Data Science & Artificial Intelligence

Data Science

1. Sample collection	Practical Motivation
2. Data Preparation	Problem Formulation
3. Exploratory Analysis	Statistical Description
4. Analytic Visualisation	Pattern Recognition
5. Algorithmic Optimisation	Machine Learning
6. Information Presentation	Statistical Inference
7. Ethical Consideration	Intelligent Decision

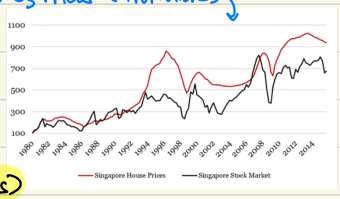
Common Problems

1. Numeric → How much? How many? (Regression) → prediction
2. Classes → Type A? Type B? (Classification) → prediction
3. Structure → How is this organised? (Clustering) → detection
4. Anomaly → Is it weird behavior? (Anomaly Detection) → detection
5. Action → what should be done next? (Adaptive Learning) → decision

Common Data Types

1. Structured Data
   - Highly Organised Data, Clearly Defined Variables, Easy to Mine & Analyse
   - a) Numeric Data → Numeric Continuous Variables
   - b) Categorical Data → Factor/Level / Class Variables (e.g. safety → high, med, low. seats → 2, 4, more)
   - c) Mixed Data → Numeric & Categorical
   - d) Time Series Data → Numeric with TimeStamps (e.g. Stock & Equity Markets, Weather Data over Time, Prices & Promotions)
   - e) Network Data → Nodes (Social Networks & Web, Transport Networks (MRT), Financial Transactions)

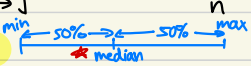
   → Spreadsheet (Excel, CSV), Standard SQL Databases, Sensors & Devices



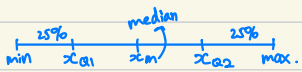
2. Unstructured Data
   - Highly Unorganised Data, Non-Obvious Variables, Highly Context-Sensitive (Words, Phrases, Emoticons)
   - a) Image Data → Pixels & Objects (social networks, mobile phone cameras, blogs, documents) & Text
   - b) Video Data → Images, Frames, Objects (youtube, video messages & calls)
   - c) Voice Data → Voice Signal & Waves (songs, microphones & cameras, recordings, announcements)

Uni-Variate Statistics

- Mean (Average) → $\bar{x} = \frac{x_1 + x_2 + \ldots + x_n}{n}$ ← no. of datapoints
- Standard Deviation (Average Deviation from mean) → $\sqrt{\frac{(x_1-\bar{x})^2 + (x_2-\bar{x})^2 + \ldots + (x_n-\bar{x})^2}{n}}$
- Median (Mid-Value) → $P(x \leq x_m) = P(x \geq x_m) = 0.5$



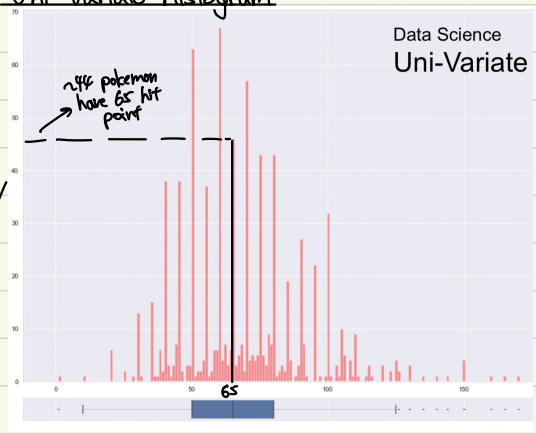
- Quartiles (Divide Data 25:50:25) → $P(x \leq x_{Q1}) = 0.25$, $P(x \geq x_{Q2}) = 0.25$

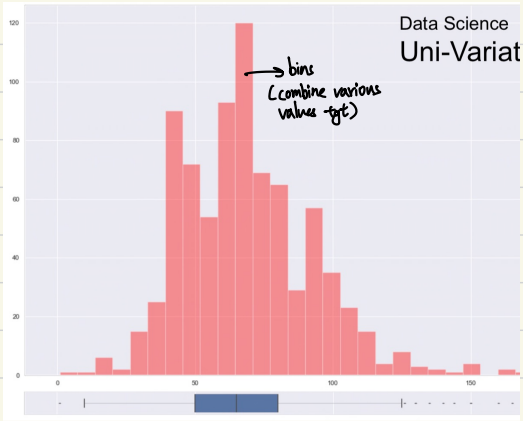


count	800.000000
mean	69.258750
std	25.534669
min	1.000000
25%	50.000000
50%	65.000000
75%	80.000000
max	255.000000

