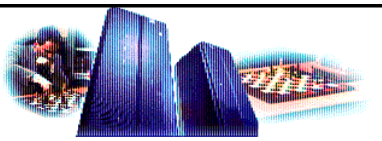
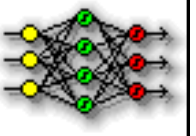
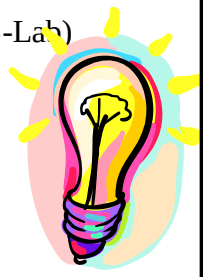




Sistemi Intelligenti Reinforcement Learning: l'apprendimento degli agenti in setting non associativo

Alberto Borghese

Università degli Studi di Milano
Laboratorio di Sistemi Intelligenti Applicati (AIS-Lab)
Dipartimento di Informatica
alberto.borghese@unimi.it



Riassunto

- Gli agenti ed il Reinforcement Learning
- Apprendimento di azioni
- La Value function

L'agente

- Inizialmente l'attenzione era concentrata sulla progettazione dei sistemi di "controllo". Valutazione, sintesi…
- L'intelligenza artificiale e la "computational intelligence" hanno consentito di spostare l'attenzione sull'apprendimento delle strategie di controllo e più in generale di comportamento.
- Macchine dotate di meccanismi (algoritmi, SW), per apprendere.
- RPA – Robot Process Automation

Why agents are important?

Agente (software): essere software che svolge servizi per conto di un altro programma, solitamente in modo automatico ed invisibile. Tali software vengono anche detti agenti intelligenti

"They are seen as a natural metaphor for conceptualising and building a wide range of complex computer systems (the world contains many passive objects, but it also contains very many *active* components as well);

They cut across a wide range of different technology and application areas, including telecoms, human-computer interfaces, distributed systems, WEB and so on;

They are seen as a natural development in the search for ever-more powerful abstractions with which to build computer systems."

How agents solve a problem

Formulate a problem. Through analysis. State, action, indentification.

Solve the problem (by searching).

Implement the solution (execute).

Evaluate the implemented solution.

- Success or fail? Adequate or not adequate?
- How much adequate? How to measure the success or failure of the performance?
- Optimization of the performance to create better agents.

- Solve a problem = achieve a given goal (= reach a final state or avoid certain states)

- An agent can examine different sequences of actions (deterministic or stochastic response by the environment) and search the best sequence.

Agente

- Può scegliere un'azione sull'ambiente tra un insieme continuo o discreto.

- L'azione dipende dalla situazione. La situazione è riassunta nello **stato** del sistema.

- L'agente monitora continuamente l'ambiente (input); l'ambiente modifica continuamente lo stato.

- La scelta dell'azione è non banale e richiede un certo grado di "intelligenza".

- L'agente ha una memoria "intelligente". Non può tenere in memoria tutto quanto successo nel passato.

Reinforcement learning

Spesso si ha a disposizione solamente un'informazione qualitativa, **detta reward**, (a volte binaria, giusto/sbagliato successo/fallimento), puntuale.

Apprendimento con rinforzo è un apprendimento attraverso reward.

*L'informazione disponibile si chiama **segnale di rinforzo.** Non dà alcuna informazione su come aggiornare il comportamento dell'agente (e.g. i pesi). Non è possibile definire una funzione costo o un gradiente.*

Obbiettivo: creare degli agenti "intelligenti" che abbiano una "machinery" per apprendere dalla loro esperienza.

Exploration vs Exploitation

Esplorazione (exploration) dello spazio delle azioni per scoprire le azioni migliori. Un agente che esplora solamente raramente troverà una buona soluzione.

Le azioni migliori vengono scelte ripetutamente (exploitation) perchè garantiscono ricompensa (reward). Se un agente non esplora nuove soluzioni potrebbe venire surclassato da nuovi agenti più dinamici.

Occorre non interrompere l'esplorazione.

Occorre un approccio statistico per valutare le bontà delle azioni.

Exploration ed exploitation vanno bilanciate. Come?

Qual è il comportamento ottimale?

Esempi

Un giocatore di scacchi. Per ogni mossa ha informazione sulle configurazioni di pezzi che può creare e sulle possibili contro-mosse dell'avversario.

Una gazzella in 6 ore impara ad alzarsi e correre a 40km/h.

