Artificial Intelligence with Python

represent info → conclusion

Search:
1. 15 puzzle
2. direction

Terms:
agent → perceives environ → act
State → configuration
initial state → action(s) (validity)
transition model → input s, a
↳ graph
↓
goal test (low cost)
↓
numerical cost [path cost]
↓
optimal sol.

Stack → LIFO → DFS frontier
↳ Deadend → backup → best sol ✗
Queue → FIFO → BFS simultaneously checking both paths

greedy best first search
↳ expands node closest to the goal
↓
approximate

A* Search **optimal sol.**
counting steps
h(n) = consistent

Adversarial Search → Minimax (tic-tac-toe) Algo
O ↓ X
recursive algo

Alpha-Beta pruning

Depth-Limited Minimax + evaluation function

knowledge
↓
deduction
propositional logic

knowledge Engineering
clue game
Mastermind
↓
Inference Rule
De morgan's law
$\neg(\alpha \vee \beta) \to \neg\alpha \wedge \neg\beta$
Distributive Prop.
Disjunction → or
Con " → and
First order logic
constant (obj.) / Predicate prop.

Universal Quantification
hold true all
Existential
↳ for some objects

Uncertainty
Probability
$0 \leq P(\omega) \leq 1$
$\sum P(\omega) = 1$
w∈Ω

conditional:
$P(a|b) = \frac{P(a \wedge b)}{P(b)}$ ⇒ $P(a \wedge b) = P(b)P(a|b)$ = $P(a)P(b|a)$
abstract evidence
↓ independence
$P(sum 12 | [die]) = \frac{1}{6} = \frac{1/36}{1/6} \times \frac{6}{1} = \frac{1}{6}$, $= \frac{1}{36}$

Bayes' Rule
$P(b|a) = \frac{P(b)P(a|b)}{P(a)}$

joint probability
↳ normalization const. α

$P(\neg a) = 1 - P(a)$
$P(a \vee b) = P(a) + P(b) - P(a \wedge b)$
$P(a) = P(a,b) + P(a, \neg b)$ [Marginalization]
$P(X = x_i) = \sum_j P(X = x_i, Y = y_j)$
$P(a) = P(a|b)P(b) + P(a|\neg b)P(\neg b)$ [Conditioning]

Bayesian network
· directed graph
· node → random var.
X → Y parent
$P(X | Parents(X))$

Sampling
fixing the evidence var.

Markov assumption
chain

Hidden Markov Model
Sensor data

Optimization
Hill climbing
Algo (moving to a neighbour)
↳ ✗ optimal sol.
Random start ✓
Simulated annealing
temperature ↑, neighbour
$\Delta E = +$ better than current state
$-$ not better
$e^{\Delta E/t}$ → go ahead to the state
TSP (NP complete) switch edges
↳ approximation
↳ local search ↑

→ focus on only solution

Linear Programming (linear constraints)
↓
1. Simplex
2. Interior-point

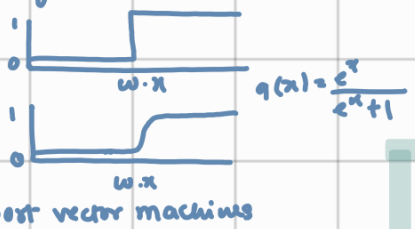
Constraint Satisfaction
Set: var, Domain, const.
↓ graph
Sudoku
Var {(0,2), (1,1), (1,2), ..}
Dom {1, ..., 9}
constraints {(0,2) ≠ (1,1), ..} → hard

binary
arc consis. → queue
node consis.
↓
unary constra.
A — B
{Mon, Tue, Wed} {Mon, Tue, Wed}
{A ≠ Mon, B ≠ Tue, B ≠ Mon, A ≠ B}

Backtracking search
try values
failure (violates)
↳ backtrack
or
check arc-consistency
↓
highest degree heuristic
↓
Domain value

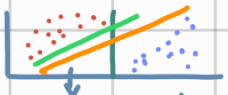
Learning

- Supervised
  - i/p → o/p (given)
  - function? ✓
  - nearest neighbor point
    - k → majority vote (naive)
  - weight vector w, input vector x } dot product
    - $h_w(x) = 1$ if $w \cdot x \geq 0$, 0 otherwise
  - perceptron learning value
    - $w_i = w_i + \alpha(\text{actual val} - \text{estimate}) \times x_i$
  - hypothesis function.

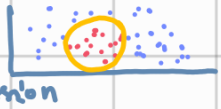


$g(z) = \frac{e^z}{e^z + 1}$

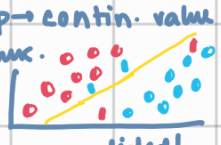
- Support vector machines



- max margin separator (decision boundary)



- regression
  - o/p → contin. value
  - loss func.