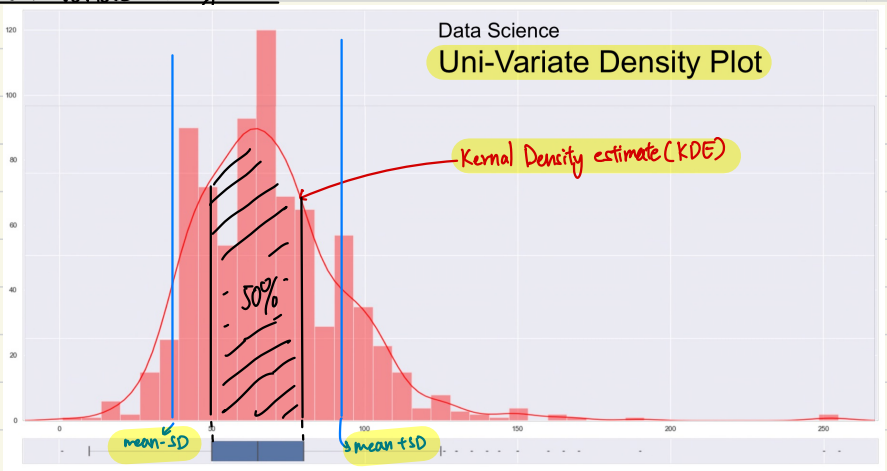
← Uni-Variate Box Plot

Uni-Variate Histogram





Uni-Variate Density Plot



need memorise (only for Normal Distribution)

mean − SD to mean + SD → 68.27%

mean − 2*SD to mean + 2*SD → 95.45%

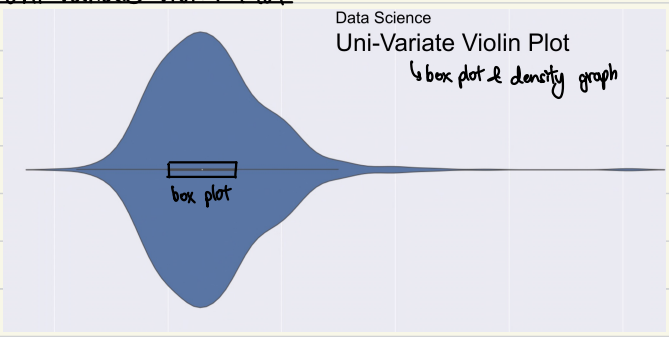
mean − 3*SD to mean + 3*SD → 99.73%

↳ % of data that lies in bw the points

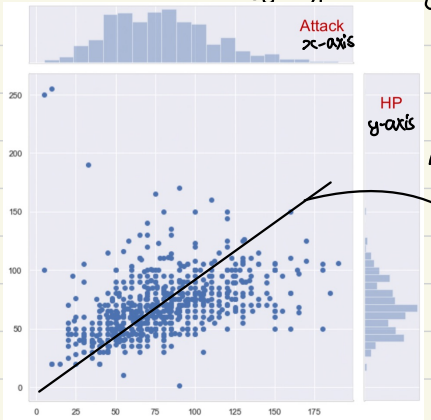
Confidence	Interval in terms of SD
90 %	Mean +/- 1.645 * SD
95 %	Mean +/- 1.96 * SD
99 %	Mean +/- 2.576 * SD

90% confidence that data lies bw mean +/- 1.65*SD

Uni-Variate Violin Plot



Bi-Variate Joint Plot (joining 2 stats tgt)



HP → Plotted along Y axis

Attack → Plotted along X axis

→ HP ↑ as Attack ↑

→ Dependence moderately strong (Correlation Coefficient → $\rho_{x,y} = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2} \sqrt{\sum_i (y_i - \bar{y})^2}}$)

→ The closer the dots to line & the closer the line to y=x → the stronger the correlation

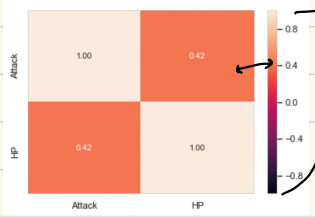
↳ covered below

- No dependence → corr = 0
- Perfect Postive → corr = +1
- Perfect Negative → corr = −1 (one ↓, one ↑)

