RV College of Engineering®

Go, change the world®

Academic Year : 2023 – 2024(Odd Sem)

USN:

Department of Artificial Intelligence and Machine Learning

Course Code : 21AI52 **Date : 19.03.2024**

Semester: V Semester **Duration : 90 Min**

Improvement Test

Course title : Artificial Intelligence and Machine Learning

(Common to CS, IS,AI &ML)

****

Sl. No.		M
1	**K means algorithm. -3M**	

Algorithm 8.1 Basic K-means algorithm.

```
Select K points as initial centroids.
repeat
    Form K clusters by assigning each point to its closest centroid.
    Recompute the centroid of each cluster.
until Centroids do not change.
```

Solution:

a) Initial Centroids : C1=A1=(2,10), C2 =A4=(5,8), C3=A7=(1,2)

A1=(2,10), A2=(2,5), A3=(8,4), A4=(5,8), A5=(7,5), A6=(6,4), A7=(1,2), A8=(4,9).

Euclidean distance from point to each centroid -1.5

		C1=(2,10)	C2=(5,8)	C3=(1,2)	
A1	2,10	**0**	3.6	8.06226	C1
A2	2,5	5	4.24264	**3.16228**	C3
A3	8,4	8.48528	**5**	7.28011	C2
A4	5,8	3.60555	**0**	7.2111	C2
A5	7,5	7.07107	**3.60555**	6.7082	C2
A6	6,4	7.2111	**4.123106**	5.38516	C2
A7	1,2	8.06226	7.2111	**0**	C3
A8	4,9	2.23607	**1.414214**	7.61577	C2

End of epoch 1.
new clusters: 1: {A1},
2: {A3, A4, A5, A6, A8},
3: {A2, A7} - 1M

New Centroids: -1M

centers of the new clusters: C1= **(2, 10)**,
C2= ((8+5+7+6+4)/5, (4+8+5+4+9)/5) = **(6, 6),**
C3= ((2+1)/2, (5+2)/2) = **(1.5, 3.5)**

10

Euclidean distance from point to each centroid -1.5

		C1=(2,10)	C2=(6, 6)	C3=(1.5, 3.5)	
A1	2,10	**0**	5.65685	6.5192	C1
A2	2,5	5	4.123106	**1.58114**	C3
A3	8,4	8.48528	**2.828427**	6.5192	C2
A4	5,8	3.60555	**2.23607**	5.70088	C2
A5	7,5	7.07107	**1.414214**	5.70088	C2
A6	6,4	7.2111	**2**	4.52769	C2
A7	1,2	8.06226	6.40312	**1.58114**	C3
A8	4,9	**2.23607**	3.60555	6.04152	C1

End of epoch 2. 1M

new clusters: 1: {A1, A8},
2: {A3, A4, A5, A6, A8},
3: {A2, A7}

New Centroids: 1M

centers of the new clusters: C1= ((2+4)/2, (10+9)/2) = **(3,9.5)**
C2= ((8+5+7+6)/4, (4+8+5+4)/4) = **(6.5, 5.25),**
C3= ((2+1)/2, (5+2)/2) = **(1.5, 3.5)**

2 a. Discuss how following issues are handle in K-means algorithm

Empty Clusters 1.5M

- Empty clusters can be obtained by applying K-means algorithm if no points are allocated to a cluster during the assignment step.
- If this happens, then a strategy is needed to choose a replacement centroid, since otherwise, the squared error will be larger than necessary.
- One approach is to choose the point that is farthest away from any current centroid. If nothing else, this eliminates the point that currently contributes most to the total squared error.
- Another approach is to choose the replacement centroid from the cluster that has the highest SSE. This will typically split the cluster and reduce the overall SSE of the clustering.

