09/19

```
for episode = 1, 2, ....
    S ~ do
    for t = 0, 1, 2, .....
        a = agent.getAction(s)
        s' = P(s, a, .)
        r = R(s, a, s') // Deterministic reward.
        #agent.train(s, a, r, s')
        if (s' == s*)
            break;
        s = s'
```

① Model Based Approach → Estimate P, R, plan
Model Free Approach

② Model Free Approach → Value function based.

③ Policy Gradient

④ Black Box Optimization (BBO)

Read CMAES

BBO for policy search:

→ Simple

→ Ignores MDP's structure.

→ Phrases the problem as a Black Box Optimization problem.

$$\underset{\pi}{\operatorname{argmax}} \; J(\pi)$$

- Only have access to an estimate $\hat{J}(\pi)$

Run π for N episodes and average the returns.

$\gamma < 1 \Rightarrow$ Rewards are bounded.

$$\hat{J}(\pi) = \frac{1}{N} \sum_{i=1}^{N} G^i = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{\infty} \gamma^t R_t^i$$

R_t^i = Reward at time t at i^{th} episode

$$G_0^i \triangleq \sum_{t=0}^{\infty} \gamma^t R_t^i$$

How to represent π?

Tabular representation.

	a_1	a_2	a_3
States s_1			
s_2			
s_3			
⋮			

"Parameters" are numbers we store to represent Something ⟶ policy.

- Tabular policy has $|S| \times |A|$

All $|S| \times |A|$ matrices are valid policies iff
- Rows all sum to 1
- All entries ≥ 0.

Tabular Softmax Action Selection:-

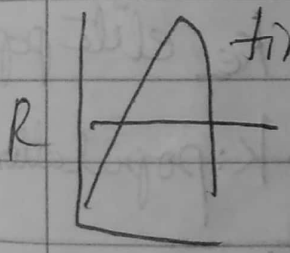
Want to store π as an $|S| \times |A|$
- No constraints on ρ.
- Increasing $\rho(s,a)$ increases $\pi(s,a)$

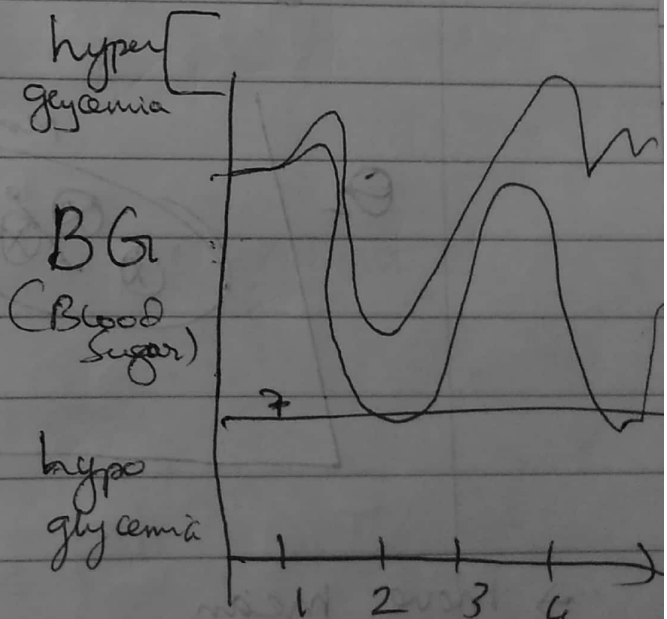
$$\pi(s,a) = \frac{e^{p(s,a)}}{\sum_{a'} e^{p(s,a')}} \rightarrow \text{Boltzmann distribution}$$

$$\sum_a \pi(s,a) = \sum_a \frac{e^{p(s,a)}}{\sum_{a'} e^{p(s,a')}} = \frac{\sum_a e^{p(s,a)}}{\sum_{a'} e^{p(s,a')}} = 1$$

$$e^{p(s,a)} > 0 \quad \forall\, p(s,a) \in R$$

Policy Parameter Vector θ = vector(p).





injection = $\frac{BG - BG_{target}}{CF\ \theta_1} + \frac{Size\_Meal}{CF\ \theta_2}$

(State → $BG - BG_{target}$; State → $Size\_Meal$)

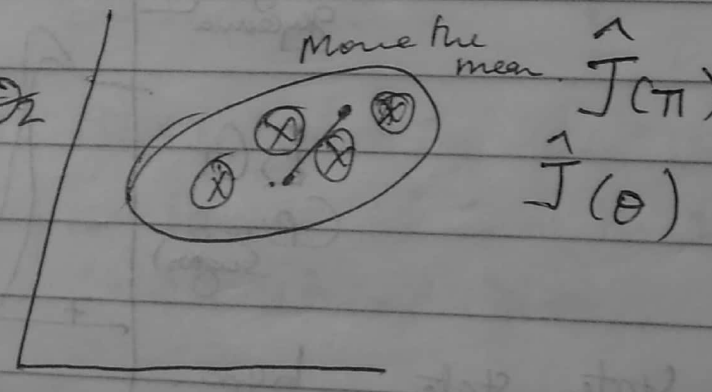
$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

Deep learning: $ANN_\theta(s_t)$

Cross-Entropy Method:

1. Generate a random data sample according to a parameterized mechanisms
2. Update the parameters of the mechanisms based on the data to provide a better sample at the next iteration

- Mechanism = distribution over policies
- Random sample = histories generated by sample policies



k_e: elite population

k: population

$k_e < k$

N: Number of episodes on each sample policy

→ Move mean

→ Covariance ⇒ covariance of top k_e elite

CEM

Input: θ: Mean parameter vector

Σ: Covariance matrix $\Sigma = \sigma \underline{I}$

K: Population ; K_e: Elite Population

Number of times to evaluate policy.

1) for $k = 1 : K$

2) $\theta_k \sim N(\theta, \Sigma)$

3) $\hat{J}_k$ = evaluate (θ_k, N)

4) Sort $(\theta_{1:k}, \hat{J}_{1:k}, \text{descending})$

5) $\theta \leftarrow \frac{1}{K_e} \sum_{k=1}^{K_e} \theta_k$

6) $\Sigma \leftarrow \frac{1}{K_e} \sum_{k=1}^{K_e} (\theta_k - \theta)(\theta_k - \theta)^T$

Numerical instability for small values.

$$\Sigma \leftarrow \frac{1}{K_e + \varepsilon} \left(\varepsilon \underline{I} + \sum_{k=1}^{K_e} (\theta_k - \theta)(\theta_k - \theta)^T \right)$$