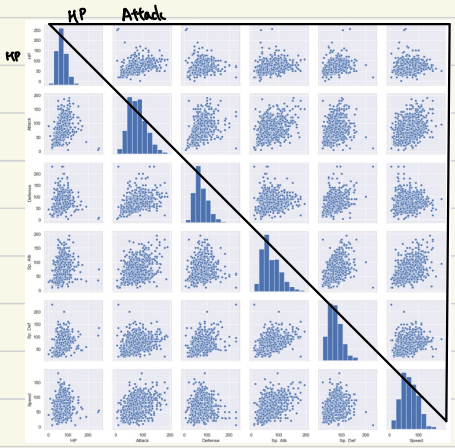
Bi-Variate Relation (Correlation Matrix & Plot)

	Attack	HP
Attack	1.000000	0.422386
HP	0.422386	1.000000

same



→ Strong correlation



→ Pair-Plot of Multi-Variate Data
- Distribution of Variables
- Inter-Variable Dependence

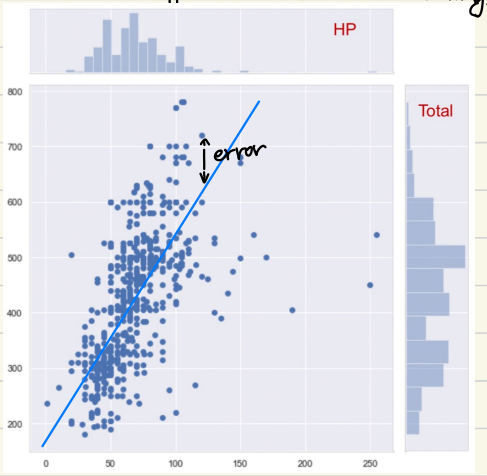
Multi-Variate Statistics

	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed
HP	1.000000	0.422386	0.239622	0.362380	0.378718	0.175952
Attack	0.422386	1.000000	0.438687	0.396362	0.263990	0.381240
Defense	0.239622	0.438687	1.000000	0.223549	0.510747	0.015227
Sp. Atk	0.362380	0.396362	0.223549	1.000000	0.506121	0.473018
Sp. Def	0.378718	0.263990	0.510747	0.506121	1.000000	0.259133
Speed	0.175952	0.381240	0.015227	0.473018	0.259133	1.000000

Bi-Variate Exploration

- Split the dataset into train (use to train the model) & test data (use to test the model)
  - train → 75% of dataset
  - test → other 25% of dataset
- Learn relationship from Train & try to predict the value on test

Uni-Variate Regression (Statistical Modeling)



Hypothesize a Linear Model

- $Total = a \times HP + b + \varepsilon$ → error
  - a → gradient
  - b → y-intercept
- Learn from Train Data & Predict Test Data (a & b same)

Goodness of Fit of Model

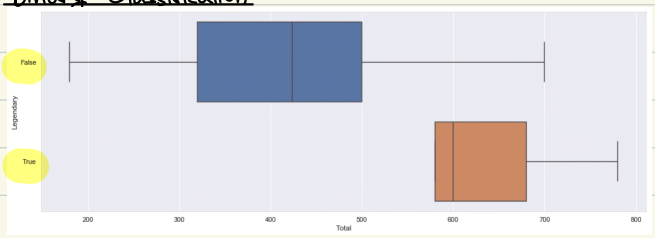
1. Mean Squared Error (MSE)

- $MSE = \frac{1}{n}\sum (Total - a \times HP - b)^2$ → test data. (model)
- The lower the MSE the better the model

2. Explained Variance (R^2)

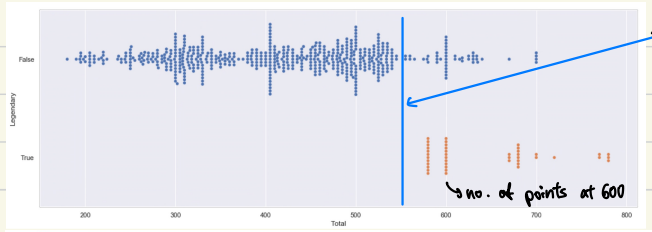
- $R^2 \rightarrow 1 - \frac{\sum (Total - a \times HP - b)^2}{\sum (Total - \overline{Total})^2}$
  - Residual sum of squares (RSS → how much error we make)
  - Total sum squared (TSS)
- The higher the R^2, the better the model, $0 \leq R^2 \leq 1$
- $R^2 = 1 - \frac{MSE}{var(Total)}$