Outliers 1.5M

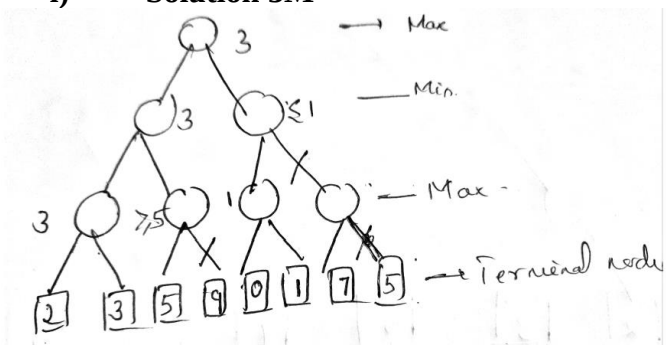
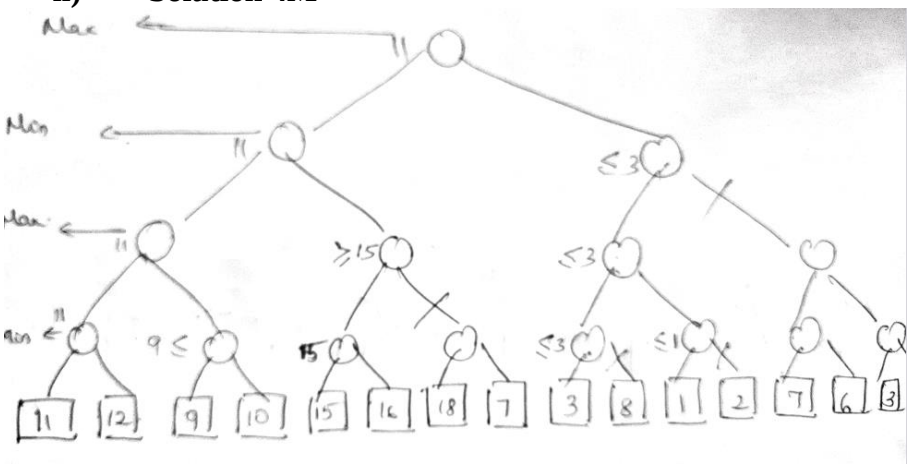
- When the squared error criterion is used, outliers can unduly influence the clusters that are found.
- When outliers are present, the resulting cluster centroids (prototypes) may not be as representative as they otherwise would be and thus, the SSE will be higher as well. Because of this, it is often useful to discover outliers and eliminate them beforehand.
- It is important, however, to appreciate that there are certain clustering applications for which outliers should not be eliminated. When clustering is used for data compression, every point must be clustered, and in some cases, such as financial analysis, apparent outliers, e.g., unusually profitable customers, can be the most interesting points.
- If we use approaches that remove outliers before clustering, we avoid clustering points that will not cluster well.

05

	iii **Reducing the SSE- 2M** • An obvious way to reduce the SSE is to find more clusters, i.e., to use a larger *K*. However, in many cases, we would like to improve the SSE, but don't want to increase the number of clusters. This is often possible because K means typically converges to a local minimum. • Various techniques are used to "fix up" the resulting clusters in order to produce a clustering that has lower SSE. • The strategy is to focus on individual clusters since the total SSE is simply the sum of the SSE contributed by each cluster. • We can change the total SSE by performing various operations on the clusters, such as splitting or merging clusters. One commonly used approach is to use alternate cluster splitting and merging phases. During a splitting phase, clusters are divided while during a merging phase, clusters are combined. Two strategies that decrease the total SSE by increasing the number of clusters are : • Split a cluster • Introduce a new cluster centroid Two strategies that decrease the number of clusters, while trying to minimize the increase in total SSE, are : • Disperse a cluster • Merge two clusters	

b.	**Bisecting K-means algorithm – 3M**
	Initialize the list of clusters to contain the cluster consisting of all points.
	repeat
	  Remove a cluster from the list of clusters.
	  {Perform several "trial" bisections of the chosen cluster.}
	  for $i = 1$ to *number of trials* **do**
	  Bisect the selected cluster using basic K-means.
	  end for
	  Select the two clusters from the bisection with the lowest total SSE.
	  Add these two clusters to the list of clusters.
	until Until the list of clusters contains K clusters.
	Explanation : 2M
	• To obtain *K* clusters, split the set of all points into two clusters, select one of these clusters to split, and so on, until *K* clusters have been produced.
	• There are a number of different ways to choose which cluster to split. We can choose the largest cluster at each step, choose the one with the larges SSE, or use a criterion based on both size and SSE. Different choices result in different clusters.
	• Although the K-means algorithm is guaranteed to find a clustering that represents a local minimum with respect to the SSE, in bisecting K-means we are using the K means algorithm "locally," i.e., to bisect individual clusters. Therefore, the final set of clusters does not represent a clustering that is a local minimum with respect to the total SSE.
		05

