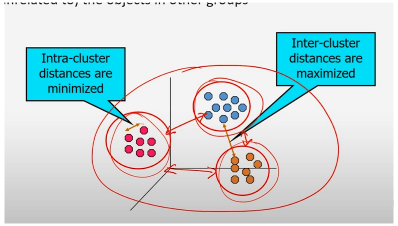
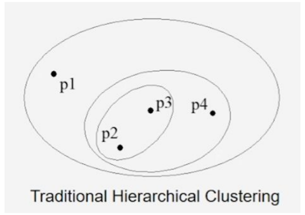
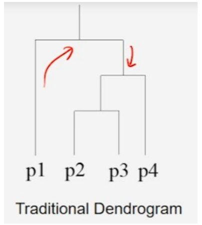
Week 4: Cluster Analysis

Cluster Analysis


Intra-cluster distances are minimized
Inter-cluster distances are maximized



Traditional Hierarchical Clustering



Traditional Dendrogram

- Finding object groupings to one another to identify similarity and difference, no label, unsupervised, clustering not classification
- Use distance metric so that intra-cluster distances are minimized and inter-cluster distances are maximized, can be ambiguous
- Definition is imprecise and depends on nature of data
-
- Types of clustering
- A clustering is a set of clusters, important distinction between hierarchical and partitional sets
- Partitional Clustering
    - A division data objects into non-overlapping subsets, each data object is one subset
- Hierarchical Clustering
    - Set of nested clusters organized as a tree
    - Cluster are permitted to have sub-clusters
    - Each node(cluster) in the tee is the union of its children (sub-clusters) and the root is the cluster containing all objs.
    - Can be represented as a dendogram, tree that lists clusters

Distinctions

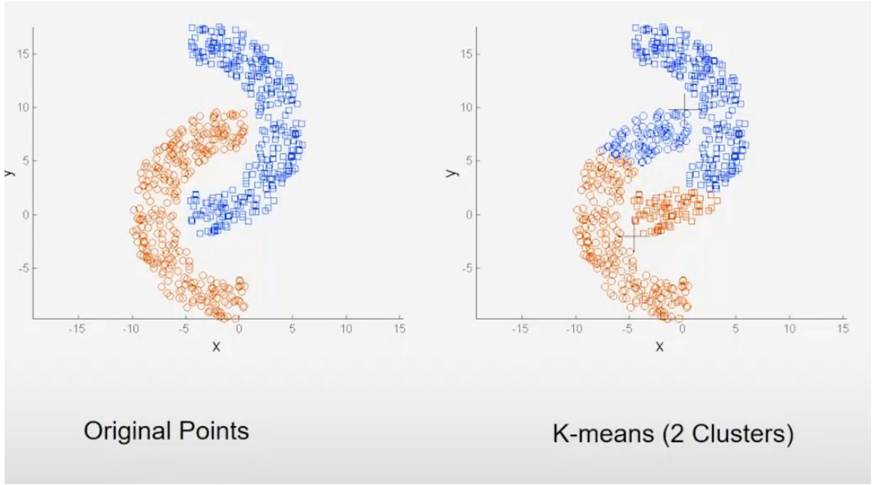
- Exclusive vs. non exclusive
    - Non exclusive, points belong to multiple cluster, person at university is enrolled as student & employee.
    - Can represent multiple classes or border points
- Fuzzy vs. Non-fuzzy
    - Fuzzy clustering, point belongs to every cluster with some weight 0-1, weights sum to 1
    - Can be converted to exclusive clustering by assigning each object to the cluster where membership weight is highest
- Partial vs. Complete
    - Cluster data and leave out sample such as noise and outliers
- Heterogenous vs. Homogenous
    - Cluster of different sizes, shapes and densities

Types of Clusters

- Well separated
  - Set of points such that any point in a cluster is closer to every other point in the cluster than to any point not in the cluster
  - Want clusters far from each other
  - Can be any shape
- Center Based
  - Set of objects such that an object in a cluster is closer to the center of another cluster
  - Center is a centroid, avg of all points in the cluster or a medium, most representative point
- Contiguous Cluster (Nearest Neighbor)
  - Clusters close but points define the edges for the most point
  - Two objects are connected only if they are within a specified distance of each other
  - Has trouble with noisy data, where a small bridge of points can merge 2 clusters
- Density Based
  - Cluster is a dense region of points, separated by low density regions, regions defined by high density
  - Used when clusters are irregular or intertwined, or when noise and outliers are present, cluster not merge by noise
- Objective Function
  - Put points through an objective function to find max or min
  - Enumerate all ways of dividing points int oclusters and evaluate
  - Map clustering problem to different domain and solve problems
  - Proximity matrix defined a weighted graph, where the nodes are points being clustered, and weighted edges represent proximities between points
  - Clustering is equivalent to breaking the graph into connected components, one for each cluster
  - Want to minimize the edge weight between clusters and maximize the edge weight within clusters

K-Means

- Centroid based, PArtitional Clustering (No overlap)
- Each point assigned to cluster with closest centroid
- Num cluckster k must be specified
    - Select k points as centroids
    - Repeat
        - Form K clusters by assigning all points to closest centroid
        - Recompute the centroid of each cluster
    - Until centroids dont change
- Initial centroids cohen randomly, mean of points
- Closeness is measured by euclidean, cosine sim, correlation
- K-means will converge for similarity measures
- Stopping condition is changed to few points change clusters
- Complexity is O(n*K*I*d) n = num points, K = clusters, d = num attributes, i = num iterations
- Initial points matter most (Initial Condition)
- When the K-Means algorithm has reached the local or global minima, it will not alter the assignment of samples to clusters for two successive iterations.
    - Chance is lower when k is large, sometimes will adjust automatically
- Strengths:
    - Simple and can be used for many data types, efficient, bisecting less susceptible to initialization problems
- Weaknesses
    - Cannot handle non-globular clusters or clusters with different sizes, and trouble clustering data with outliners
    - Data must be homogenous
    - Non circular/glob shapes does not classify well (image)



Solving Initial Centroid Problem

- Multiple runs
- Sample using hierarchical clustering to find initial centroid
  - K cluster are extracted from hierarchical clustering and centroid of these clusters are used as initial centroids, only works if sample is small as hierarchical clustering is expensive and k is small
  - Select more than k initial centroids and select among these initials select spread of centroids
  - Bisecting K-means
  - Update centers incrementally
    - Update centroids after each assignment (incremental approach)
    - Each assignment updates zero or 2 centroids (0 if same cluster, 2 if different)
    - More expensive
    - Introduces order dependency, clusters produced may depend on the order in which the points are processed.