Come fa un robot veramente autonomo ad imparare a muoversi in una stanza per uscirne? (cf. competizione Robocare@home).

Come impostare i parametri di una raffineria (pressione petrolio, portata….) in tempo reale, in modo da ottenere il massimo rendimento o la massima qualità?

Riassunto

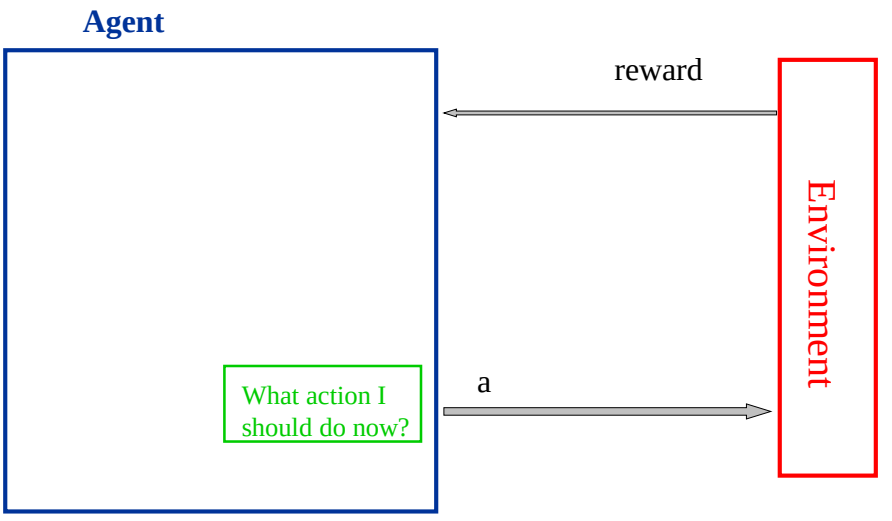
- Gli agenti ed il Reinforcement Learning
- Apprendimento di azioni
- La Value function

Esempio

Dove posizionare gli avvertimenti pubblicitari? Quanto valgono?

Ogni click sull'avvertimento è un rinforzo positivo.
Vengono provate diverse posizioni.
Si cerca di massimizzare il reward, cioè il numero di click.

Schematic diagram of an agent



Il problema del "n-Armed bandit"

Situazione iniziale sempre uguale.

Scelta tra n azioni (azione + reward esaurisce l'episodio)

La richiesta di scegliere viene ripetuta più volte nel tempo.

La ricompensa è stocastica (e.g. slot machine).

Obbiettivo: viene massimizzata la ricompensa a lungo termine.

Soluzione possibile: selezionare l'azione che fornisce la massima ricompensa a lungo termine.

Come?

Slot machine stocastica

Il reward della slot machine è completamente definito dalla densità di probabilità associata alla macchina.

Si suppone la densità di probabilità costante nel tempo.

Per semplicità si suppone che la densità di probabilità sia descrivibile da una funzione analitica, ad esempio una Gaussiana. In questo caso la densità di probabilità è definita dai parametri della Gaussiana: media e standard deviation.

Che cosa rappresenta la densità di probabilità?

Come massimizzare la ricompensa

Consento all'agente di avere memoria.

Memorizzo il valore associato alle diverse azioni.

Posso ad un certo punto scegliere SEMPRE l'azione che mi ha dato la RICOMPENSA MAGGIORE.

GREEDY ACTION (Greedy = Goloso).

EXPLOITING KNOWLEDGE.

Perché dovremmo scegliere un'azione che non appare la migliore (NON GREEDY)?

Exploration

Perchè esploriamo soluzioni diverse?

La ricompensa non è deterministica. Potremmo ottenere di più con altre azioni.

Quello che conta non è la ricompensa istantanea ma la somma delle ricompense ottenute.

Occorre quindi mantenere un istinto ad esplorare azioni diverse.

Il bilanciamento di "exploration" e di "exploitation" è un compito complesso.

Riassunto

- Gli agenti ed il Reinforcement Learning
- Apprendimento di azioni
- La Value function

La Value Function e la scelta delle azioni

Posso selezionare n-azioni: a = $a_1 \ldots a_n$.

Ciascuna di queste azioni ha un suo valore medio: $Q^*(a_k) = \mu_k$. Long-time reward.

