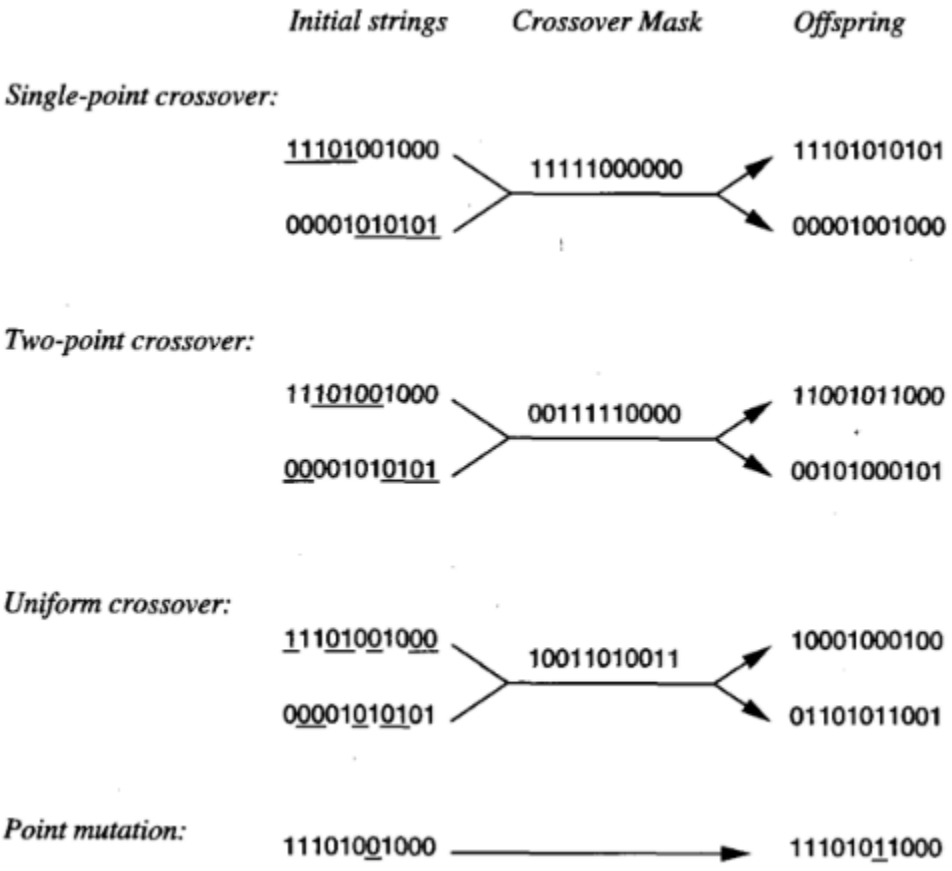
  1. *Select:* Probabilistically select $(1-r)p$ members of P to add to P_S. The probability $\Pr(h_i)$ of selecting hypothesis h_i from P is given by

$$\Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{p} Fitness(h_j)}$$

  2. *Crossover:* Probabilistically select $\frac{r \cdot p}{2}$ pairs of hypotheses from P, according to $\Pr(h_i)$ given above. For each pair, $\langle h_1, h_2 \rangle$, produce two offspring by applying the Crossover operator. Add all offspring to P_s.
  3. *Mutate:* Choose m percent of the members of P_s with uniform probability. For each, invert one randomly selected bit in its representation.
  4. *Update:* $P \leftarrow P_s$.
  5. *Evaluate:* for each h in P, compute $Fitness(h)$
- Return the hypothesis from P that has the highest fitness.

Typical GA operators for manipulating bit string hypotheses. These operators correspond to idealized versions of the genetic operations found in biological evolution. The two most common operators are crossover and mutation. The crossover operator produces two new offspring from two parent strings, by copying selected bits from each parent. The bit at position i in each offspring is copied from the bit at position i in one of the two parents. The choice of which

parent contributes the bit for position i is determined by an additional string called the crossover mask. In two-point crossover, offspring are created by substituting intermediate segments of one parent into the middle of the second parent string. Uniform crossover combines bits sampled uniformly from the two parents. of two parents, a second type of operator produces offspring from a single parent. In particular, the mutation operator produces small random changes to the bit string by choosing a single bit at random, then changing its value.



