

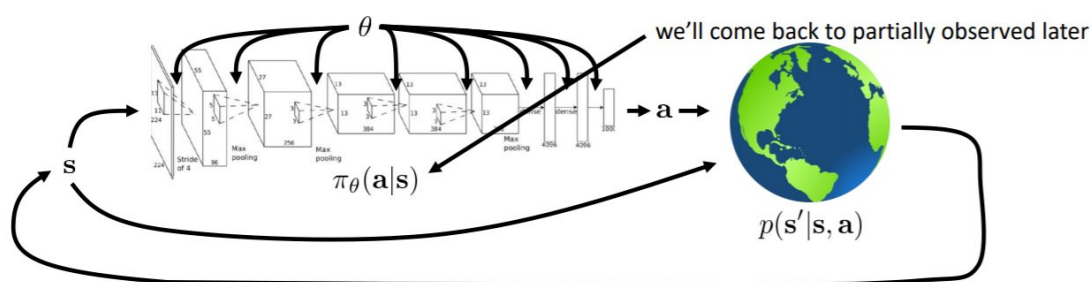
Policy Gradient

Policy Gradient: non genera un valore per ogni coppia stato-azione e poi ne calcola il migliore (supporta anche casi in cui le azioni sono nel dominio continuo) ma produce direttamente un'azione associata ad ogni stato. Quello che si fa è costruire direttamente una policy (rappresentata con una rete neurale il cui input dipende dallo stato del problema e l'output dalle azioni) e la si migliora tramite backpropagation.

Per sfruttare la backprop abbiamo bisogno di 3 cose:

- una funzione obiettivo da minimizzare/massimizzare
- un dataset
- un criterio per valutare l'efficacia della rete (policy)

Policy Evaluation:



Per calcolare la bontà del set di pesi della policy calcoliamo il valore atteso del reward totale in base alla policy corrente.

$$U(\theta) = \mathbb{E}\left[\sum_{t=0}^H R(s_t, u_t); \pi_{\theta}\right]$$

La formula pratica poi quando si implementa è:

$$\frac{1}{N} \sum_i \sum_t r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

sum over samples from π_{θ}

Dataset:

Per aggiornare la rete noi dobbiamo produrre un batch di dati eseguendo un certo numero di episodi e raccogliendone le informazioni necessarie (stat-azione-reward per ogni step).

Ad ogni epoca di addestramento verrà usato un diverso batch di dati che definirà la traiettoria del gradiente per quell'epoca.

(da ora ci riferiremo ai dati raccolti col termine di traiettoria e il simbolo τ)

Il reward di una traiettoria è quindi:

τ = sequenze di stato-azione

Quindi il valore di total reward di un episodio (H) è dato dal reward di tutte le s,a raccolte in τ

$$R(\tau) = \sum_{t=0}^H R(s_t, u_t).$$

Funzione obiettivo

A questo punto possiamo definire la funzione obiettivo

$$\max_{\theta} U(\theta) = \max_{\theta} \sum_{\tau} P(\tau; \theta) R(\tau)$$

Ovvero il valore che vogliamo massimizzare è la sommatoria dei reward prodotti da ogni traiettoria pesata in base alle probabilità che ha la policy di generare tale traiettoria. In altre parole vogliamo massimizzare le probabilità delle traiettorie migliori in modo che contribuiscano maggiormente al calcolo della sommatoria.

Le probabilità che una policy di generare una certa traiettoria sono espresse in questo modo:

$$\underbrace{p_{\theta}(s_1, a_1, \dots, s_T, a_T)}_{p_{\theta}(\tau)} = p(s_1) \prod_{t=1}^T \pi_{\theta}(a_t | s_t) p(s_{t+1} | s_t, a_t)$$

Ovvero dato lo stato iniziale (s_1), si prende il valore di probabilità che la policy associ a quello stato la stessa azione della traiettoria, lo si moltiplica per le probabilità che a quello stato e a quell'azione sia associato lo stesso nuovo stato presente nella traiettoria, moltiplicato per l'azione 2 della policy e così via

Riscriviamo la funzione obiettivo in questo modo:

$$\theta^* = \arg \max_{\theta} E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

Ovvero vogliamo trovare il parametro ideale che massimizzi il valore atteso delle traiettorie (valore traiettoria * prob. traiettoria).