Ciascuna di questa azioni ha anche una stima del suo valore a lungo termine (VALUE):
Supponiamo questa stima funzione del tempo t: $Q_t(a_k)$ è la funzione valore.

$Q_t(a_k)$ -> $Q^*(a_k)$

Voglio scegliere a_k che massimizza: $Q(a_k)$.

In caso di exploitation di a_k, posso stimare il "value" all'istante t, come:

$$Q_t(a_k) = \frac{r_1 + r_2 + \ldots + r_{N(a_k)}}{N(a_k)}$$

Dove r_j è il reward per avere scelto a_k una j-esima volta

Caratteristiche della Value Function

Value function calcolata come media campionaria:

$Q_t(a_k)$ -> $Q^*(a_k)$ per t -> ∞
$Q_t(a_k)$ = 0 t = 0. Nessuna stima disponibile.

Prima possibilità di selezione dell'azione che dà all'istante t, la massima VALUE FUNCTION stimata:

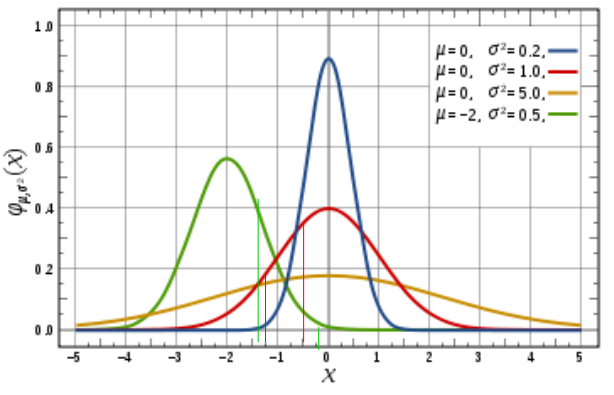
k : $Q_t(a_k) > Q_t(a_j)$ ∀j ≠ k. $\quad a^* : Q_t(a^*) = \max_{a_k} \{Q_t(a_k)\}$

Così viene EXPLOITED la conoscenza accumulata, è una politica GREEDY.

Non vengono esplorate soluzioni alternative.

Come si può formalizzare un'alternativa?

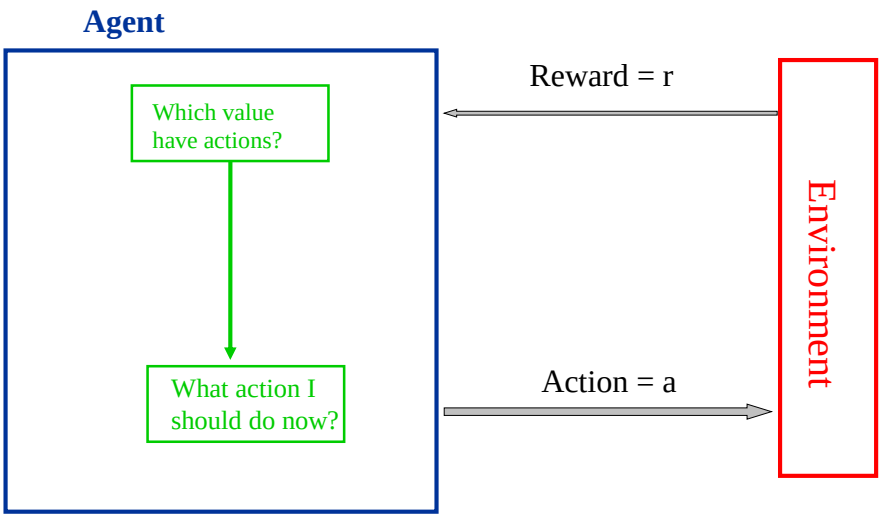
Esempio di Q(.) non adeguata



Running average of red curve: -0.5+(-1.25) = -1,75 / 2 = -0,875 = Q_2(red)
Running average of green curve: -1.3+(-0.2) = -1,5 / 2 = -0,75 = Q_2(green)

Less reward using the red action!

Schematic diagram of an agent



Exploitation and Exploration

Suppongo che con probabilità, ε, viene scelta un'azione diversa.

Questa azione viene scelta con probabilità uniforme tra le n possibili azioni a disposizione (**ε–Greedy method**).