Bisecting K-Mean

- Take all data points and divide them into 2 clusters
    - Keep going until k has been produced
- Can choose largest cluster at each step, largest sse, or own criterion on size and SSE resulting in different clusters
- Has less trouble with initialization because performs several trial bisections and takes the one with the lowest sse
- Clustering results are refined using centroids from bisecting k-means as initial centrods for basic k-mens algorithm.

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

Bisecting K Means

- Evaluate K-means cluster
  - Sum of squared error
  - For each point, the error is the distance to the nearest cluster, we then square and sum
  - X i s a data point in cluster ci and mi is the representative point for cluster ci
  - Ci corresponds to mean of the cluster
  - Choose smallest error
    - Can reduce SSE by increasing K
    - K-Means algorithm is guaranteed to find a clustering that represents a local min with respect to SSE

Knowledge Check

- Evaluate K-means cluster
  - Sum of squared error
  - For each point, the error is the distance to the nearest cluster, we then square and sum
  - X i s a data point in cluster ci and mi is the representative point for cluster ci
  - Ci corresponds to mean of the cluster
  - Choose smallest error
    - Can reduce SSE by increasing K
    - K-Means algorithm is guaranteed to find a clustering that represents a local min with respect to SSE

Hierarchical Clustering

•Agglermaritive: Start with the ppoints as invidivdual clusters and at each step merge the closest pair of clusters

Divisive: Start with one all inclusive cluster and at each step, split a lucster until only singleton clusters of individual points remain

•Often displayed graphically using tree like diagram

•Produces set of nested clusters records sequences of merges or splits

Can cut dendogram to certain amount of clusters

Strengths: Doesnt assume k clusters, corresponding classes such as sumarrizes whole animal kingdom

Agglomerative Clustering

Compute proximity - Can be distance

Let each data point be a cluster

Repeat

    Merge the two closest clusters

    Update the proximity matrix

Until only a single cluster remains

Key operation is proximity of two clusters

Agglomerative Clustering Algorithm

How to find proximity between 2 clusters

Min: Defines a cluster proximity as the proximity between the closest two points that are in different clusters

Max: Farthest two points in different clusters

Group Average: of all distances between point distances of 2 clusters

Distance between centroids

Other methods: Wards methods uses squared error

Max and Min metrics used to determine the distance between a point and a cluster and to determine distances between clusters

Performance Analysis of Clustering

Strength of min: Good performance when cluster sizes are different, can handle non elliptical shapers

Weakness: Susceptible to noise

Strength of max: Not sensitive to noise and outliers

Weakness: If big clusters and small cluster, max will break down clusters

Strength of group average: Compromise between single and complete links, less susceptible to noise

Weakness: Biased towards globular clusters, convex shapes to clusters

Time and space: O(N^2) for proximity, n is # of points

O(N^3) - There are n steps and at each step the size must be updated and searched, time can be reduced to O(N^2log(N) time using avandaced data structures like a sorted list or heap

Density Based Clustering

DBScan

Locates regions of high density separated from one another by low density

Center based approach

Density of a point is estimated by counting # of points in a specific radius

Core points - Interior of a density based cluster, epsilon and min points defined. Must be min points in specified area

Border points - Not core, falls in neighborhood of core point, does not have min points in radius

Noise point neither a core point nor a border point, outside all neighborhoods, has les than min points and not in the neighborhood of a core point

Algo

1. Label all points as core, border, or noise

2. Eliminate noise

3. Put an edge between all core points that within eps of each other

Make each group of connected core points into a separate cluster

Assign each border point to one of the clusters of its associated core points.

Time complexity of dbscan is O(m*time to find points in the epsilon neighborhood) where m is the number of points

Worst case Oo(m^2)

In low dim space dat structures like kd-trees allow efficient retrieval of all points within a given distance of a specified point and the time complexity can be reduced to O(m log m)

Space requirement of dbscan is O(m) just need to keep the data of each poin

Steps

Look at behavior of the distance from a point to its kth nearest neighbor

For points that belong to some cluster, the value of k-dist wil be small if k is not largr than the cluster size

For points that not in a cluster such as noise points the k-dist will be relatively large

The value of epsilon determined in theis way depends on k but does not change drastically as k changes

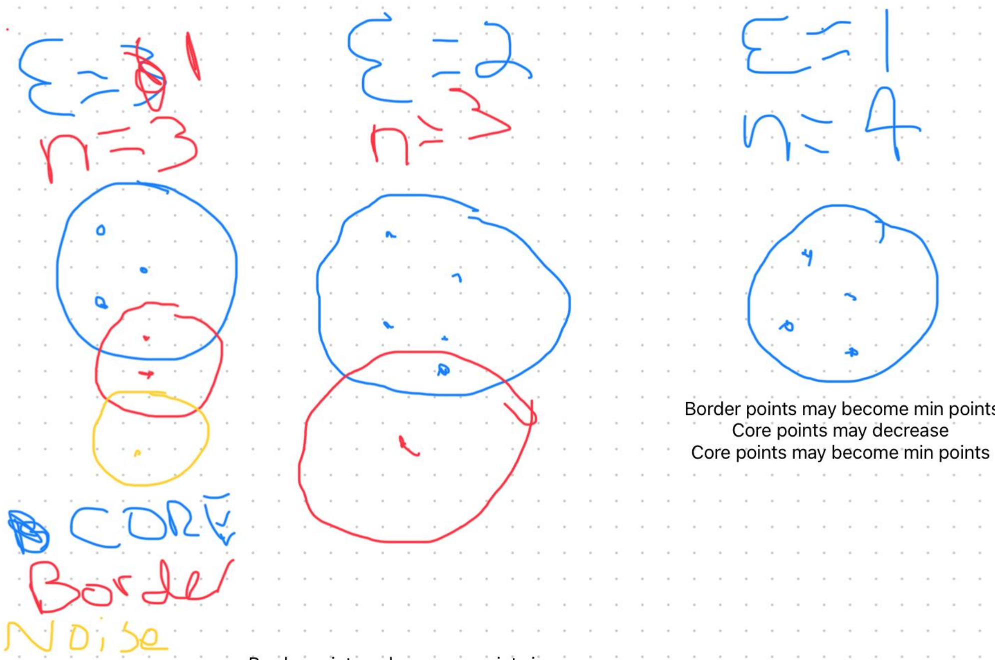
Strengths: uses a density based definition, relatively resistant to noise and can handle clusters of arbitrary shapes

Weaknesses - Has trouble when the clusters have widely varying densities

Has trouble with high dim data, density is more difficult to define

Cam be expensive when the computation of nearest neighbors requirescomputing all pairwise proximities, as is usually the case for high dim datat

DBSCAN Example



Border points may become min points
Core points may decrease
Core points may become min points

Border points reduce, core points increase

Min is noise\

For right

Cluster Validity

For supervised classification there are metrics for model eval.