Ora che abbiamo una funzione obiettivo dobbiamo riuscire a **derivarla**:

$$J(\theta) = E_{\tau \sim \pi_{\theta}(\tau)} \left[\underbrace{r(\tau)}_{\sum_{t=1}^T r(\mathbf{s}_t, \mathbf{a}_t)} \right] = \int \pi_{\theta}(\tau) r(\tau) d\tau$$

Specifichiamo che per $r(\tau)$ si intende la sommatoria dei reward accumulati. *L'integrale appare perché stiamo lavorando con una distribuzione di probabilità.*

Introduciamo il gradiente:

$$\nabla_{\theta} J(\theta) = \int \nabla_{\theta} \pi_{\theta}(\tau) r(\tau) d\tau$$

Prima di procedere sfruttiamo il log trick:

a convenient identity

$$\pi_{\theta}(\tau) \nabla_{\theta} \log \pi_{\theta}(\tau) = \pi_{\theta}(\tau) \frac{\nabla_{\theta} \pi_{\theta}(\tau)}{\pi_{\theta}(\tau)} = \nabla_{\theta} \pi_{\theta}(\tau)$$

Per cui se io moltiplico e divido per $\pi_{\theta}(\tau)$ non cambia il risultato ma mi permette di usare sostituire la parte blu con la rispettiva gialla e usare il log. In questo modo otteniamo:

$$\nabla_{\theta} J(\theta) = \int \nabla_{\theta} \pi_{\theta}(\tau) r(\tau) d\tau = \int \pi_{\theta}(\tau) \nabla_{\theta} \log \pi_{\theta}(\tau) r(\tau) d\tau$$

A questo punto dovremmo calcolare le probabilità che ha la policy di generare la traiettoria il che è complicato visto che non abbiamo conoscenza del modello di transizione. Avendo

però raccolto tante traiettorie sotto la data policy, possiamo approssimare le probabilità semplicemente facendo una media sul batch raccolto.

Sostituiamo l'integrale con il valore atteso:

$$\nabla_{\theta} J(\theta) = \int \nabla_{\theta} \pi_{\theta}(\tau) r(\tau) d\tau = \int \pi_{\theta}(\tau) \nabla_{\theta} \log \pi_{\theta}(\tau) r(\tau) d\tau = E_{\tau \sim \pi_{\theta}(\tau)} [\nabla_{\theta} \log \pi_{\theta}(\tau) r(\tau)]$$

E arriviamo alla formula

$$\nabla_{\theta} J(\theta) \approx \frac{1}{m} \sum_{i=1}^m \nabla_{\theta} \log \pi_{\theta}(\tau^{(i)}) r(\tau^{(i)})$$

dove m è uguale al numero degli episodi memorizzati (ampiezza del batch).

Cosa significa esattamente $\log \pi_{\theta}(\tau)$?

Riprendiamo quanto detto all' inizio, ovvero cosa significa avere assegnare una probabilità sotto una certa policy ad una certa traiettoria:

$$\underbrace{p_{\theta}(s_1, a_1, \dots, s_T, a_T)}_{p_{\theta}(\tau)} = p(s_1) \prod_{t=1}^T \pi_{\theta}(a_t | s_t) p(s_{t+1} | s_t, a_t)$$

Notiamo che:

$\pi_{\theta}(a_t | s_t)$ = queste probabilità sono dettate dalla policy e dipendono da θ

$p(s_{t+1} | s_t, a_t)$ = queste probabilità sono dettate dal modello di transizione dell' env (cosa che noi non conosciamo) e non sono influenzate da θ .

Possiamo sfruttare questo per semplificare la formula.

