Coupling planning and learning

Corrado Possieri

Machine and Reinforcement Learning in Control Applications

Introduction

Accoppiamo DP con apprendimento. Apprendiamo la funzione valore e stimiamo il modello.
Il vantaggio è quello che si può migliorare l'apprendimento rivisitando l'esperienza più volte.
L'esperianza smiulta permermette di velocizzare l'apprendimento.

Planning: model-based. Dato un modello trovo funzione valore(qualità) e policy ottima.

Learning: model free. Dato il reward e transizione stato azione successiva per aggiornare la funzione valore.

- The hearth of both methods is computing a value function
  - look ahead to future events;
  - compute a backed-up value;
  - update target.
- Can these methods be intermixed?

Queste tecniche possono essere combinate poiché i metodi sono basati sulla stima della funzione valore. In un caso si parte da un modello, dall'altro si parte dall'esperienza, ma stimano entrambi la funzione valore. La parte in comune è che si guardano gli eventi futuri: in un caso nel planning (matrice transizione e reward) per tenedere alla funzione valore ottima. Entrambi i metodi aggiornano le stime tramite l'eq. di Bellmann

La differenza tra planning e learning è che cosa mi implica l'aggiornamento: il primo è il modello, il secondo è l'esperienza. Sulla base di questo si stima la funzione valore.

Model, learning, and planning

- A model is anything the agent can use to predict the behavior of the environment

  distribution models: gives all possibilities and their probability;
  sample models: produce just one of the possibilities.

- Distribution models can be used to generate samples.

$$\text{Model} \longrightarrow \text{Simulated experience} \xrightarrow{\text{Backup}} \text{Values} \longrightarrow \text{Policy}$$

funzione qualità

- Learning can be used for planning.

Non utilizzo PI e VI, ma utilizzo il modello per generare campioni del modelllo.

Sample-based planning

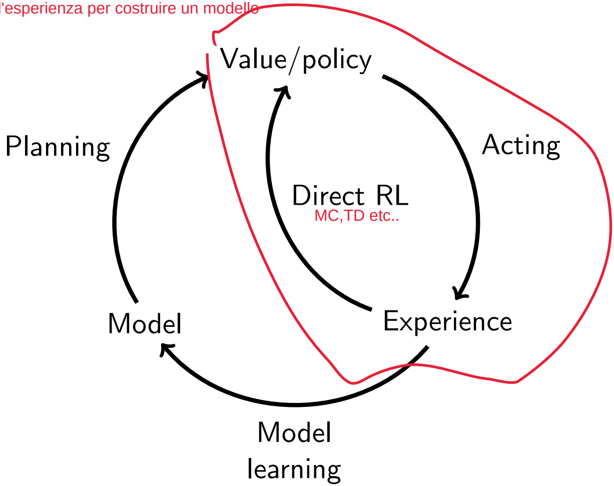
- Use the model only to generate samples.
- Sample experience from model

$$
\begin{aligned}
S_{t+1} &\sim p(S_{t+1}|S_t, A_t),\\
R_{t+1} &= r(S_t, A_t).
\end{aligned}
$$

- Apply model-free RL to samples.
- Often more efficient than planning.

Indirect reinforcement learning

Indiretto perché utilizzo l'esperienza per costruire un modello



FIno a che non ho sufficiente esperienza

Notes on indirect learning

- Indirect methods often make fuller use of a limited amount of experience.
- Direct methods are simpler and are not affected by biases.
- Model-based RL is only as good as the estimated model.
- When the model is inaccurate, planning will compute a suboptimal policy.

How to learn a model Model Learning

- In deterministic environments
  - in state S_t, take action A_t;
  - observe R_{t+1} and S_{t+1};
  - $p(S_{t+1}|S_t, A_t) \leftarrow 1$, $p(s|S_t, A_t) \leftarrow 0, \forall s \neq S_{t+1}$;
  - $r(S_t, A_t) \leftarrow R_{t+1}$.

- In MDP environments
  - observing the history
    - determining $p(s'|s, a)$ is a *density estimation* problem;
    - determining $r(s, a)$ is a *regression* problem.