For cluster analysis we dont have that ground truth

Why do cluster analysis? Avoid finding patterns in noise, compare clustering algorithms, compare 2 sets of clusters, compare 2 clusters

Aspects of Cluster Validation

1. Determine clustering tendency of a set of data, maintain data characteristics
2. Determining number of clusters
3. Evaluating how well the results fit data without reference to external information, do the clusters fit the data and does it make sense?
4. Comparing the results of a cluster analysis to externally known results, can take the clustering results and match them with external labels, maintains properties of data
5. Comparing the results of 2 different sets of cluster analyses to determine which is better, if given labels can compare clustering algorithms and see which model has highest accuracy

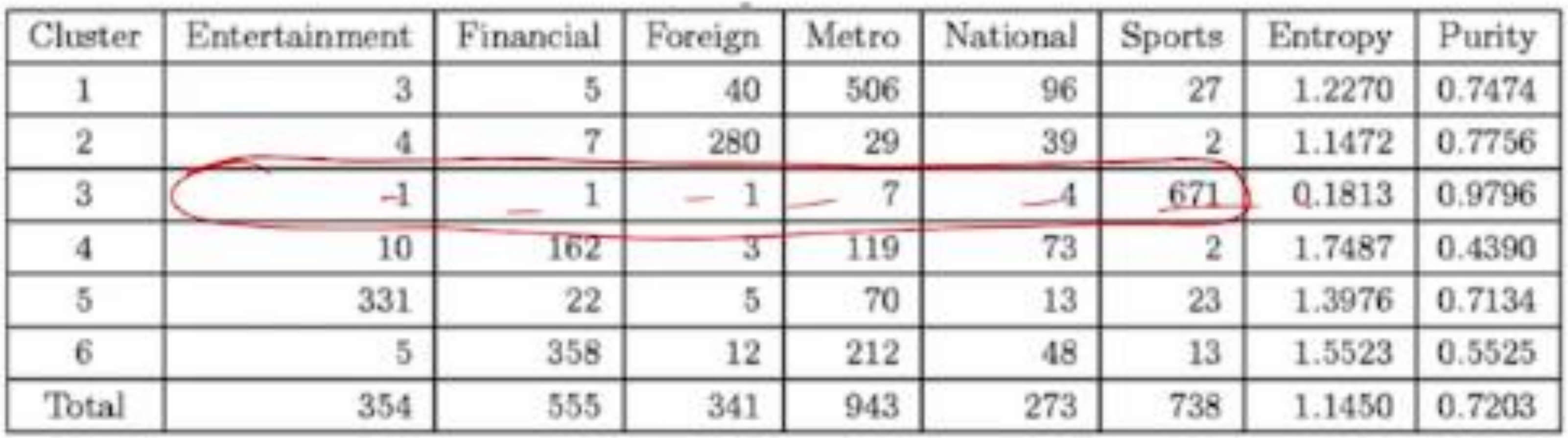
Measures of Cluster Validity

Unsupervised -Measures the goodness of a clustering structure without respect to external info. Two classes cluster cohesion and cluster separation. Often called internal indices. No class levels.

Supervised - Measures when a cluster structure matches some external structure titled external indices. Have class levels.

Relative - Compares different clustering or clusters. A supervised or unsupervised measure that is used for the purpose of comparison. Not a separate type of measure, but are instead a specific use of such measures. Compares clustering techniques.

External Measures of Cluster Validity: Entropy and Purity



Cluster	Entertainment	Financial	Foreign	Metro	National	Sports	Entropy	Purity
1	3	5	40	506	96	27	1.2270	0.7474
2	4	7	280	29	39	2	1.1472	0.7756
3	1	1	1	7	4	671	0.1813	0.9796
4	10	162	3	119	73	2	1.7487	0.4390
5	331	22	5	70	13	23	1.3976	0.7134
6	5	358	12	212	48	13	1.5523	0.5525
Total	354	555	341	943	273	738	1.1450	0.7203

Unsupervised Cluster Validity

No external information

SSE - Measures intra cluster, as clusters increase, SEE decreases, does not capture inter cluster distances, SSE zero when all data points are their own cluster

Cohesion and Separation

Cohesion: How closely related are objects in a cluster, sum of the weight of all links in cluster

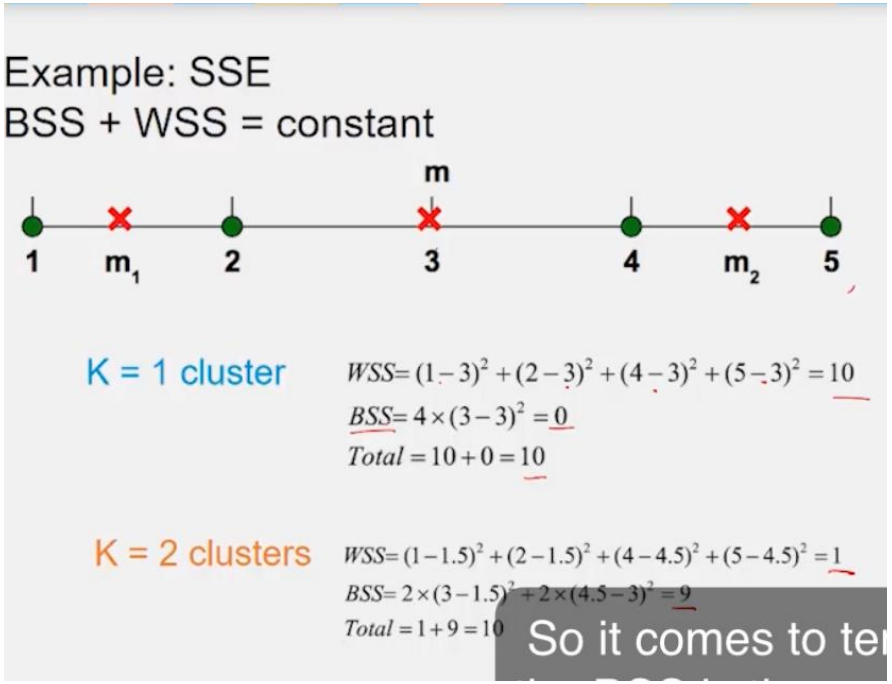
WSS - Std of given cluster, minimize

Cluster Separation: Measures how well-separated a cluster is from other clusters, sum of weights between nodes in the cluster and nodes outside the cluster

