Reinforcement learning.

⊛ Deep Q-learning //

→ GPU for RL.
→ CPU is enough

Action / Reward + Penalty.

Theory of RL : Part 1

⊛ RL → [agents → take action in order to make the notion of cumulative reward.]

⊛ RL is teaching a software agent how to behave in environment by telling how good it is.

((CPU is enough))

Deep-Q-learning

RL using NN (deep neural networks).

Agent.

- game (env)
- model

Training loop.

↳ state.
↳ action (predict)
↳ next step
↳ reward.
↳ remember (model_train)

Env. game

⊛ play-step (action)
→ reward, game_over, score

Model.

⊛ Linear Q_Net. (DQN)

⊛ DQN..
- model_predict (DQN)

⊛ **Reward in Game** :- : +10 (eat food)

: -10 (game over)

: - else (0)

⊛ **Action** :-

Next move :- { [1,0,0] -> straight

[0,1,0] - right turn

design [0,0,1] - left turn

// depends on current direction

⊛ **State** :-

(tell our snake some information about environment)

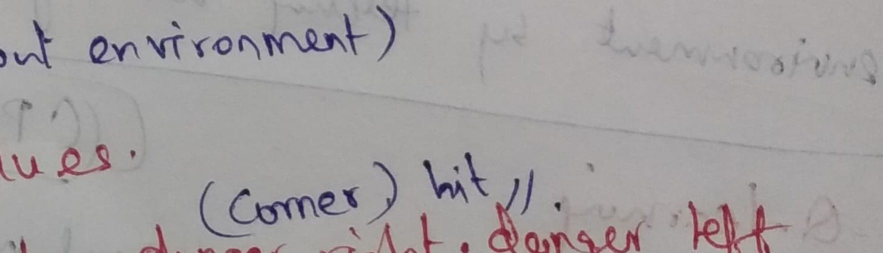
11 values.

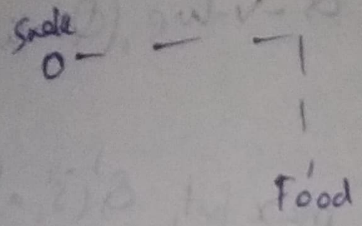
(Corner) hit //

[danger straight, danger right, danger left,

↑ dir, →, ↓, food left ↑ ↓ → (←)

All are boolean values.





(1,3,3)-(1,2)=(1,2,1)?

[0,0,0, 0,1,0,0, 0,1,0,1]

danger, direction, food

boolean code

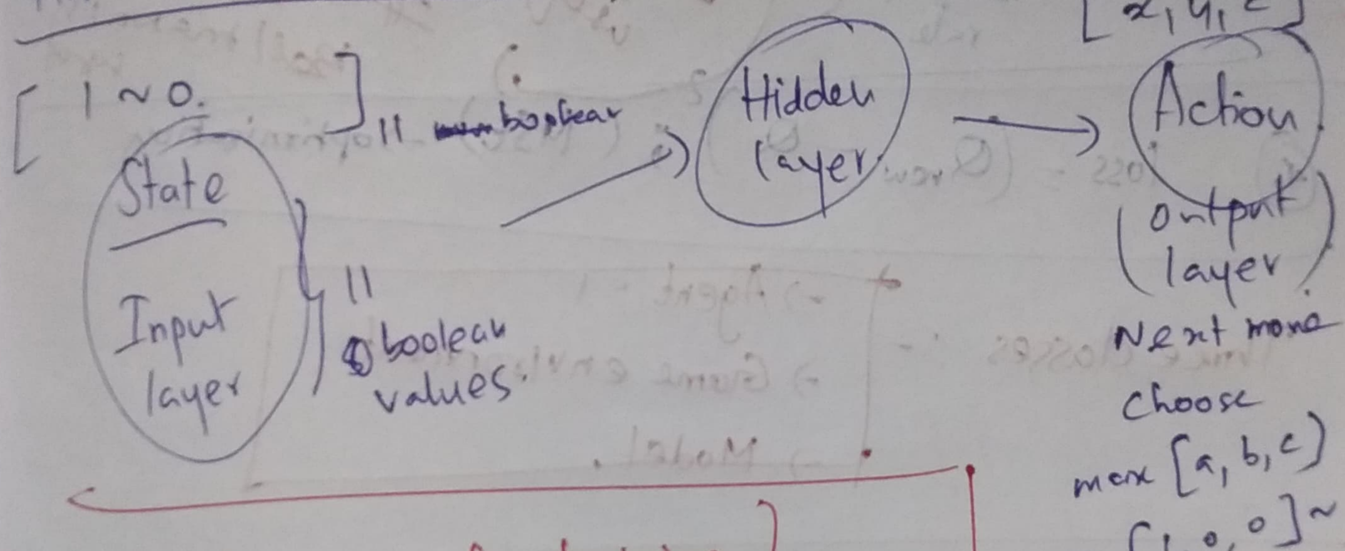
input

g(1,(1,2)

cyclic

g(c,(1,2))

FNN neural network :-



Deep Q - learning for training the model

Q value = quality of action (improve Q value).

1. init Q-value (= init model)
2. Choose action (model.predict(state)) → random
3. Perform action
4. Measure reward.
5. Update Q value (+ train model).

repeat training loop

Need loss fn for training :- } update Q-value, (Quality)

Bellman equation :-

New $Q(s,a) = \alpha$ [Reward $R(s,a) + \gamma$ max fut reward $Q'(s',a') - Q(s,a)$]

learning rate

(Updated Bellman is used)

discount rate

Updated Bellman is used

loss $= (Q_{new} - Q)^2$; (MSE) → optimisation

Three classes :-
- → Agent - 1
- → Game environment
- → Model.

part-2

Implementation in (Pytorch).

→ Create environment from conda.

git :-
→ Snake_game_human.py

Place food → randomly is helper function //

*) Need to update → snake_game_ai.py

(1.) # reset fn (2.) Reward (3.) Game_iteration (4.) is_collisio

#. Many helper functions.

#. Change :-

Tutorial-2 game is define

Tutorial-3 → Agent is defined

Lear Patric

(*) Deque Memory Storage device.

→ Exploration / exploitation.

→ predict = self.model.predict (state0).

Run python agent.py

Tutorial 4

1.) [Model.py] → Q-Net →
→ Optimization →

1.) game . 2.) agent 3.) model

Game → Agent ⟷ Model