3. **Explain the K-nearest neighbour (K-NN) algorithm. Highlight the differences between it KNN and the distance-weighted K-nearest neighbour algorithm**. (5+5)

Training algorithm:

- For each training example $\langle x, f(x) \rangle$, add the example to the list *training_examples*

Classification algorithm:

- Given a query instance x_q to be classified,
  - Let $x_1 \ldots x_k$ denote the k instances from *training_examples* that are nearest to x_q
  - Return

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(v, f(x_i))$$

where $\delta(a, b) = 1$ if $a = b$ and where $\delta(a, b) = 0$ otherwise.

Basic Concept:

- K-NN:
  - In the standard K-NN algorithm, all the KKK nearest neighbors contribute equally to the prediction, regardless of their distance from the query point.
- Distance-Weighted K-NN:
  - In DW-KNN, closer neighbors have a higher influence on the prediction compared to farther ones. This weighting is based on the distance between the query point and the neighbors.

2. Weight Assignment:

- K-NN:
  - No weighting is applied. All KKK neighbors are treated equally. Each neighbor gets a weight of 1.
- DW-KNN:
  - Neighbors are weighted inversely proportional to their distance from the query point. For example, a common weighting function is: wi=1d(x,xi)+ϵw_i = \frac{1}{d(x, x_i) + \epsilon}wi=d(x,xi)+ϵ1 where d(x,xi)d(x, x_i)d(x,xi) is the distance between the query point xxx and neighbor xix_ixi, and ϵ\epsilonϵ is a small constant to avoid division by zero.

3. Prediction:

- K-NN:
  - Classification: The class label is determined by majority voting among the KKK neighbors.
  - Regression: The prediction is the mean (or sometimes median) of the target values of the KKK neighbors.

- DW-KNN:
    - Classification: The class label is determined by weighted voting, where the weights are inversely proportional to the distance.
    - Regression: The prediction is a weighted average of the target values, with weights determined by the distance.

4. Sensitivity to Distance:

- K-NN:
    - Less sensitive to the actual distances since all neighbors have equal influence.
- DW-KNN:
    - More sensitive to the distance, giving more importance to closer neighbors, which often leads to improved accuracy, especially in scenarios where closer points are more relevant.

5. Use Cases:

- K-NN:
    - Works well in scenarios where the relevance of neighbors does not vary much with distance.
- DW-KNN:
    - More effective when nearby data points are more likely to belong to the same class or have similar output values, such as in non-uniformly distributed datasets.

6. Performance:

- K-NN:
    - Simpler and computationally less intensive since it doesn't compute weights for neighbors.
- DW-KNN:
    - Slightly more computationally intensive due to the need for calculating weights, but often yields better performance in terms of accuracy.

4. **Explain the rule used to find the combination of items frequently bought.**
Explain the metrics used to calculate the frequent item set referring to Example: {Milk, Diaper}->{Beer}. (6+4)

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Association rule finds combinations of items that frequently occur together in orders or baskets (in a retail context). The items that frequently occur together are called itemsets. Itemsets help to discover relationships between items that people buy together and use that as a basis for creating strategies like combining products as combo offer or place products next to each other in retail shelves to attract customer attention. An application of association rule mining is in Market Basket
Analysis (MBA). MBA is a technique used mostly by retailers to find associations between items purchased by customers.
Association rule considers all possible combination of items in the previous baskets and computes various measures such as support, confidence, and lift to identify rules with stronger associations.