$Q_t(a_k)$ -> $Q^*(a_k)$ t -> ∞

Near-gready action selection. Come funziona?

$$a^* : Q_t(a^*) = \max_{a_k} \{Q_t(a_k)\} \qquad P = 1 - \varepsilon$$

$$a \neq a^* \qquad P = \varepsilon$$

Uniforme sulle altre a_k

Beyond ε-greedy: pursuit methods

Dopo ogni episodio, la probabilità di scegliere un'azione viene aggiornata:

- L'azione associata alla value function migliore aumenta la probabilità di essere prescelta.
- La probabilità di scegliere le altre azioni viene decrementata.

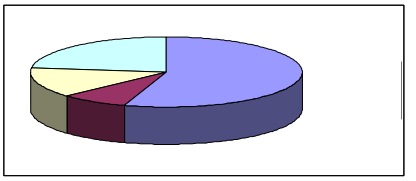
$$\pi_{t+1}(a^*_{t+1}) = \pi_t(a^*_{t+1}) + \beta[1 - \pi_t(a^*_{t+1})]$$
$$\pi_{t+1}(a) = \pi_t(a) + \beta[0 - \pi_t(a)] \quad for\, a \neq a^*_{t+1}$$

The preference for an action is always "**pursuing**" (inseguendo) the action that is greedy according to the current action-value estimate.

Approccio generale della roulette

I valori vengono selezionati proporzionalmente ad una misura di performance nel momento attuale. Migliore essa è e più alta è la probabilità di selezione

1. Si immagini una roulette dove siano sistemate tutte le possibili azioni, con area unitaria.
2. La dimensione della sezione nella roulette è proporzionale al valore di misura di performance di ciascuna azione.
3. La pallina viene lanciata all'interno della roulette e l'azione in corrispondenza della quale si ferma è quella selezionata



In questo esempio ci sono 4 possibili azioni.

Esempio: 10-armed testbed

n-armed bandit problem: n = 10: $a = a_1, a_2 \ldots, a_k, .. a_{10}$.

Per ogni task (esperimento), eseguo 1000 volte la scelta di ciascuna azione:

$t = t_1, t_2 \ldots.. t_{1000}$.

$a = a(t_1), a(t_2) \ldots.. a(t_{1000})$

$r = r(a(t_1)), r(a(t_2)) \ldots.. r(a(t_{1000}))$

$r(a_k)$ viene selezionato in modo random da una distribuzione Gaussiana con **media μ_k, diversa per le diverse azioni**, ma costante per tutto il task, e **varianza 1**. $\mu_k = Q^*(a_k)$

Misuro per ogni istante di tempo, t:

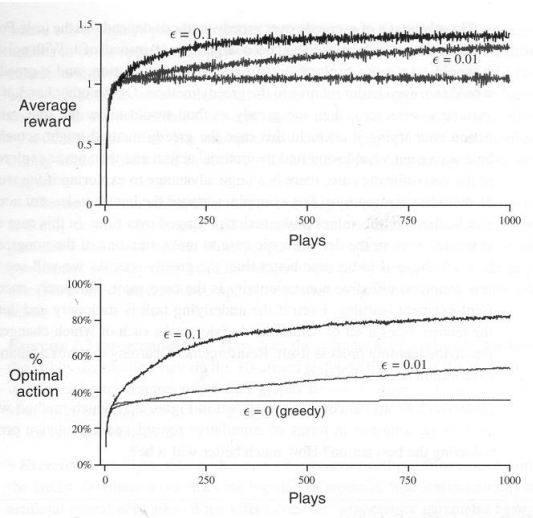
Il reward dell'azione (in questo caso viene dato un reward $\neq$ 0 per ogni azione)

Calcolo la ricompensa totale (Value Function).

Valuta la performance dopo le 1000 giocate di ogni task.

Quanto vale μ_k? Per ogni task (esperimento) vario **μ_k estraendolo da una distribuzione Gaussiana con media = 0 e varianza = 1**.

Risultati



Media su 2000 task, ciascuno di 1000 giocate (azioni, plays)

Si potrebbe implementare una politica ε-greedy variabile: ε↓ #Plays↑

Domande

Supponiamo che la distribuzione da cui si sceglie il valore medio del reward abbia varianza nulla. Quale metodo funziona meglio: Greedy o ε-Greedy?

Supponiamo che la distribuzione da cui si scegli il valore medio del reward abbia varianza maggiore (e.g. = 10). Cosa succede? Quale metodo si comporterebbe meglio?

