* Week4 - Clustering
* DBSCAN is density based.
* Bisecting K-means is divisive clustering.
* DBSCAN works well with any shape of data.
* DBSCAN can handle clusters of arbitrary sizes.
* DBSCAN can identify noise while clustering.
* Select all correct sentences that are true regarding k-Means clustering.
* Cluster boundaries are circle if Euclidean distance is used.
* ~~Robust to outliers.~~
* Bisecting k-Means can reduce the initial centroids issue.
* ~~Robust to clusters of different sizes and densities~~
* In cluster validation entropy should be used when class labels are available for the data points
* In k-means clustering, a lower entropy value indicates higher cluster purity.
* The basic idea behind DBSCAN is to cluster data points based on their density and distance from each other.
* The advantage of DBSCAN over k-means clustering is DBSCAN can handle clusters of arbitrary shape.
* Table

  Description automatically generated
* Entropy: entropy (3,7) = -1\*((B2/D2) \*LOG(B2/D2,2)+(C2/D2)\*LOG(C2/D2,2))
  + entropy (5,1) = -1\*((B3/D3) \*LOG(B3/D3,2)+(C3/D3)\*LOG(C3/D3,2))
  + entropy for a = C10\*D2/D4+C11\*D3/D4
  + note entropy of equal classes ie (8,8) will always be = 1. So no need to calc class
* Graphical user interface, application, table, Excel

  Description automatically generated
* Validating clustering accuracy
  + Supervised Purity
  + Ith cluster = 1/# samples in cluster \* max(number of class that appears most in cluster)
  + Total cluster = 1/# samples \* (max number of sample for a particular class in cluster 1 + max number of sample for a particular class in cluster 2….)
* Table