Metrics

Support indicates the frequencies of items appearing together in baskets with respect to all possible baskets being considered (or in a sample). Assume that X and Y are items being considered. Let

1. N be the total number of baskets.
2. N_{XY} represent the number of baskets in which X and Y appear together.
3. N_X represent the number of baskets in which X appears.
4. N_Y represent the number of baskets in which Y appears.

hen the support between X and Y, Support(X, Y), is given by

$$\text{Support}(X, Y) = \frac{N_{XY}}{N} \qquad (9.1)$$

9.2.1.2 Confidence

Confidence measures the proportion of the transactions that contain X, which also contain Y. X is called antecedent and Y is called consequent. Confidence can be calculated using the following formula:

$$\text{Confidence}(X \rightarrow Y) = P(Y \mid X) = \frac{N_{XY}}{N_X} \tag{9.2}$$

where $P(Y|X)$ is the conditional probability of Y given X.

9.2.1.3 Lift

Lift is calculated using the following formula:

$$\text{Lift} = \frac{\text{Support}(X, Y)}{\text{Support}(X) \times \text{Support}(Y)} = \frac{N_{XY}}{N_X N_Y} \tag{9.3}$$

s= ({Milk, Diaper, Beer}) \div |T|
= 2/5
= 0.4

c= s(Milk, Diaper, Beer) /s(Milk, Diaper)
= 2/3
= 0.67

l= Supp({Milk, Diaper, Beer}) /(Supp({Milk, Diaper})*Supp({Beer}))
= 0.4/(0.6*0.6)
= 1.11

5. Explain the different types of collaborative filtering with an example and the challenges associated with it. (4+4+2)

Collaborative filtering is based on the notion of similarity
Collaborative filtering comes in two variations:
1. User-Based Similarity: Finds K similar users based on common items they have bought.
2. Item-Based Similarity: Finds K similar items based on common users who have bought those items.
User-Based Collaborative Filtering is a technique used to predict the items that a user might like on the basis of ratings given to that item by other users who have similar taste with that of the target user. Many websites use collaborative filtering for building their recommendation system.

- Calculating Cosine Similarity between Users
- Filtering Similar Users

Item-Based Similarity

If two movies, movie A and movie B, have been watched by several users and rated very similarly, then movie A and movie B can be similar in taste. In other words, if a user watches movie A, then he or she is very likely to watch B and vice versa.

- Calculating Cosine Similarity between Movies
- Finding Most Similar Movies

Finding user similarity does not work for new users. We need to wait until the new user buys a few items and rates them. Only then users with similar preferences can be found and recommendations can be made based on that. This is called cold start problem in recommender systems. This can be overcome by using item-based similarity. Item-based similarity is based on the notion that if two items have been bought by many users and rated similarly, then there must be some inherent relationship between these two items. In other terms, in future, if a user buys one of those two items, he or she will most likely buy the other one.

6. **Explain the different steps in text preprocessing for sentiment analysis (Include relevant code snippets where necessary). List the challenges. of text analytics. (8+2)**

Text Pre-processing

Unlike structured data, features (independent variables) are not explicitly available in text data. Thus, we need to use a process to extract features from the text data. One way is to consider each word as a feature and find a measure to capture whether a word exists or does not exist in a sentence. This is called the bag-of-words (BoW) model. That is, each sentence (comment on a movie or a product) is treated as a bag of words. Each sentence (record) is called a document and collection of all documents is called corpus.

- Bag-of-Words (BoW) Model

  The first step in creating a BoW model is to create a dictionary of all the words used in the corpus. At this stage, we will not worry about grammar and only occurrence of the word is captured. Then we will convert each document to a vector that represents words available in the document. There are three ways to identify the importance of words in a BoW model:

  1. Count Vector Model
  2. Term Frequency Vector Model
  3. Term Frequency-Inverse Document Frequency (TF-IDF) Model

  - Count Vector Model

    Consider the following two documents:

    1. Document 1 (positive sentiment): I really really like IPL.
    2. Document 2 (negative sentiment): I never like IPL.

    Note: IPL stands for Indian Premier League.

    The complete vocabulary set (aka dictionary) for the above two documents will have words such as I, really, never, like, IPL. These words can be considered as features (x1 through x5). For creating count vectors, we count the occurrence of each word in the document as shown in Table 10.4. The y-column in Table 10.4

indicates the sentiment of the statement: 1 for positive and 0 for negative sentiment.