Binary Classification

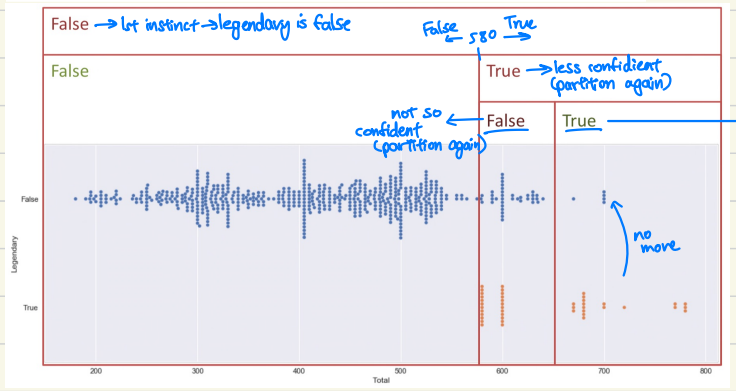


Box Plot

Legendary vs Total have a strong relationship
↳ draw a line at 550, when total < 550, most likely False



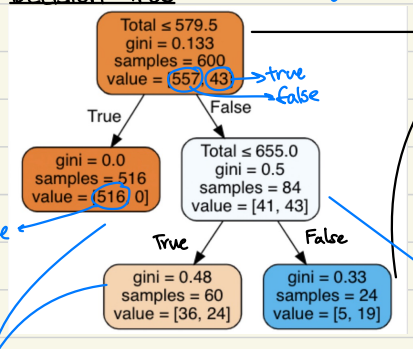
SwarmPlot



Data Space

based on this kinda

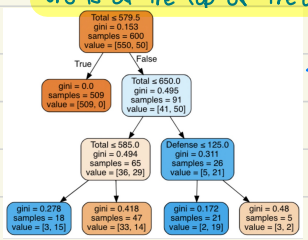
Decision Tree



darken red → confident of it being false
darken blue → confident of it being true

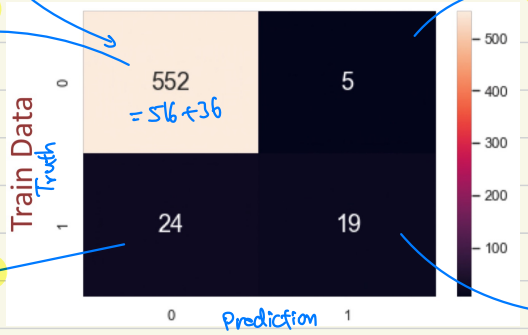
Gini Index (metric of misclassification)
↳ $gini = \frac{x}{n}(1-\frac{x}{n}) + \frac{y}{n}(1-\frac{y}{n})$
$0.5 = \frac{41}{84}(1-\frac{41}{84}) + \frac{43}{84}(1-\frac{43}{84})$

* Most impt factor → based on which one is at the top of the decision tree



total, then defence

Goodness of Fit of Model



True Negatives (TN)
↳ correct prediction of it being false

False Positive

False Negatives

True Positive (TP)

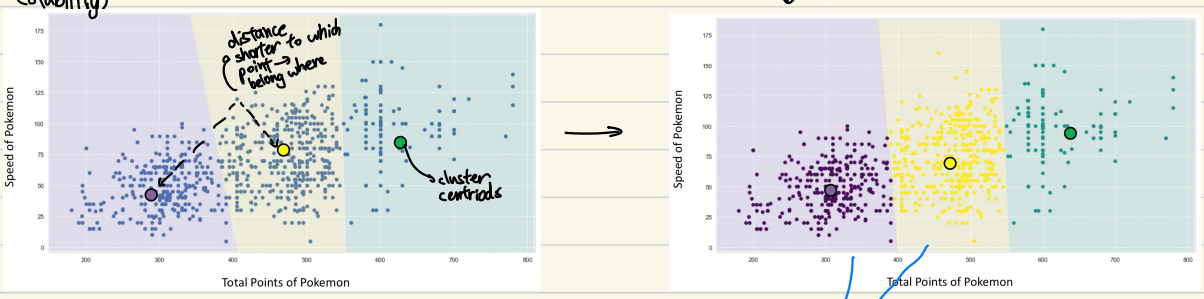
Classification Accuracy (Fraction of Correct Predictions)
↳ Accuracy: $\frac{552+19}{552+5+24+19} = \frac{552+19}{600} = 0.952$

Classification Errors
↳ false positive rate = $\frac{FP}{TN+FP} = \frac{5}{552+5} = \frac{5}{557}$
↳ false negative rate = $\frac{FN}{TP+FN} = \frac{24}{24+19} = \frac{24}{43}$

Precision = $\frac{TP}{PP}$ → predicted positive (TP + FP)