BSS - Overall mean of the data, maximize overall mean means well separated data

Relation: WSS and BSS is a constant and is equal to the total sum of squares. TSS is computed as the total sum of squares of the distance to each point to the overall mean of the data, Minimizing WSS is equivalent to maximizing BSS

Example



Example: SSE

BSS + WSS = constant

K = 1 cluster

$$WSS = (1-3)^2 + (2-3)^2 + (4-3)^2 + (5-3)^2 = \underline{10}$$

$$\underline{BSS} = 4 \times (3-3)^2 = \underline{0}$$

$$Total = 10 + 0 = \underline{10}$$

K = 2 clusters

$$WSS = (1-1.5)^2 + (2-1.5)^2 + (4-4.5)^2 + (5-4.5)^2 = \underline{1}$$

$$BSS = 2 \times (3-1.5)^2 + 2 \times (4.5-3)^2 = \underline{9}$$

$$Total = 1 + 9 = 10$$

So it comes to te

Silhouette Coefficient

Calculate avg distance to all other objs in cluster (a)

Calculate its avg distance to all objs in a cluster not containing the obj. Find the minimum of these values with respect to all clusters (b)

Si = (bi - a)/max(ai,bi)

Correlation Validity

Similarity matrix for a dataset, cluster labels from a cluster analysis, goodness of clustering can be by the correlation between the actual similarity matrix and the ideal version of the similarity matrix based on cluster labels.

Ideal Cluster is one - Whose points have a similarity of 1 to all pts in the cluster, and a similarity of 0 to all points in other clusters

Created by - Matrix that has one row and one column for each data point and assigning a 2 to an entry if the associated pair of points belongs to the same cluster

Week 5: Association Rule Mining

Extract if buying milk, they are more likely to buy bread

Association rules should not have practical explanations (i.e. milk and diapers, must have a baby at home, only care about data)

Antecedent (Notion if you buy) -> Consequent (You'll also buy)

Itemset - A collection of one or more items. Kitemset = itemset of k items

Support Count - Frequency of occurrence of an itemset

Support - Fraction of transactions that contain an itemset i.e 2(matches)/5(total)

Frequent Itemset - An itemset whose support is greater than or equal to a minsup threshold

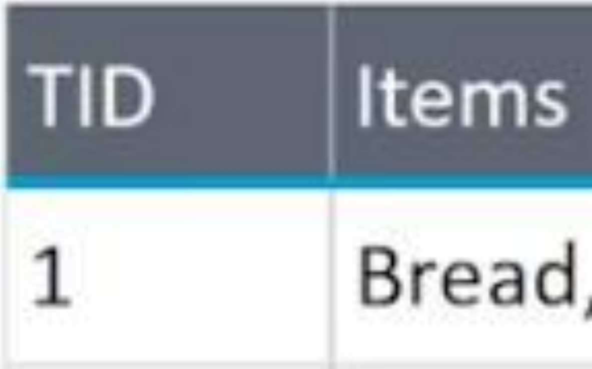
Association Rule Implication expression X-> Y, where X and Y are itemsets (Milk, Diaper) -> (Beer)

$$Support, s(X \to Y) = \frac{\sigma(X \bigcup Y)}{N}$$

$$Confidence, c(X \to Y) = \frac{\sigma(X \bigcup Y)}{\sigma(X)}$$

Example:

$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \text{Beer}$$



TID	Items
1	Bread,

Confidence and support

Support - May occur by change, can occur simply by chance

Confidence - Measures the reliability of the inference made by a rule

Need both to understand association rule mining task

Have to find rules having support >= minsup threshold and confidence >= minconf threshold

Brute force: List all possible association rules, Computer support and confidence for each rule, prune rules that fail minsup and minconf thresholds = Computationally prohibitive. Total number of rules extracted from a dataset that contains d items is R = 3^d - 2^(d+1) + 1, itemset * rules

High confidence, low support -> Not common

Same itemset, same support, different confidence

Want moderate support and high confidence

Confidence A -> {} = 1

Intro to Apriori Principle

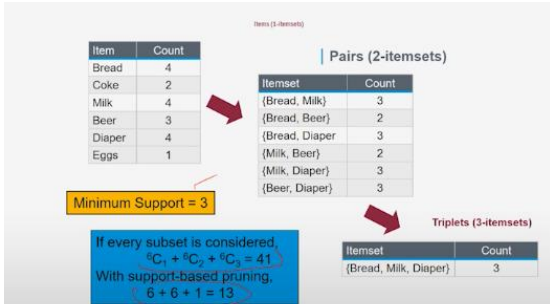
Reduce complexity of brute force search

If an item set is frequent, then all of its subsets must also be frequent X & Y >= minsup. Support(X) >= Support(Y)

Calculate bottom support, all items in top connecting are subsets and support doesn't need to be calculated

Bottom up, calculate infrequent, all supersets are also infrequent, prune

Support Based Pruning



Items (1-itemsets)

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Pairs (2-itemsets)

Itemset	Count
{Bread, Milk}	3
{Bread, Beer}	2
{Bread, Diaper	3
{Milk, Beer}	2
{Milk, Diaper}	3
{Beer, Diaper}	3

Minimum Support = 3

Triplets (3-itemsets)

Itemset	Count
{Bread, Milk, Diaper}	3

If every subset is considered,
$^6C_1 + {^6C_2} + {^6C_3} = 41$
With support-based pruning,
$6 + 6 + 1 = 13$

Apriori Algorithm

Method:

Let k=1

Generate frequent itemsets of length 1

Use the Apriori algorithm to prune part of the lattice that is infrequent

Generate k+1 itemsets from k itemsets

Repeat until no new frequent itemsets are identified

Needs to make an additional pass over dataset after selecting candidate items sets

Avoid generating as manny candidates as possible

Generates new candidate k-itemsets based on frequent {k-1} itemsets found in previous iteration

Apriori Algorithm

Single pass over data - Eliminates all candidate itemsets whose support is less than minsup

Generates new candidate k-itemsets using frequent k-1 - Algorithm terminates when there are no new frequent itemsets generate