In quali altre condizioni sarebbe utile avere esplorazione?

Problemi di memoria

$$Q_t(a_k) = \frac{r_1 + r_2 + ... + r_{N(a_k)}}{N(a_k)}$$

Occorre scegliere un algoritmo che calcoli $Q_t(.)$ con un piccolo carico computazionale e di memoria.

Supponiamo di Exploit l'azione a_k. Calcoliamo la media dei reward dopo N reward e la chiamiamo $Q_N(a_k)$. $Q_N(a_k)$ coinciderà con la media delle prime $N(a_k)$ ricompense:

$$Q_t(a_k) = \frac{r_1 + r_2 + ... + r_{N(a_k)}}{N(a_k)}$$

Scegliendo ancora a_k, otteniamo il seguente valore di Q dopo N+1 reward:

$$Q_{N+1}(a_k) = \frac{r_1 + r_2 + ... + r_N + r_{N+1}}{N+1}$$

Determinazione ricorsiva di Q_N

$$Q_N(a_k) = \frac{r_1 + r_2 + ... + r_N}{N} \qquad Q_{N+1}(a_k) = \frac{r_1 + r_2 + ... + r_N + r_{N+1}}{N+1}$$

$$Q_{N+1} = \frac{r_1 + r_2 + ... + r_N}{N+1} + \frac{r_{N+1}}{N+1} = \frac{Q_N N}{N+1} + \frac{r_{N+1}}{N+1} =$$

$$\frac{Q_N(N+1-1)}{N+1} + \frac{r_{N+1}}{N+1} = \frac{Q_N(N+1) - Q_N}{N+1} + \frac{r_{N+1}}{N+1} =>$$

$$Q_{N+1} = Q_N - \frac{Q_N}{N+1} + \frac{r_{N+1}}{N+1}$$

Dipende da N + 1

Non dipende da N + 1

$Q_1 = r_1(a_k)$ Q_0 arbitraria

Osservazioni su Q_N

$$Q_{N+1} = Q_N - \frac{Q_N}{N+1} + \frac{r_{N+1}}{N+1} = Q_N + \alpha[r_{N+1} - Q_N] \qquad \alpha = 1/(N+1)$$

Occupazione di memoria minima: Solo Q_N e N.
NB N è il numero di volte in cui è stata scelta a_j, non è necessariamente coincidente con il tempo t!

Questa forma è la base del RL. La sua forma generale è:

NewEstimate = OldEstimate + StepSize [Target – OldEstimate]
*NewEstimate = OldEstimate + StepSize * Error.*

StepSize = α = 1/(N+1)

Osservazioni su Q_N

$$Q_{N+1} = Q_N - \frac{Q_N}{N+1} + \frac{r_{N+1}}{N+1} = Q_N + \alpha\left[r_{N+1} - Q_N\right] \qquad \alpha = 1/(N+1)$$

Questa forma è la base del RL. La sua forma generale è:
NewEstimate = OldEstimate + StepSize [Target – OldEstimate]
*NewEstimate = OldEstimate + StepSize * Error.*

StepSize = α = 1/(N+1)

Si può anche scrivere come:

$$Q_{N+1} = \alpha r_{N+1} + (1-\alpha) Q_N \qquad \alpha = 1/(N+1)$$

Dove α pesa il bilanciamento tra ''innovazione'' e ''tradizione''

Esempio

$$Q_{N+1} = Q_N - \frac{Q_N}{N+1} + \frac{r_{N+1}}{N+1} = Q_N + \frac{1}{N+1}\left(r_{N+1} - Q_N\right)$$

$r_1 = 2$, $r_2 = 3$; $r_3 = 7$; $r_4 = 2$
$Q_1 = 2$; $Q_2 = 2{,}5$ $Q_3 = 4$ $Q_3 = 3{,}75$

$$Q_3 = Q_2 + \frac{1}{3}\left(r_3 - Q_2\right)$$

$Q_3 = 2{,}5 + 1/3\ (7 - 2{,}5) = 2{,}5 + 1{,}5 = 4{,}0$

Caso stazionario

$$Q_{N+1} = \frac{r_1 + r_2 + ... + r_N + r_{N+1}}{N+1}$$

$$Q_{k+1} = \sum_{i=1}^{k+1} \frac{r_i}{N_{k+1}}$$

Il peso di ciascun campione è pari a 1/(N+1)

$$Q_{N+1} = Q_N - \frac{Q_N}{N+1} + \frac{r_{N+1}}{N+1}$$

Ogni nuovo campione viene pesato con 1/(N+1)

Il peso segue un'iperbole.

$$Q_{N+1} = Q_N + \alpha\left[r_{N+1} - Q_N\right]$$

Peso decrescente ai nuovi campioni

Cosa succede se il task è non stazionario?
Cosa succede se voglio pesare i diversi campioni in modo derescente nel tempo?