K-Means Clustering

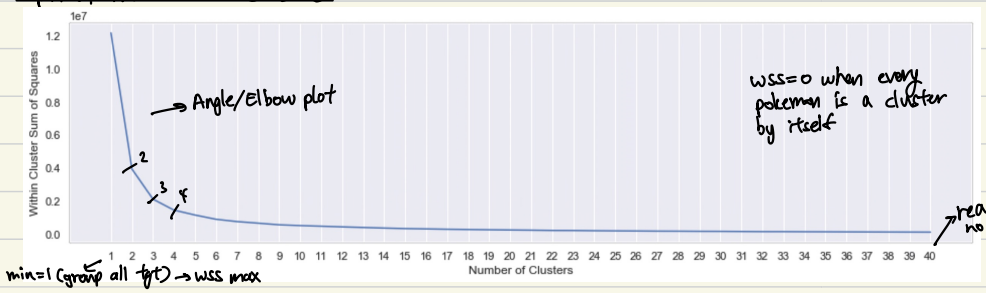
1. Choose K → potential number of clusters & choose K cluster centroids from dataset
2. for each point in dataset → re-label according to nearest centroid → recompute the centroid of cluster → until convergence (stability)



Features	Total	Speed
Cluster 0:	305.65	49.36
Cluster 1:	622.57	97.08
Cluster 2:	474.27	73.55

- Within Cluster Sum of Squares = 2118651
  - ↳ sum of squares of distance of each data point in cluster from centroid $[\Sigma dist(C_1 - c)]^2 + \Sigma dist(C_2 - c)]^2$

Optimal Number of Clusters

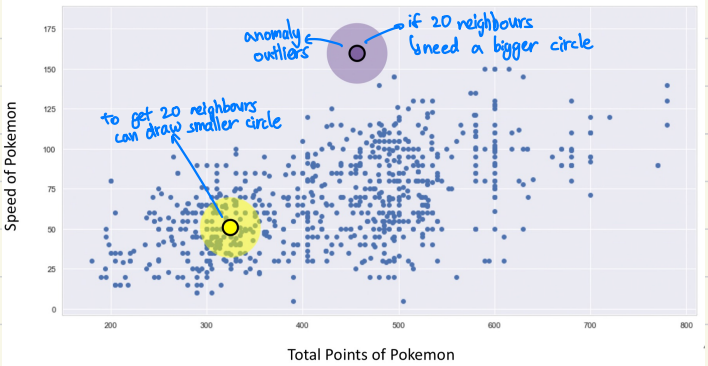


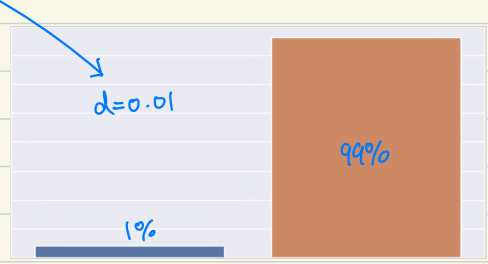
min=1 (group all together) → wss max

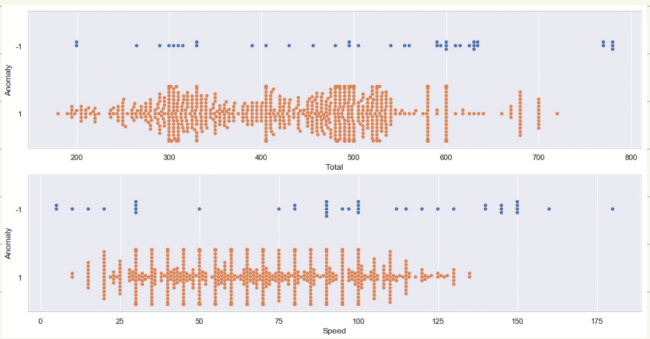
↳ choose no. of clusters by where it last bent → 4

Anomaly Detection (Nearest Neighbor)

1. Choose K → total no. of neighbors → if K is 1 → only compare to one neighbour, if further than neighbour → considered anomaly → emphasizes local anomalies within data, even if they are not at the boundary.
2. Choose d → fraction of anomalies in data
3. for each point in dataset → find K nearest neighbors in data & compute if density is high enough







→ swarm plot

Views of AI

↳ falls into 4 category: Thinking Humanly, Acting Humanly, Thinking Rationally, Acting Rationally

- cognitive modeling → Thinking Humanly
- laws of thought → Thinking Rationally
- rational behavior: doing the right thing → Acting Rationally
- Turing Test → ask question to both machine & human & try to determine them → Acting Humanly