Algorithm needs to make an additional pass over dataset

Candidate Generation - Generate new candidate k-itemset based on frequent k-1 itemsets found in previous iteration

Candidate Pruning- Eliminates some of candidate k-itemsets using support based pruning

Avoid generating unnecessary candidates, Candidate set is complete, should not generate same candidate itemset more than once

Subsets of itemsets that are frequent are freuqnt

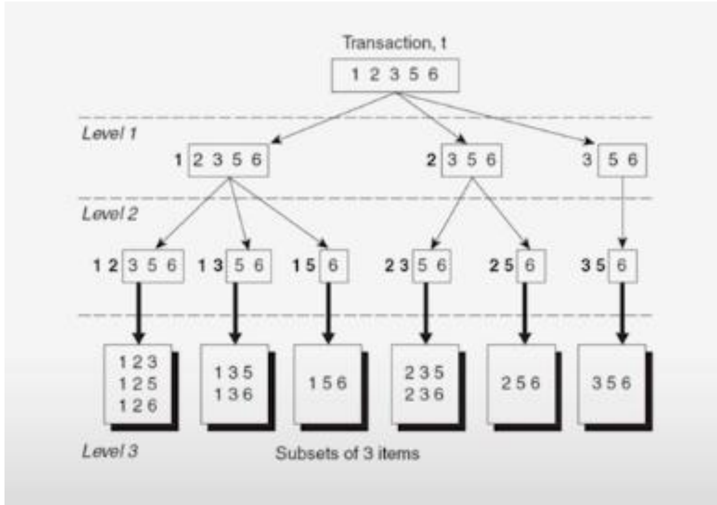
Supersets of itemsets that are not frequent are not frequent

Support Counting

Frequency of occurrence for every candidate items that survives candidate pruning step

Compares each transaction against ever candidate itemset

Enumerate the itemsets combined in each transaction



Factors affecting complexity in Apriori Algorithm

Choice of min supp

Dimensionality (number of items) of data set

Size of db

Average transaction width

Rule generation: Frequency of occurrence for every candidate itemset that survives pruning step, compare each transaction against every candidate itemset

X = {1,2,3} = 3^2 = 9 combinations make {1,2} -> {3} etc. , goes down to 6 rules

Confidence of sigma{1,2,3}/sigma{1,2}

Anti-monotone property of support ensures that {1,2} must be frequent too

Challenges include finding initial itemsets to start with and itemset ordering

Reduce number of rules

Confidence based pruning - Confidence does not have pruning theorem. Whatever is in x dont have it in y

X = {1,2} Y = {1,2,3} Y-X={3}

Rule X->Y-X does not satisfy confidence threshold

X’ -> Y-X’ where X’ is a subset of X must not satisfy the confidence threshold as well

{1,2} -> {3} - Does not satisfy confidence threshold

{1} -> {2,3} -Does also not satisfy confidence threshold

Proof

X’->Y-X’ and X->Y-X where X’ is a subset of X

Confidence of the rules are sigma(Y)/sigma(X’) and sigma(Y)/sigma(X) respectively

X’ is a subset of X, sigma(X’)>=sigma(X)

(Sigma(Y)/Sigma(X’)) < (Sigma(Y)/Sigma(X))

If confidence threshold not met,dont continue

Only generate rules that satisfy confidence threshold

Apriori Algorithm

Rule Generation

Level Wise Approach

High confidence rules that have only one item in the rule consequent are extracted

These rules are candidates

{a,c,d} -> {b} and {a,b,d} -> {c}

{a,d} -> {b,c}

Optimize association rule mining

Frequent itemset generation - Generate all itemsets whose support >= minsup

Rule Generation - High confidence rules from each frequent itemset

Given d items, there are 2^d possible candidate itemsets (including the null set)

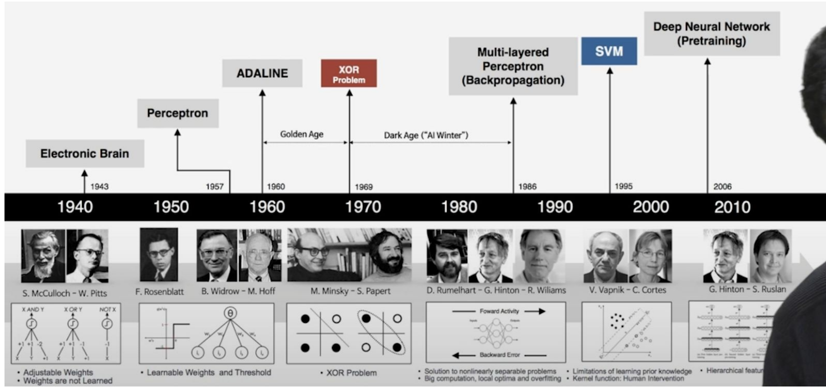
First search rules with highest support, then find minimum confidence

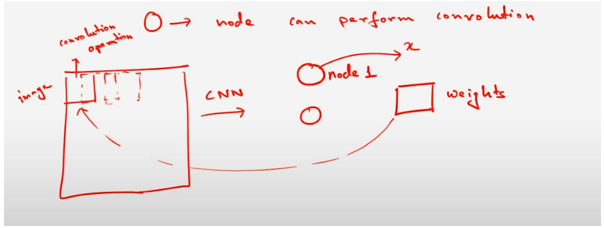
Can reduce number of candidate itemsets - Apriori principle

Reduce number of comparisons Use more advanced data structures

Week 6: Deep Learning Intro

- GPUs changed deep learning
- Deep learning helps overcome cold start problems
- XOR problem led to utilizing non-linear neural networks in form of multi-layer perceptron.Single perceptron has linear boundaries and can not solve nonlinear problem of XOR. This gap resulted in utilizing multi-layer perceptron to resolve the issue.
- radio_button_unchecked
- Curse of dimensionality
- radio_button_unchecked
- Noise and outliers.
-
- Deep Neural Network
  - ANN -> Feed Forward
  - Greater than 10 hidden layers, each with 50-100 nodes
- Recurrent Neural Network
  - Hidden layer with inputs and outputs
  - Output fed back to input with a weight to then unroll to go on infinitely as hidden layers
  - Helps solve sequential problems to extract knowledge from time series
  - Has self loops
- Recursive Neural Network
  - Tree Nets
  - Sets of hidden nodes that feed forward into each other which then each node at each step goes to another hidden layer
- Convolution Neural Network
  - Knowledge from images
  - 3D Neural Network
  - Each node performs convolution operation
  - X is a number, hidden layer is series of convolutions
  - Feature extraction is a convolution with filters
- Popular Tools
  - Caffe
  - Tensor Flow
  - Keras
  - Theano