Innanzitutto ora non abbiamo più a che fare con la probabilità in se ma con il suo logaritmo, e siccome il logaritmo di un prodotto è la somma dei logaritmi allora possiamo portare dentro il log:

$$\begin{aligned} \log \text{ of both sides } & \left(\underbrace{\pi_{\theta}(s_1, a_1, \dots, s_T, a_T)}_{p_{\theta}(\tau)} = p(s_1) \prod_{t=1}^T \pi_{\theta}(a_t | s_t) p(s_{t+1} | s_t, a_t) \right) \\ & \log \pi_{\theta}(\tau) = \log p(s_1) + \sum_{t=1}^T \log \pi_{\theta}(a_t | s_t) + \log p(s_{t+1} | s_t, a_t) \end{aligned}$$

quindi:

$$\nabla_{\theta} \log \pi_{\theta}(\tau^{(i)}) = \nabla_{\theta} (\log p(s_1) + \sum_{t=1}^T \log \pi_{\theta}(a_t | s_t) + \log p(s_{t+1} | s_t, a_t))$$

Tutti i fattori che non dipendono direttamente da theta vanno a 0 quando derivo rispetto a loro, quindi posso ignorarli nella somma finale

$$\nabla_{\theta} \left[\cancel{\log p(\mathbf{s}_1)} + \sum_{t=1}^T \log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) + \cancel{\log p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)} \right]$$

Quindi rimettendo tutti i pezzi alla fine ottengo:

$$\nabla_{\theta} J(\theta) \approx \frac{1}{m} \sum_{i=1}^m \nabla_{\theta} \left(\sum_{t=1}^T \log \pi_{\theta}(a_t | s_t) * r(\tau^{(i)}) \right)$$

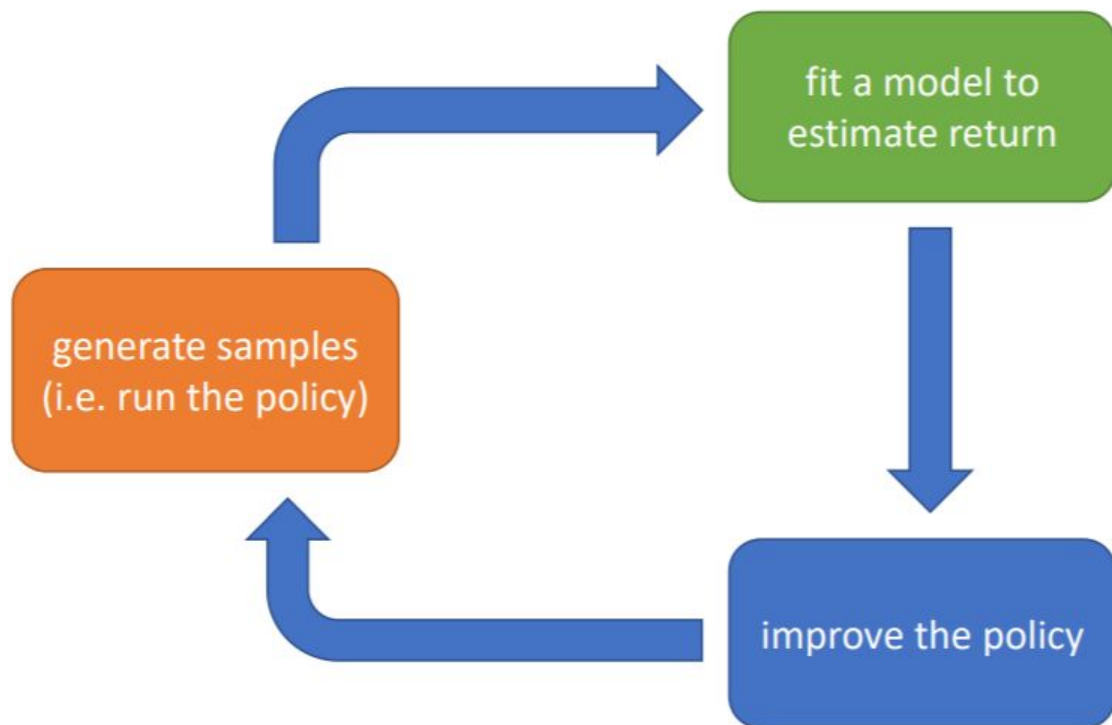
Il gradiente di una somma equivale alle somme dei gradienti, per questo lo possiamo portare dentro. Inoltre $r(\tau^{(i)})$ non è altro che la sommatoria dei reward per cui la formula diventa:

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^T r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$$

ps: notare che il gradiente tiene conto delle probabilità delle traiettorie pesandole con il loro livello di bontà (reward cumulativo). Il parametro influenza solo il primo fattore mentre il livello di bontà è lo stesso.