Agent

- → entity that perceives through sensors (eyes, ears, cameras, infrared range sensors)
- → acts through effectors (hands, legs, motor)
- ↳ percepts environment → action by effectors → percepts environment

1. Rational Agents

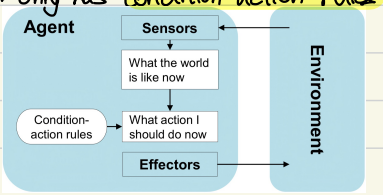
- → do right thing (rational action → maximise expected value of an objective performance measure given percept sequence to date)
- ↳ rationality depends on: performance measure, everything agent perceived so far, built-in knowledge about environment, actions that can be performed

2. Autonomous Agents

- → do not rely entirely on built-in knowledge about the environment → agent only operates successfully when built-in knowledge are correct
- ↳ adapt to environment through experience

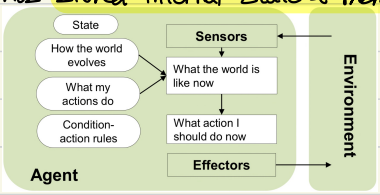
3. Simple Reflex Agents

↳ only has condition-action rules (if car in front brakes then initiate braking)



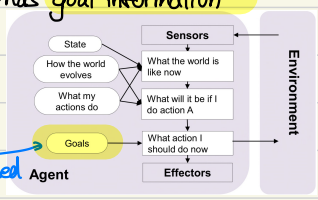
4. Reflex Agents with State

↳ has stored internal state & memory (if yesterday at NTU & no traffic jam now then go Orchard)



5. Goal-Based Agents

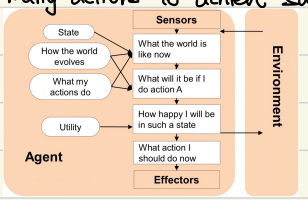
↳ has goal information



when goal reached

6. Utility-Based Agents

↳ many actions to achieve same goal → how happy agent will be if it attains a certain state? → utility



Types of Environments

1. Accessible (knowledge of environment) vs Inaccessible (partial/no knowledge)
2. Deterministic (outcome controlled by action) vs Non-deterministic (outcome cannot be controlled by action)
3. Episodic (previous action does not affect the next) vs Sequential (previous action affect the next)
4. Static (environment does not Δ when agent is deliberating) vs Dynamic (environment Δ when agent deliberating)
5. Discrete (state & action integer) vs Continuous (going left −1 to 5 etc.)

Eg. Chess → accessible (all position in chessboard can be observed), deterministic (outcome of each movement can be determined), sequential (action depends on previous movements), static (no clock, considering moves, opponent can't move), discrete (all position & movements in discrete domain)

Design of Problem-Solving Agent

1. Goal formulation
2. Problem formulation
3. Search process (no knowledge → uninformed search, knowledge → informed search)
4. Action execution

Uninformed search strategies

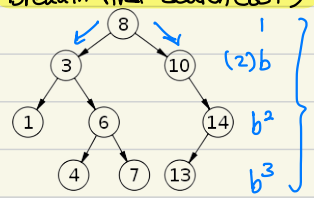
↳ use only information available in problem

Criterion	Breadth-first	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Time	b^d	b^d	b^m	b^l	b^d	$b^{d/2}$
Space	b^d	b^d	bm	bl	bd	$b^{d/2}$
Optimal	Yes	Yes	No	No	Yes	Yes
Complete	Yes	Yes	No	Yes, if $l \geq d$	Yes	Yes

→ cutoff on max depth
only expand another depth when goal not found on current depth

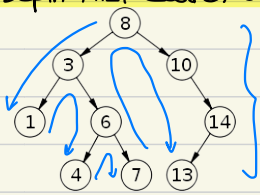
b: branching factor, d/m: max depth (goal depth)

1. Breadth first search (BSF)



space complexity: 2^3

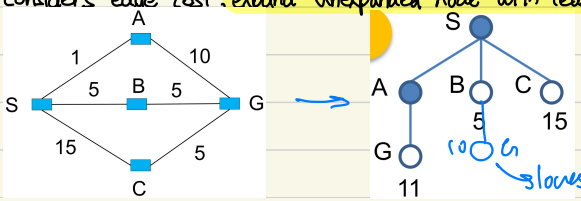
3. Depth-First Search (DFS)



space complexity = 2X3
will delete nodes when backtrack

2. Uniform-Cost Search

↳ considers edge cost. expand unexpanded node with least path cost, weighted graph



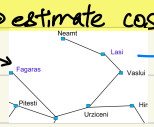
☆ sometimes they give weighted graph, but ask for PURE BFS/DFS