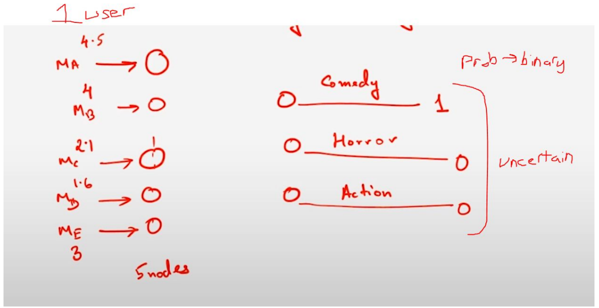




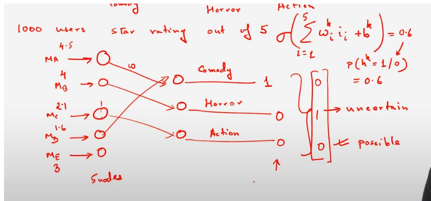
Recommendation System

- First entry point is user profile (age, gender, preferences)
- Usage history (What you watched, length of watching, rating, season) -> Gain preference knowledge w/o asking and changes dynamically
- Recommendation System Problems
    - Cold Start Problem
        - Item Based - To whom do i recommend new items, no watch history, select salient features and match with other items to find a set similar to new item. Find users that like the matched set the most
        - User Based - No usage history, match user profile with other users

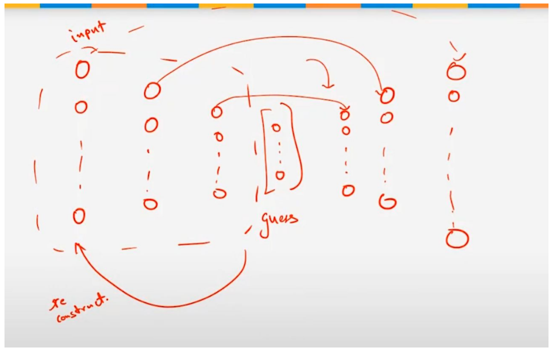
Restricted Boltzmann



- Bayesian Network
  - Naive Bayes
    - Given a class, each attribute is conditionally independent of one another
  - Multiple children nodes that each have subnodes that are directional to the subnodes (Complex Conditional Dependence Architecture)
- Deep Belief Network
  - Belief Network - Nodes are observable facts, roots are potential reasons/beliefs (may or may not be observable)
  - Has several hidden nodes (hidden beliefs, stochastic output(random) from some activation function 0-1) to then set as a 1 or 0 if probability is greater than a threshold (i.e. .8)
  - Beliefs are associated with each other
  - Difficult to learn a belief network due to hidden layers and number of hidden layers
  - Simplify using restricted boltzmann
- Restricted Boltzmann
  - ANN with input layer (observable) and hidden belief layer (not observable) to produce a bipartite graph
  - Depends on educated guess with information extract gain information such as usage history
  - Used in recommendation systems
  - Output is wild guess about the preferences
  - Hidden layer then becomes input layer to reverse it, same calcs in reverse
  - Must do back propagation to reduce error in reverse calculations, must update weights to get true value
  - Reiterate to adjust output and get correct values in the reversed model (guess closer to reality)
  - Can now correctly guess based on new model with updated weights



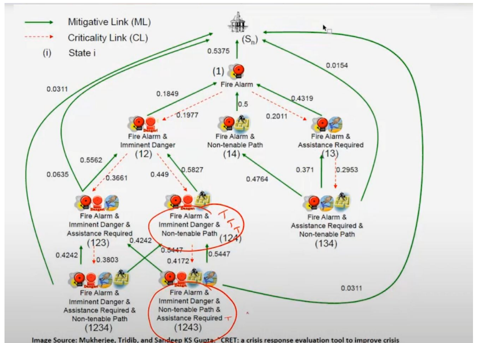
Autoencoder - PCA



- Stacked RBMs
  - Input layer (Reality) -> Hidden (Wild Guess) in Boltzmann
  - Then make another hidden input -> Hidden Layer and so on and so on
- Autoencoder
  - Another type of principal component of deep neural networks
  - Input to smaller wild guesses, then reconstruct the input
  - Upon reconstruct just runs the input layer in reverse
  - Treat as feed forward network and input is the same as output
  - Reduces dim to small set of guesses so that guesses are cognizant, then uses those guesses to reestimate the data to try and produce the initial input layer
  - Then removes input layer and just use the guess and reconstruction since it will be in a lower dimension initially
  - Classification, clustering

Week 7: Markov Decision Processes

- Model complex probabilistic approaches
- MDP - Mathematical construct (S,A,T,R)
    - S -> Set of states, current status of system, valuation of variables V-> Set of variables
    - A -> Set of actions taken
    - T ->Transition Matrix, S x A, Deterministic transition matrix
        - Goes to probability value
    - R -> Set of rewards: S x A -> Real Set R
        - R(s1, a1) provides Real Number
- Evaluate Policies
    - Need definition of policy
        - Pi -> Policy
        - Pi -> A mapping from S->A, a collection of tuples, state 1 means action 1, state 2 has action 2, and so on. Given a state what action should you take
        - How to evaluate -> Calculate the reward, maximize it, and accumulate the reward over time
        - Execute a policy: Choose initial state, perform action pi(si), Use transition matrix T to go to the current state si, then repeat perform action
        - Compare two policies by seeing which has best reward
    - Next state only depends on current state and next action, hence why its called Markov, Markovian Property



Mitigative Link (ML)
Criticality Link (CL)
(i) State i
(S_n)
0.5375
0.0154
0.0311
(1) Fire Alarm
0.1849
0.4319
0.5
0.1977
0.2011
Fire Alarm & Imminent Danger (12)
Fire Alarm & Non-tenable Path (14)
Fire Alarm & Assistance Required (13)
0.5562
0.0635
0.3661
0.449
0.5827
0.4764
0.371
0.2953
Fire Alarm & Imminent Danger & Assistance Required (123)
Fire Alarm & Imminent Danger & Non-tenable Path (124)
Fire Alarm & Assistance Required & Non-tenable Path (134)
0.4242
0.4242
0.3803
0.5447
0.4172
0.5447
0.0311
Fire Alarm & Imminent Danger & Assistance Required & Non-tenable Path (1234)
Fire Alarm & Imminent Danger & Non-tenable Path & Assistance Required (1243)

Image Source: Mukherjee, Tridib, and Sandeep KS Gupta. "CRET: a crisis response evaluation tool to improve crisis preparednesss." *Proceedings of the International Conference on Technologies for Homeland Security*. 2009.