Osservazione:
Se $\alpha = 0$, conta solo il reward non viene modificato.
Se $\alpha = 1$, Q_{N+1} assume il valore di Q_N (dimentica tutto il passato).

Caso non stazionario

$$Q_N = \alpha r_N + (1-\alpha) Q_{N-1}$$

Al tempo k, ottengo r_k
Q_0 è il valore a cui è inizializzata Q

Suppongo α = cost ➔ $\alpha_k = \alpha \ \forall k \qquad 0 \le \alpha \le 1$

$$Q_N = Q_{N-1} + \alpha\left[r_N - Q_{N-1}\right] =$$
$$= \alpha r_N + (1-\alpha) Q_{N-1} =$$
$$= \alpha r_N + (1-\alpha)[\alpha r_{N-1} + (1-\alpha) Q_{N-2}] =$$
$$\alpha r_N + (1-\alpha)\alpha r_{N-1} + (1-\alpha)^2 Q_{N-2}] =$$

$$= (1-\alpha)^0 r_N + (1-\alpha)^1 \alpha r_{N-1} + (1-\alpha)^2 \alpha r_{N-2} + ... + (1-\alpha)^{N-1} \alpha r_1 + (1-\alpha)^N Q_0 \Rightarrow$$

Caso non stazionario

$$Q_N = \alpha r_N + (1-\alpha) Q_{N-1}$$

Al passo N, ottengo r_k
Q_0 è il valore a cui è inizializzata Q

Suppongo α = cost ➔ $\alpha_k = \alpha \ \forall k$ $\quad 0 \le \alpha \le 1$

$$\alpha r_N + (1-\alpha)\alpha r_{N-1} + (1-\alpha)^2 \alpha r_{N-2} + \ldots\ldots + (1-\alpha)^{N-1} \alpha r_1 + (1-\alpha)^N Q_0 =$$
$$(1-\alpha)^0 \alpha r_N + (1-\alpha)^1 \alpha r_{N-1} + (1-\alpha)^2 \alpha r_{N-2} + \ldots\ldots + (1-\alpha)^{N-1} \alpha r_1 + (1-\alpha)^N Q_0$$

Per r_i $\quad 1 \le i \le N$ $\quad r_1$ -> r_N
Per $(1 - \alpha)^{N-i}$ $\quad 1 \le i \le N$ $\quad (1 - \alpha)^{N-1}$ -> $(1 - \alpha)^0$

$$Q_{N+1} = \sum_{i=1}^{N} \alpha (1-\alpha)^{N-i} r_i + (1-\alpha)^N Q_0$$

Osservazioni

$$Q_{N+1} = (1-\alpha)^N Q_0 + \sum_{i=1}^{N} \alpha (1-\alpha)^{N-i} r_i = (1-\alpha)^N Q_0 + \sum_{i=1}^{N} w_i r_i$$

$$w_i = \alpha (1-\alpha)^{N-i} \qquad \alpha < 1$$

I reward **non** sono pesati tutti allo stesso modo: weighted average.

Il peso di ciascun campione decresce esponenzialmente a partire da i = N (tempo presente) fino a i = 0 (tempo iniziale), secondo:

$$\alpha (1-\alpha)^{N-i} \qquad \alpha < 1$$

Exponential, recency-weighted average.

Somma dei pesi dei reward è unitaria

$$Q_{N+1} = (1-\alpha)^N Q_0 + \sum_{i=1}^{N} \alpha(1-\alpha)^{N-i} r_i \qquad i=1 \to (1-\alpha)^{N-1} \qquad i=N \to (1-\alpha)^0$$

Riscrivo considerando solamente i coefficienti.