↳ lowest cost

Informed search

- use problem-specific knowledge to decide the order of node expansion
- Uniform-cost search → path-cost function $g(n)$ → no information on cost towards goal
- Greedy search → "heuristic" function $h(n)$ → estimate cost of cheapest path from n to goal state each round

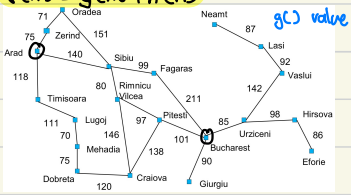
Complete	No
Time	$O(b^m)$
Space	$O(b^m)$(keeps all nodes in memory)
Optimal	No



→ when at Lasi will keep running Neamt since it is shortest path & will never reach Fagaras

A* Search

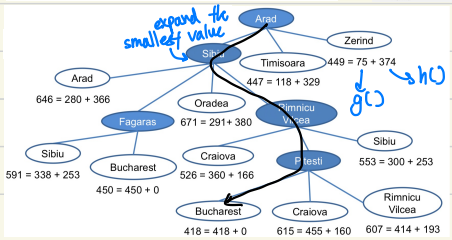
↳ $f(n) = g(n) + h(n)$



g() value

Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Lasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

h() value

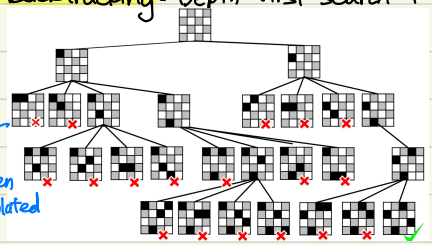


Time	Exponential in length of solution
Space	(all generated nodes are kept in memory) Exponential in length of solution

} with a good h(), significant savings are possible compared to uninformed search methods

Constraint Satisfaction

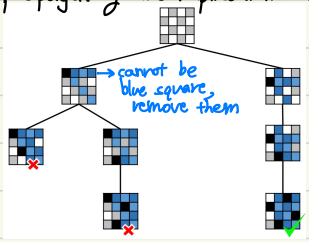
1. Backtracking: Depth first search + constraint checking

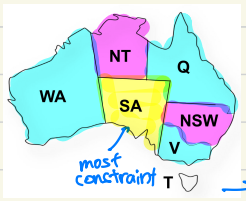


do not waste time searching when constraints violated

2. Forward Checking & Constraint Propagation

↳ propagating the implications of a constraint on one variable onto other variables

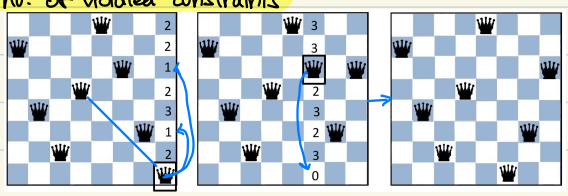




→ set the color of most constraint one first → & go down list

3. Min-Conflicts Heuristic

↳ given initial assignment, selects a variable in scope of violated constraint & assigns it to value that minimises no. of violated constraints



Minimax Search Strategy

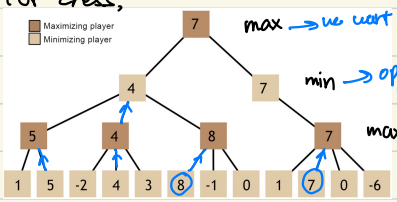
→ minimax search happens everytime the agent make decision → find a sequence of moves that leads to goal

→ maximise one's own utility & minimise opponent's. Assume opponent also does the same

→ 3 steps:

1. Generate entire game down to terminal states
2. Calculate utility → determine best utility at terminal state up to root
3. Select best move (highest utility value)

E.g. For chess,

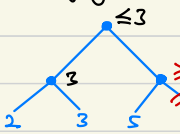


max → we want to maximise our score

min → opponent want to minimise our score

max

- Game Playing: Pruning (depth-first search)

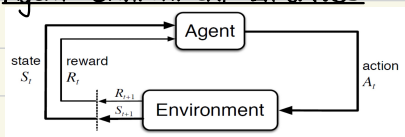


min — 1st step

max — 2nd step

prune away min

Agent-Environment Interface



→ state, action, reward

→ maximise reward received as agent operate in environment (start to end)

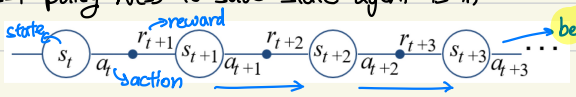
Markov Decision Process

→ probability of going to next state (s') from s depends on s & a & not other past action/states