S in Red

A - Give assistance to people

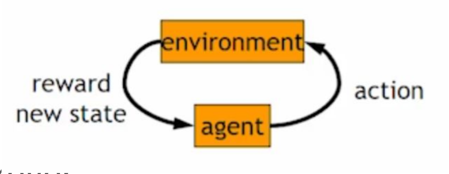
T - Probability of occurrence

R - Reward to encourage or discourage actions

Week 7: Markov Decision Process Cont.

- Execute a policy infinite time and compare cumulative rewards, can have infinite reward making it difficult to compare
  - 1. Finite horizon analysis -> Just stop after certain time so infinity doesn't get hit
  - 2. Discounting reward to make it harder to get to infinity, ignore past rewards, used most often
  - 3. Expected reward instead of cumulative
- Discounting
  - Value function -> Vpi(s) => total discounted reward if we start from state s and follow policy pi
  - Calculate value function with bellman equations -> Recursive way of evaluative value function
  - Vpi(s) = R(s,pi(s)) + Sum(T(s,pi(s),s’)) add discounting factor multiplier from 0 to 1
  - Find pi s/t vpi(s) is maximized
  - Fix a policy execute it for a longer time, Markov Chain

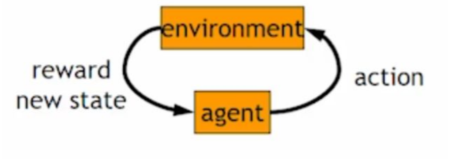
Reinforcement Learning



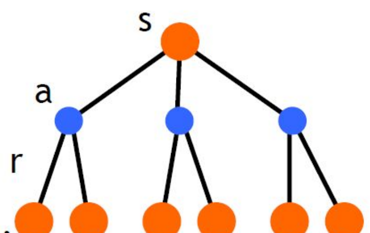
- Reinforcement Learning
  - More general than supervised/unsupervised learning
  - Learn from interaction w/ environment to achieve a goal
  - Does Not assume plan or model of environment
  - Terminal state, any other action will not give a reward, will be in state forever
  - Strategy, each step costs -.04 reward
  - Stochastic or deterministic
- MDP vs. Reinforcement Learning
  - Transitions and rewards usually not available
  - How to change the policy based on experience
  - How to explore the environment
- Computing return from rewards
  - Episodic tasks
    - Game over after N steps
    - Optimal policy depends on N; harder to analyze
  - Additive Rewards
    - Infinite value for continuing tasks
  - Discounted rewards
    - Values bounded if rewards bounded (add a weight multiplier)

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} P^{a}_{ss'} \left[r^{a}_{ss'} + \gamma V^{\pi}(s') \right] = \sum_{a} \pi(s,a) Q^{\pi}(s,a)$$

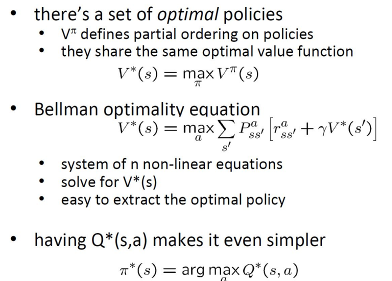
Value Functions



- State Value function: V^pi (s)
  - Expected return when starting s and following pi
- State-action value function: Q^pi (s,a)
  - Expected return when starting in s, performing a, and following pi
- Vπ (state-value function) represents the expected return starting from a particular state under a given policy π. It calculates the value of being in a particular state and following the policy thereafter.
- Qπ (action-value function) represents the expected return starting from a particular state, taking a particular action, and then following the policy π thereafter. It calculates the value of taking a particular action in a particular state and following the policy thereafter.
  -
- Useful for finding optimal policy
  - Estimate from experience, pick best state action value function
- Bellman Equation



- Optimal Value Functions



- there's a set of *optimal* policies
  - V^{π} defines partial ordering on policies
  - they share the same optimal value function

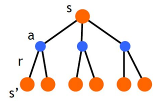
$$V^{*}(s) = \max_{\pi} V^{\pi}(s)$$

- Bellman optimality equation

$$V^{*}(s) = \max_{a} \sum_{s'} P^{a}_{ss'} \left[r^{a}_{ss'} + \gamma V^{*}(s') \right]$$

  - system of n non-linear equations
  - solve for V*(s)
  - easy to extract the optimal policy
- having Q*(s,a) makes it even simpler

$$\pi^{*}(s) = \arg\max_{a} Q^{*}(s,a)$$

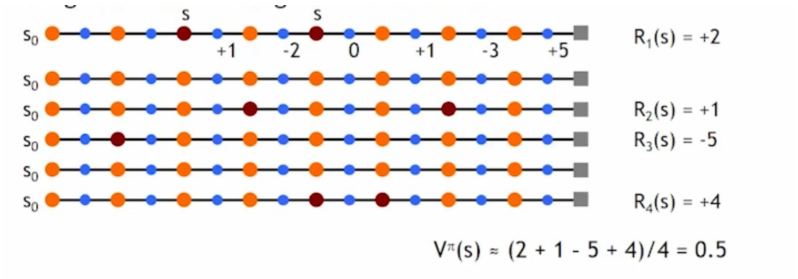


Dynamic Programming to Solve RL

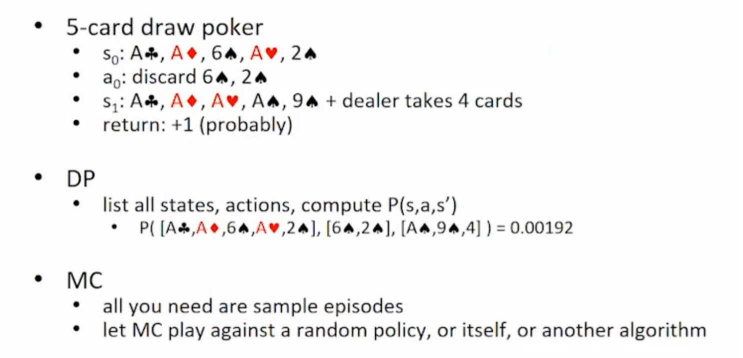
- Need: Divide into subproblems and have optimality of overall problem
- Main Idea
    - Use value functions to structure the search for good policies (policy exploration), need perfect model of env
- 2 Main components (Continuous)
    - Policy eval: Compute V^pi from pi
    - Policy improvement: Improve pi based on V^pi
    - Start with arbitrary policy
    - Repeat evaluation/improvement until convergence
    - Explore all possible options
    - For dynamic programming, Policy Evaluation refers to the process of computing the state-value function (Vπ) for a given policy (π)
    - Policy improve is improve pi from Vpi
- Need complete model of the environment and rewards (State space, action space, transition model)
- Need to start with an arbitrary policy and repeat evaluation and improvement unti lconvergence
- Can use DP to solve
    - Robot in a room - Know end result and environment
    - Backgammon - Know end result and environment
    - Helicopter - No, do not know environment