Questo procedimento può essere riassunto così:



Questo è esattamente l' algoritmo:

REINFORCE

REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run the policy)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i) \right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i) \right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Implementato in questo modo l'algoritmo è soggetto ad una altissima variabilità dei risultati. Per cui è necessario apportare 2 ottimizzazioni per stabilizzare il risultato e che non comportano l'aggiunta di bias e non necessitano di iper-parametri aggiuntivi.

Reward to Go:


Attualmente noi consideriamo il reward associato ad un'azione in uno stato in modo assoluto, ovvero indipendentemente da quando quell'azione è stata eseguita all' interno dell' episodio che ha generato quel reward. Quello che vogliamo fare noi è eliminare dalla sommatoria tutti i reward precedenti all' azione che stiamo prendendo in considerazione

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^T r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

Causality: policy at time t' cannot affect reward at time t when $t < t'$

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \left(\sum_{t'=1}^T r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right)$$

A questo punto si nota facilmente che quando per ogni azione è basata in base al reward totale dell'episodio in cui viene eseguita, il che non è il massimo. Infatti io vorrei che il giudizio su una certa azione venisse dato in base alle conseguenze di tale azione, e non anche in base a ciò che è successo prima. Per realizzare questo ci basta cambiare gli indici:

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \left(\sum_{t'=t}^T r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right)$$


$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_{it} | s_{it}) \left(\sum_{t'=t}^T \gamma^{t'-t} r(s_{it'}, a_{it'}) \right).$$

Ma a questo punto noto che:

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \underbrace{\left(\sum_{t'=t}^T r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right)}_{\text{"reward to go"}}$$

$\hat{Q}_{i,t}$

$$Q^{\pi}(s, u) = \mathbb{E}[r_0 + r_1 + r_2 + \dots | s_0 = s, a_0 = a]$$

Quindi posso riscrivere il tutto come:

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \hat{Q}_{i,t}$$

Baselines

Si tratta di normalizzare il reward di una traiettoria utilizzando un valore medio.

Intuitivamente: non prendiamo totalmente il valore di reward di una traiettoria ma lo prendiamo in relazione al caso medio. In questo modo abbiamo comunque un indice di quale sia migliore delle altre ma manteniamo bassi i valori

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log \pi_{\theta}(\tau) [r(\tau) - b]$$

$$b = \frac{1}{N} \sum_{i=1}^N r(\tau)$$

Questa baseline può essere rappresentata con una rete neurale, in questo caso la funzione di loss sarà il mean-square-error rispetto al reward di quell'episodio (value di quello stato).

$$\phi_k = \arg \min_{\phi} \mathbb{E}_{s_t, \hat{R}_t \sim \pi_k} \left[\left(V_{\phi}(s_t) - \hat{R}_t \right)^2 \right]$$

$$b(s_t) = \mathbb{E} [r_t + r_{t+1} + r_{t+2} + \dots + r_{H-1}] = V^{\pi}(s_t)$$

→ Increase logprob of action proportionally to how much its returns are better than the expected return under the current policy

Il reward-to-go trasforma $R(s,a)$ in $Q(s,a)$ e la baseline si può approssimare in $V(s)$.

Quindi con queste ottimizzazioni si passa dal prendere in considerazione il reward generale dell'episodio, all' agire sul valore del singolo stato (comunque in fase di training la q-function e la V-function le calcolo tramite i valori raccolti nei batch).

La versione finale della formula è quindi:

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_{it}|s_{it}) \left(\left(\sum_{t'=t}^T \gamma^{t'-t} r(s_{it'}, a_{it'}) \right) - V_{\phi}^{\pi}(s_{it}) \right)$$

dove:

$$V_{\phi}^{\pi}(s_t) \approx \sum_{t'=t}^T \mathbb{E}_{\pi_{\theta}} [r(s_{t'}, a_{t'}) | s_t],$$