- $L_1 = |\text{actual} - \text{predicted}|$
- $L_2 = (\text{Regularization cost}(h) = \text{loss}(h) + \text{complexity}(h))^2$
- scikit-learn

Reinforcement learning

- what to do after experiences/actions
- Markov Decision Model
  - Transition model $P(s'|s,a)$ ← action
  - next state
  - Reward/Punishment ← how much
- Q-learning
  - $Q(s,a) \leftarrow Q(s,a) + \alpha(\text{new} - \text{old})$
  - value estimate
  - new info
  - $\leftarrow Q(s,a) + \alpha(r + \max_{a'} Q(s',a') - Q(s,a))$

Explore vs Exploit

- ε-greedy
- $1-\varepsilon$ → estimated best move
- or
- ε → random move

Nim

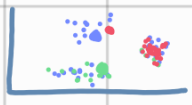
- ↳ obj from pile
- ↳ remove
- ↳ last obj → lose

- func approximation

Unsupervised Learning

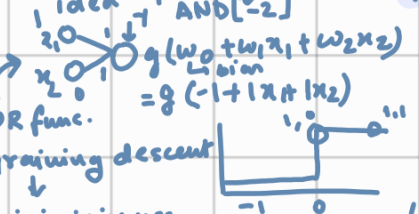
No labelling

- ↳ clustering
  - Genetic research
  - Market research
  - K-means updating cluster's centres



Artificial Neural Network

- biological idea →



- graining descent
  - ↳ minimizing loss
    - ↳ one data point
  - Mini Batch
    - ↳ one small batch
    - ↳ more accurate

multilayer

↳ hidden layer

back propagation algorithm

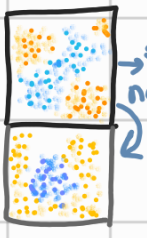
- calculate error for output layer
- ↓
- hidden layer
- ↓
- back layer

Deep neural network

↳ multiple ← hidden layer

- dropout technique
  - ↳ over reliance on certain units
  - dropping out random nodes

Tensor-flow



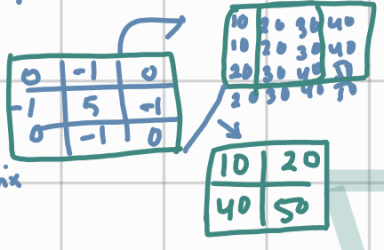
computer-vision

[tagging photos] [handwriting recognition]

- extracting info

image convolution

- add pixel value
  - ↳ its neighbor
    - ↳ weighted acc. to kernel matrix



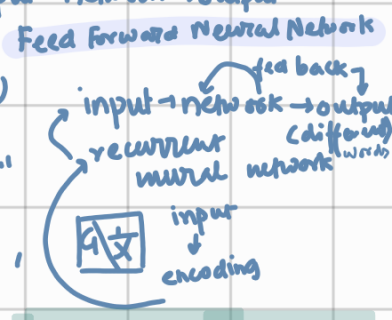
pooling

- ↳ max-value
- ↓
- smaller representation of big picture

Convolution Neural Network

- ↓
- Av. Pooling [reducing dimension]
- ↓
- Flattening

Feed forward Neural Network



Natural Language Processing

Syntax: structure
Semantics: slight diff. word representation
↓
same meaning

Context free grammar

NP → N | D N
↓ ↓
Noun Phrase | Determiner

VP → V | V NP
↓
Verb

S → N? VP
↓
Sentence

Natural Language Tool Kit

n-gram
↳ common words
tokenization
↳ markov chains
Text classification:
↳ Email < good / spam
↳ Product review < good / bad

Bag of words model
↳ Naive Bayes

$P(☺) \rightarrow P(☺ \mid$ "my grandson loved it")

$P(☹)$

$P(\text{"my"}, \text{"grandson"} \ldots \mid ☺)\, P(☺)$

$P(☺) = \frac{\text{no of true sample}}{\text{no of total "}}$

additive smoothing
↳ Laplace → $\alpha / 1$

word Representation
↳ No.

he	[1,0,0,0]
work	[0,1,0,0]
a	
book	

} One-hot representation ✗ ideal
↓
distributed representation

word2vec
↳ relationship b/w words

Recurrent Neural Networks → previous step (dependency)
English ▭ French

Encoder → output
Attention scores
↳ hidden state vectors

Transformer architecture.
↳ parallelize each i/p + Independently
positional encoding + Self Attention (Multi-headed)
+
Attention
↳ encoded representation
[Decoding]

Artificial Intelligence

Machine Learning ← Subset of AI

parse data, learn to make informed decision

ML
feature engin
small data
simpler

DL
feature extrac
large data
complex

Activation Function
ReLU: $f(x) = \max(0, x)$
Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$
Tanh: $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
Leaky ReLU: $f(x) = \max(\alpha x, x)$

Types
Feedforward
Convolutional (image processing)
Recurrent (sequential data)
GAN (generator / discriminator)

Parameters
adjusted to ↓ loss func (weights)

Hyper
set before the training proces (no of epochs)

Multiclass classification
each data point → exactly one class

Multi-label
each data point → multiple class (news article)

Loss Function
discrepancy btw predicted o/p & actual target value
MSE

Bayesian learning
probability of each hypothesis

LLM
AI: human quality text generation

CNN
work: applying filters → i/p data extracting edges

Importance of Pooling Layers
↓ comp
sensitive to small shifts in i/p
feature extraction

Autoencoder
Encoder, $z = f(x)$
Decoder, $\hat{x} = g(z)$
goal: ↓ the reconstruction loss btw x & $\hat{x}$
$L = \|X - \hat{X}\|^2$

RNN
handle long sequences ✓
difficult to learn long term dependencies ✗