Monte Carlo Policy Evaluation

- Difference between Dynamic Programming is you don't need to know environment
  - Just need experience or simulated experience
- Similar to Dynamic Programming in policy evaluation and improvement
- Policy Evaluation
  - Want to estimate V^pi(s), explore environment N times and average reward value
  - Estimate as average of observed returns in state s
  - first -visit MC, average returns following the first visit to state s
- Averaging sample returns defined only for episodic tasks



Monte Carlo Policy Control

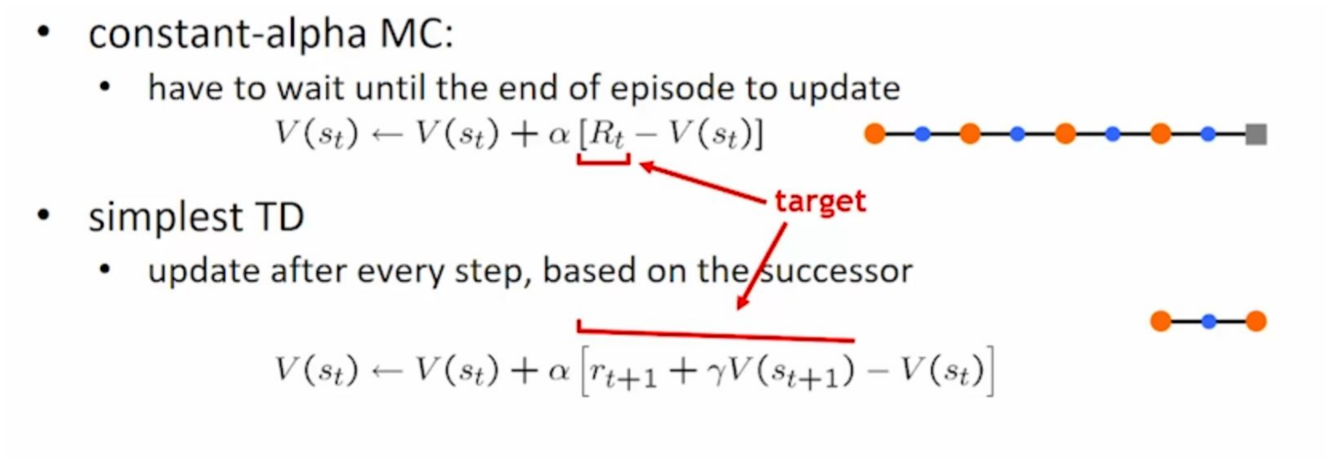


- 5-card draw poker
  - s_0: A♣, A♦, 6♠, A♥, 2♠
  - a_0: discard 6♠, 2♠
  - s_1: A♣, A♦, A♥, A♠, 9♠ + dealer takes 4 cards
  - return: +1 (probably)
- DP
  - list all states, actions, compute P(s,a,s')
    - P([A♣,A♦,6♠,A♥,2♠], [6♠,2♠], [A♠,9♠,4]) = 0.00192
- MC
  - all you need are sample episodes
  - let MC play against a random policy, or itself, or another algorithm

- At every state you treat them as the maximum
- Exploration
  - Deterministic/greedy policy does not explore all actions
    - Don't know environment, need to find all
  - Maintain exploration
    - Use soft policies instead pi(s,a) > 0 for all s,a
  - Greedy policy
    - With probability 1-epsilon perform the optimal/greedy action and with probability epsilon perform a random action
    - Explore environment and slowly move it towards greedy policy epsilon -> 0

Temporal Difference Learning

- Combines ideas from MC and DP
  - Like MC: Learn from experience (Dont need a model)
  - Like DP: Learn from values of successors
  - Works for continuous tasks, usually faster than MC, evaluates at every step



- constant-alpha MC:
  - have to wait until the end of episode to update

$$V(s_t) \leftarrow V(s_t) + \alpha\,[\underbrace{R_t}_{} - V(s_t)]$$

target

- simplest TD
  - update after every step, based on the successor

$$V(s_t) \leftarrow V(s_t) + \alpha\left[\overbrace{r_{t+1} + \gamma V(s_{t+1})}^{} - V(s_t)\right]$$

Quiz

- For dynamic programming, what do we mean by Policy Evaluation?
  - Compute Vπ from π
- For dynamic programming, what do we mean by Policy Improvement?
  - The correct answer is: Improve π from Vπ
- Select all the methods used in Dynamic Programing in RL.
  - Having a perfect model of the environment
  - Start with an arbitrary policy
  - Repeat evaluation and improvement until convergence
- How does Reinforcement Learning overcome not having a complete model of the system?
  - By learning from the environment.
- What is the similarity between Monte Carlo (MC) and Temporal Difference Learning (TDL)?
  - Not needing a perfect model of the environment.
- what is the difference between vpi and qpi in reinforcement learning
  - Vπ evaluates the value of states under a given policy.
  - Qπ evaluates the value of state-action pairs under a given policy.
- what is markovian property in the markov decision process
  - The current state encapsulates all relevant information from the history of the system.
  - The future evolution of the system depends only on the current state and action, not on the entire history that led to the current state.
- when solving for reinforcement learning which method is suitable for continuous taks
  - Temporal Difference
- When comparing the Markov Decision Process, you can execute the policies for an infinite time and compare the accumulated rewards. What is the problem with this?
  - It can result in infinite rewards, which consequently cannot be compared
- Which requires reinforcement learning
  - Sending a space rover to explore the surface of another planet.
- What is a shortcoming of the monte carlo method for reinforcement learning
  - It uses a greedy approach, which is an approximation of the optimal solution but not the optimal solution.
- What is a benefit of the state action value function (Q(s,a)) over the state value function V(s))
  - The state-action value function can be used in a greedy manner for choosing the best action at a current state