3 a	Explain different classification-oriented measures of cluster validity **Definition of each measure 5*1M** Entropy Purity Precision Recall	**05**

	F-measure	
b.	Two unsupervised approaches for assessing cluster validity that are based on the proximity matrix. – **Explanation of each 2.5* 2 = 5M** • Measuring Cluster Validity via Correlation • Judging a Clustering Visually by Its Similarity Matrix	**05**
4	**Explain Alpha-Beta pruning -3** • The problem with minimax search is that the number of game states it has to examine is exponential in the depth of the tree. Unfortunately, we can't eliminate the exponent, but it turns out we can effectively cut it in half. • The trick is that it is possible to compute the correct minimax decision without looking at every node in the game tree. • When alpha–beta pruning applied to a standard minimax tree, it returns the same move as minimax would, but prunes away branches that cannot possibly influence the final decision. • Alpha–beta pruning can be applied to trees of any depth, and it is often possible to prune entire subtrees rather than just leaves. • The general principle is this: consider a node n somewhere in the tree such that Player has a choice of moving to that node. If Player has a better choice m either at the parent node of n or at any choice point further up, then n will never reached in actual play. So once we have found out enough about n to reach this conclusion, we can prune it. Apply the Alpha-Beta Pruning on the given search trees **i) Solution 3M**  **ii) Solution 4M**  	**10**

5	i) Simulated annealing-5M **Algorithm 2.5 M Explanation 2.5 M** **function** SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state **inputs**: *problem*, a problem *schedule*, a mapping from time to "temperature" *current* ← MAKE-NODE(*problem*.INITIAL-STATE) **for** $t = 1$ **to** ∞ **do** $T \leftarrow schedule(t)$ **if** $T = 0$ **then return** *current* *next* ← a randomly selected successor of *current* $\Delta E \leftarrow next.\text{VALUE} - current.\text{VALUE}$ **if** $\Delta E > 0$ **then** *current* ← *next* **else** *current* ← *next* only with probability $e^{\Delta E/T}$ ii) Simulated annealing to combines hill climbing with a random walk in some way that yields both efficiency and completeness. Annealing is the process used to temper or harden metals and glass by heating them to a high temperature and then gradually cooling them, thus allowing the material to reach a lowenergy crystalline state iii) The innermost loop of the simulated-annealing algorithm is quite similar to hill climbing. Instead of picking the *best* move, however, it picks a *random* move. If the move improves the situation, it is always accepted. Otherwise, the algorithm accepts the move with some probability less than 1. The probability decreases exponentially with the "badness" of the move—the amount ΔE by which the evaluation is worsened. iv) The probability also decreases as the "temperature" T goes down: "bad" moves are more likely to be allowed at the start when T is high, and they become more unlikely as T decreases. If the schedule lowers T slowly enough, the algorithm will find a global optimum with probability approaching 1. **Local beam search – 5M** • The local beam search algorithm keeps track of k states rather than just one. • It begins with k randomly generated states. At each step, all the successors of all k states are generated. If any one is a goal, the algorithm halts. Otherwise, it selects the k best successors from the complete list and repeats. • At first sight, a local beam search with k states might seem to be nothing more than running k random restarts in parallel instead of in sequence • In a local beam search, useful information is passed among the parallel search threads. In effect, the states that generate the best successors say to the others, "Come over here, the grass is greener!" • The algorithm quickly abandons unfruitful searches and moves its resources to where the most progress is being made. • In its simplest form, local beam search can suffer from a lack of diversity among the k states—they can quickly become concentrated in a small region of the state space, making the search little more than an expensive version of hill climbing. A variant called stochastic beam search, analogous to stochastic hill climbing, helps alleviate this problem populate the next generation according to its "value" (fitness).	**10**