  Description automatically generated with medium confidence
* FORMULAS
* Euclidean=SQRT(($B$2-$B$4)^2+($C$2-$C$4)^2)
* Silhouette= get A, and B. B is the avg of the other points. Itself is not included.
* =(N4-M4)/MAX(M4,N4)
* Week5 – Association rule mining. Pretty straight forward.
* Association does not imply correlation.
* Which are some of the challenges of generating frequent itemset?
  + Generating unnecessary itemset. Generating same itemset more than once.
* Which aspects impact the complexity of rule mining for a dataset?
  + Size of dataset. Min support threshold
* If itemset {a,r,t,y,z} is frequent, what can we say about itemset {r,z}?
  + Frequent. Correct! Based on apriori principal subset of frequent sets are also frequent.
* If itemset {A,b,c,d} is frequent, then which of the transactions in the table are also frequent? All subsets are also frequent. So go through each transaction, and if it is a subset then its good. Obviously, a subset is a itemset that exists in the superset.
* There’s a rule. Confidence is .6 and support is .2. What is the support of the subset? What they are doing here is giving you 2 different parts of the confidence equation, so just solve for the missing piece, in this case it’s the denominator of the confidence calculation.
* FORMULAS:
* Support = itemset occurrences count / total items count.
* confidence ({a, e} -> {d}) = support ({a, e, d}) / support ({a, e})
* Week6 – Deep Learning
* A tree net is an example of a recursive neural network.
* Why deep learning methods were not popular several decades ago when they were invented?
  + Data amount was low. Computation power was low.
* You want to decide on which architecture of neural network to use for processing stock market prices and dynamics. Which would be a better option? (Also, for hourly weather data for a city, predict next 24 hours weather)
  + Recurrent neural network
* If you want to train a neural network to recognize handwritten digits, the network architecture most appropriate would be convolutional neural network.
* Select which models given a varying number of layers and accuracy looks like you just chose highest accuracy…
* Inputs to recommendation system: User profile, user usage history
* In recommendation systems, we do NOT know the true user preference.
* User based cold start problem is how to recommend items to a new user without usage history.
* Restricted *Boltzmann machine* is an example of a belief network type of learning approach.
* The architecture of a neural network better suited to solve the problem of recognizing a cat in a photo is: convolutional neural network.
* Autoencoders are used in application to data dimension reduction.
* The architecture used in Autoencoders is Boltzmann Machines
* Restricted Boltzmann Machines (RBM) is a type of belief network.
* A neural network can be used to extract knowledge for complex data.
* The reason for using deep learning in recommendation systems is to overcome cold start problems.
* The deep learning system most suitable for sequential problems is the recurrent neural network.
* Convolutional neural networks (CNN) are mostly used for image classification systems because: image processing techniques commonly use convolution operations, and therefore CNNs are effective in extracting features from images.
* The purpose of a hidden layer in a restricted Boltzmann machine in a recommendation system is it is the wild guess about the user preferences (based on user ratings for items)
* Recurrent neural networks have self-loops in their architecture (Boltzmann, convolution, recursive do not)
* Cold start problem in a move recommendation system: when a new user has arrived, we do not know which movies to recommend. And when a new movie has arrived, we do not know to whom to recommend the movie.
* In a restricted Boltzmann machine (RBM), we do back propagation learning on the reversed machine to adjust weights for better user preference extraction based on items rating.
* Autoencoders are used for reducing dimension of data.
* The XOR problem led to utilizing nonlinear neural network in form of multi-layer perceptron. Single perceptron has linear boundaries and cannot solve nonlinear problem of XOR. This gap resulted in utilizing multi-layer perceptron to resolve the issue.
* Week7 – Reinforcement Learning
* Which problem cannot be solved by Dynamic programming based on reinforcement learning? Helicopter. The environment would be the sky. So, it is extremely difficult to define the space. And DP you need perfect model env.
* Monte Carlo
  + Don’t need model of environment.
  + Can concentrate on ‘important states.
  + Need to maintain exploration.
  + solution learn from sample episode of simulated experience.
* Temporal difference learning:
  + Does not require the model of environment.
  + Learns from values of successors.
  + It works for continuous tasks.
  + It is usually faster than MC.
* Dynamic programming is the only among monte Carlo and temporal difference learning that require perfect model of the system.
* Reinforcement learning is more general than supervised or unsupervised. It learns from interaction with the environment to achieve its goal.
* Select the correct statement about reinforcement learning compared to the Markov Decision Process: it explores the environment.
* For which problems will you need to use reinforcement learning? Navigating a robot through an unknown territory while avoiding obstructions
* What are the actions in reinforcement learning? Stochastic or deterministic
* Reinforcement learning, we do not have a model of the environment.
* The issue with additive rewards approach is that rewards can go to infinite value.
* The process of creating a Markov chain is defining a set of state and transition probabilities, fixing a policy, and executing it for a long period of time.
* The following are true about the difference between Markov Decision Process and Reinforcement Learning: RL learns from interaction with environment while MDP has a model of the environment. RL does not necessarily have knowledge of transition matrix and rewards, MDP always knows them.
* The goal of Markov Decision Process is to maximize cumulative reward in the long run.
* The reward computation approach for the bellman equation is Discounted Rewards
* A policy in Markov Decision Process is a mapping from the set of states to the set of actions.
* Rewards in Markov Decision Process are used to compare between policies, and to evaluate policy.
* Some of the ways to solve the infinite reward problem is to only evaluate policies until a maximum time horizon is reached, and discount factor.
* Dynamic programming is the reinforcement learning solutions requiring the problem be divided into subproblems.
* We can mitigate the weakness of greedy policy for monte Carlo methods by using E-greedy policy.
* The difference between Vπ and Qπ is: Vπ is the value function for the policy starting from a given state, where Qπ is the value function of the policy for performing an action, starting from a given state.
* The similarity between monte Carlo and temporal difference learning is not needing a perfect model environment.
* You can use DP for solving RL problems when you have a perfect model of environment.
* Methods in dynamic programming: repeat evaluation and improvement until convergence, start with an arbitrary policy, need to have perfect model of environment.
* For dynamic programming, policy improvement means improving π from Vπ.
* For dynamic programming, policy evaluation means computing Vπ from π.
* The main difference between RL and MDP is RL doesn’t assume that you have a model.
* RL overcomes not having a model of environment by learning from the environment and learning sample episodes of simulated experience.
* NOTES: S = State. A=Actions. T=deterministic (or probabilistic) transition matrix. R=rewards
* π=policy. Basically, a mapping between S and A. π is a collection of tuples. (state1, action1).
* Markov property is that it only depends on the current state and action. Doesn’t consider the past.
* Good policy max rewards. Accumulate rewards over time.
* chose initial state.
* perform action.
* use T aka transition matrix to get to current state.
* then repeat 2 for infinite time and compare rewards.
  + This implies that you might have infinite reward, so how do you compare infinite?
  + That’s a problem many different solutions:
    - finite horizon analysis
    - discounting rewards
      * Whatever reward you got in the past, ignore it. How do you do it?
      * typically used very often to use we need to define a value function. V π (s) is the total discounted reward if we start at state s and follow policy π
      * calculate value function.
        + Bellman equation is the recursive way.
    - expected reward.
* Overall problem of RL much more general than unsupervised or supervised.
* Agent performs action to explore environment, each action has rewards and results in new state.
* Does not assume planned knowledge.
* Transitions and rewards are not available. Big problem. Change starting condition with experience.
* V π(s) = expected return when starting in S and following π
* Q π(s,a)= expected return when starting in S, performing A, and following π.
* Good policy max rewards. Accumulate rewards over time.