Faccio partire la sommatoria da 0 a N-1 e sostituisco i a N-i.
Aggiungo e sottraggo il termine $(1-\alpha)^N$ dentro la sommatoria e

$$1 = \alpha\left(\sum_{i=0}^{N-1}(1-\alpha)^i + (1-\alpha)^N - (1-\alpha)^N\right) + (1-\alpha)^N =$$

$$= \alpha\left(\sum_{i=0}^{N}(1-\alpha)^i - (1-\alpha)^N\right) + (1-\alpha)^N =$$

$$\alpha\left[\frac{(1-\alpha)^{N+1}-1}{(1-\alpha)-1} - (1-\alpha)^N\right] + (1-\alpha)^N =$$

Somma dei pesi dei reward è unitaria

$$Q_{N+1} = (1-\alpha)^N Q_0 + \sum_{i=1}^{N} \alpha(1-\alpha)^{N-i} r_i \qquad i=1 \to (1-\alpha)^{N-1} \qquad i=N \to (1-\alpha)^0$$

Riscrivo considerando solamente i coefficienti.

$$1 = \alpha\left[\frac{(1-\alpha)^{N+1}-1}{(-\alpha)}\right] - \alpha(1-\alpha)^N + (1-\alpha)^N =$$

$$-(1-\alpha)^{N+1} + 1 - \alpha(1-\alpha)^N + (1-\alpha)^N = \qquad \text{Raccolgo } (1-\alpha)^N$$

$$-(1-\alpha)^{N+1} + 1 + (1-\alpha)^N (1-\alpha)^N \; cvd$$

Condizioni iniziali

$$Q_{N+1} = (1-\alpha)^N Q_0 + \sum_{i=1}^{N} \alpha(1-\alpha)^{N-i} r_i$$

Metodi ad $\alpha = 1/N_{k+1}$, Q_0 non viene utilizzato se non al primo passo, viene poi sostituito da Q_1.

Metodi ad α costante, Q_0, conta sempre meno, ma la polarizzazione è permanente ($Q_0 = 0$).

Q_0 può essere utilizzato per fornire della conoscenza a-priori o per favorire l'esplorazione.

Come posso gestire una situazione in cui la slot machine cambia improvvisamente la sua densità di probabilità di reward?

Pseudo-codice per il calcolo di Q_k.

```
####### 1) Definizione delle variabili:
N_scelte = m; eps_gready = 0.1;          // epsilon dipende dal grado di greedy che voglio dare all'agente
          ## Variabili dell'agente
A = {1, 2,    m};                        // Azioni possibili
Q = {Q1, Q2 ….. Qm} = 0;                 // Value function per ogni azione
N_azioni = {1, 2, …… m}                  // Numero di volte in cui è scelta l'azione j (e collezionato il reward associato).
          ## Variabili dell'ambiente. Date nella simulazione, misurate nell'ambiente nella realtà
// Inizializzo i parametri della distribuzione (stazionaria) dei reward per ogni azione
meanReward = [mean_1, mean_2, ….. mean_m]; stdReward = [std_1, std_2, ….. std_m];

###### 2) Ciclo di funzionamento
while (true)
{         eps = rand_unif([0 1]);                    // Per politica epsilon-greedy
    // Exploitation
          [a_attuale Q_attuale] = SearchMax(Q);      // Cerca l'azione ottima secondo Q
    // Exploration: se eps < eps_greedy, allora exploration
          if (eps < epsilon_gready)
          // Devo trovare un'azione diversa da a_attuale -> a_ref
          {         trovato = false; a_ref = a_attuale;
                    while (trovato == false)
                    {         a_attuale = rand_unif(A);
                              if (a_attuale != a_ref)
                              {  trovato = true; Q_attuale = Q(a_attuale);
                              }
                    }
          }
    // Eseguo l'azione a_attuale e misuro il reward ottenuto dalla slot machine
          r_attuale = rand_Gauss[meanReward(a_attuale), stdReward(a_attuale)];
    // Update i dati per l'azione a_attuale: il numero di azioni e la value function Q
          N_azioni(a_attuale)++;
          Q(a_attuale) = Q(a_attuale) + 1/[N_azioni(attuale)] * (r_attuale – Q(a_attuale));
}
```

Riassunto

- Gli agenti ed il Reinforcement Learning
- Gli elementi del RL
- Setting non associativo
- La Value function