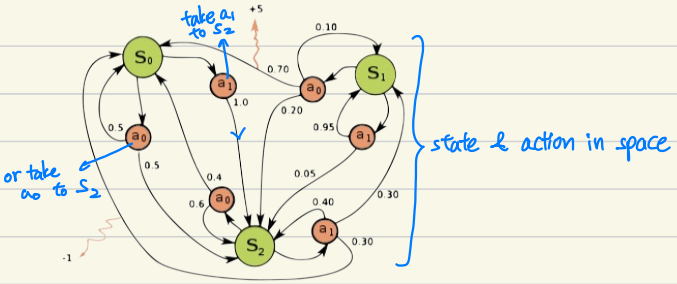
→ components: states (s), actions (a), transition model (P(s'|s, a)), Reward function (R(s)).

→ solution: best policy π(s) to solve state agent is in

linear model:

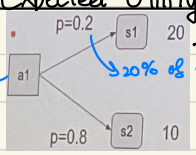


behaviour in time



state & action in space

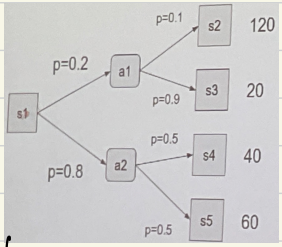
Expected Utility



action

20% of time reach state 1

↳ expected utility of action "a1" = 20×0.2 + 0.8×10 = 12



↳ Expected utility of state "s1" = 46 (probabilistic)

→ deterministic policy

a1 vs a2

12+18 20+30

30 vs 50 → give more expected utility → take a_2

State sequence Utility / Discounted Utility

- discount the individual state rewards by a factor $0<\gamma<1$
- sooner rewards count more than later rewards → γ^t will decrease to 0 as $t\uparrow$

$$U([s_0, s_1, s_2 \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots$$

$$\star = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \frac{R_{max}}{1-\gamma} \quad (0<\gamma<1)$$

Bellman Equation

memorise ★

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) U(s')$$

- $U(s)$: current state utility
- $R(s)$: current state reward
- γ: discounted coefficient
- $\max_{a \in A(s)}$: best action of a_1 & a_2 → so → for a_2
- $\sum_{s'} P(s'|s,a)$: state in future
- $U(s')$: next state utility; s': state now

$$41 = 1 + 0.8(50)$$

Value Iteration

- start out with $U(s) = 0$ → iterate until convergence

Greedy policy
↳ pick state with highest utility

→ apply Bellman eqn multiple times

Policy Iteration

- Policy evaluation: calculate $U^{\pi_i}(s)$ for every state s
- Policy improvement: calculate new policy π_{i+1} based on updated utilities

} repeat

$$\pi^{i+1}(s) = \arg\max_{a \in A(s)} \sum_{s'} P(s'|s,a) U^{\pi_i}(s')$$

Temporal difference (TD) Prediction

- for a given policy ρ → compute state-value function V^{π}

$$TD(0): V(s_t) \leftarrow V(s_t) + \alpha [r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

target: an estimate of return

$s_t \rightarrow s_{t+1}, r_{t+1}$

- do not require model of environment, only experience
- can learn before knowing final outcome (less memory, peak computation)
- can learn without final outcome (from incomplete sequences)

- In which of the following ways can you interpret LinkedIn Data as a Network (or graph).
  - Accounts as nodes & Contacts (yes/no) as links. ✓
  - Accounts as nodes & common Experience as links ✓
  - Accounts as nodes & number?? of individual contacts as links. ✗

need becareful to see if question get ~~easy~~ say if it is a normal distribution. if not cannot use. SD 68.27%

- Which ones of the following are decent choices for Cost Function in Linear Regression? Multiple options may be correct.
  1. $\Sigma_i (\text{Response} - \text{Prediction})^2 \rightarrow$ sum square of errors / residual sum of squares over train set ✓
  2. $\Sigma_i |\text{Response} - \text{Prediction}| \rightarrow$ absolute sum of errors / residual absolute sum over train set ✓
  3. $\min \{\text{Response} - \text{Prediction}\} \rightarrow \min$
  4. $\max (\text{Response} - \text{Prediction})^2 \rightarrow \max$

don't wanna get max/min of train data.

- What is the relation b/w outliers of two Uni-Variate datasets & the anomalies in their joint Bi-Variate data set? That is, what is the relation b/w outliers in the boxplot of X & Y with anomalies in jointplot of X-Y?
  Point not tagged as outliers in individual variables X & Y may be tagged anomalies in the X-Y case.