TABLE 10.4 Count vector for document 1 and document 2

Documents	x1	x2	x3	x4	x5	y
	I	really	never	like	ipl	
I really really like ipl	1	2	0	1	1	1
I never like ipl	1	0	1	1	1	0

Term Frequency Vector Model

Term frequency (TF) vector is calculated for each document in the corpus and is the frequency of each term in the document. It is given by,

$$\text{Term Frequency } (TF_i) = \frac{\text{Number of occurrences of word } i \text{ in the document}}{\text{Total number of words in the document}} \quad (10.1)$$

where TF_i is the term frequency for word (aka token). TF representation for the two documents is shown in Table 10.5.

TABLE 10.5 TF vector

	x1	x2	x3	x4	x5	y
	I	really	never	like	ipl	
I really really like ipl	0.2	0.4	0	0.2	0.2	1
I never like ipl	0.25	0	0.25	0.25	0.25	0

- Term Frequency-Inverse Document Frequency (TF-IDF)
  TF-IDF measures how important a word is to a document in the corpus. The importance of a word (or token) increases proportionally to the number of times a word appears in the document but is reduced by the frequency of the word present in the corpus. TF-IDF for a word i in the document is given by

$$TF - IDF_i = TF_i \times \ln\left(1 + \frac{N}{N_i}\right) \quad (10.2)$$

where N is the total number of documents in the corpus, N_i is the number of documents that contain word i.

The IDF value for each word for the above two documents is given in Table 10.6.

TABLE 10.6 IDF values

	x1	x2	x3	x4	x5
IDF Values	0.693	1.098	1.098	0.693	0.693

The TF-IDF values for the two documents are shown in Table 10.7

TABLE 10.7 TF-IDF values

	x1	x2	x3	x4	x5	y
	1	really	never	like	ipl	
I really really like ipl	0.1386	0.4394	0.0	0.1386	0.1386	1
I never like ipl	0.1732	0.0	0.2746	0.1732	0.1732	0

- Creating Count Vectors for sentiment_train Dataset
  Each document in the dataset needs to be transformed into TF or TF-IDF vectors. sklearn.feature_ extraction.text module provides classes for creating both TF and TF-IDF vectors from text data. We will use CountVectorizer to create count vectors. In CountVectorizer, the

documents will be represented by the number of times each word appears in the document.

- Removing Low-frequency Words
  One of the challenges of dealing with text is the number of words or features available in the corpus is too large. The number of features could easily go over tens of thousands. Some words would be common words and be present across most of the documents, while some words would be rare and present only in very few documents.
- Removing Stop Words
  sklearn.feature_extraction.text provides a list of pre-defined stop words in English, which can be used as a reference to remove the stop words from the dictionary, that is, feature set.
- Creating Count Vectors
  All vectorizer classes take a list of stop words as a parameter and remove the stop words while building the dictionary or feature set. And these words will not appear in the count vectors representing the documents. We will create new count vectors by passing the my_stop_words as stop words list.
    1. Stemming: This removes the differences between inflected forms of a word to reduce each word to its root form. This is done by mostly chopping off the end of words (suffix). For instance, love or loved will be reduced to the root word love. The root form of a word may not even be a real word. For example, awesome and awesomeness will be stemmed to awesom. One problem with stemming is that chopping of words may result in words that are not part of vocabulary
    2. Lemmatization: This takes the morphological analysis of the words into consideration. It uses a language dictionary (i.e., English dictionary) to convert the words to the root word. For example, stemming would fail to differentiate between man and men, while lemmatization can bring these words to its original form man.
- Distribution of Words Across Different Sentiment
  The words which have positive or negative meaning occur across documents of different sentiments.This could give an initial idea of how these words can be good features for predicting the sentiment of documents. For example, let us consider the word awesome.

CHALLENGES OF TEXT ANALYTICS

- | Using n-Grams
- Build the Model Using n-Grams