Table lookup model

- Assuming $\mathcal{S}$ and $\mathcal{A}$ known
  - for and MDP we need to estimate P and R
    - use empirical samples

episodio diverso

Stima densità di probabilità
density estimation

$$\hat{P}_{s,s',a} = \frac{1}{N(s,a)} \sum_{k=1}^{K} \sum_{t=1}^{T_k} \mathbf{1}(s_{t+1}^k = s', s_t^k = s, a_t^k = a),$$

$$\hat{R}_{s,a} = \frac{1}{N(s,a)} \sum_{k=1}^{K} \sum_{t=1}^{T_k} r_t^k \mathbf{1}(s_t^k = s, a_t^k = a).$$

Dyna-Q

- Use deterministic model learning
- Use one-step Q-learning as planning method
    - randomly sample from state-action pairs that have been previously experienced;
    - the model returns the last-observed next state and next reward as its prediction.
- Use one-step Q-learning as direct RL method.
- Learning and planning differ only in the source of their experience.

Applуco Q-learning

Ricavati dal modello

$$S_{t+1} = p(S' | S, A)$$

$$R_{t+1} = \dot{r}(S, A)$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t)\right)$$

Dyna-Q algorithm

Si emula la TD poiché nelle TD cerco di propagare le informazioni all'indietro mentre qui lo si effettua in altro modo: campiono randocamente coppie stato azioni già visitato poco tempo prima.
Non appena si ha una nuova esperienza, si pesce randocimante fea quelle precedenti e si aggiorna in base alla nuova funzione qualità. Non si ha il vincolo di lavorare su un singolo episodio.

Dyna-Q algorithm

La problematica è che aggiorno le coppia stato azione in maniera casuale.Ciò causa un apprendimento più lento.

Input: $\alpha > 0$, $\varepsilon > 0$
Output: q_*, π_*

Initialization

$Q(s,a) \leftarrow \text{arbitrary}, \forall a \in \mathcal{A}(s), \forall s \in \mathcal{S}; Q(\text{terminal}, \cdot) \leftarrow 0$
$\text{Model}(s,a) \leftarrow \emptyset, \forall a \in \mathcal{A}(s), \forall s \in \mathcal{S}$

Loop

$S \leftarrow$ current state
$A \leftarrow \varepsilon - \text{greedy}(S, Q)$
take action A and observe R, S'
$Q(S,A) \leftarrow Q(S,A) + \alpha(R + \gamma \max_a Q(S',a) - Q(S,A))$ Aggiornamento del Q-learning
$\text{Model}(s,a) \leftarrow S', R$
repeat
$(S,A) \leftarrow$random previously experienced pair
$R, S' \leftarrow \text{Model}(S,A)$
$Q(S,A) \leftarrow Q(S,A) + \alpha(R + \gamma \max_a Q(S',a) - Q(S,A))$
until next sample is available

Notes on Dyna-Q

- Learns much faster in deterministic environments.
- If the environment changes, it can adapt.
- However, the formerly correct policy may not reveal improvements.
- The planning process is likely to compute a suboptimal policy.
- Exploration/exploitation conflict in a planning context
  - exploration: try to improve the model;
  - exploitation: take the best possible action according to the available model.

Dyna-Q+

Modifica la stima della funzione di qualità. modifica il reward in base a quanto tempo è passato prima che sceglo quell'azione: si aggiunge un bias

- Keep track for each state–action pair of how many time steps have elapsed since last visit.
- The more time that has elapsed, the more is like that the model is incorrect.
- Encourage behavior that tests long-untried actions
  - the modeled reward for a transition is r;
  - the transition has not been tried in τ time steps;
  - planning assumes that the reward is $r + \kappa\sqrt{\tau}$; Reward vero+bias*sqrt(tempo trascorso)
  - an alternative is to select action as that maximizing

e.greedy sulla funzione estesa

$$Q(S_t, a) + \kappa\sqrt{\tau(S_t, a)}.$$

Prioritized sweeping

- Planning is more efficient if simulated transitions and updates are focused on particular state–action pairs. Quelle che devono essere cambiate di più.
- If simulated transitions are generated uniformly, then many wasteful updates are made.
- In general, we want to work back from any state whose value has changed
  - the predecessor pairs of those that have changed are more likely to also change;
  - prioritize the updates according to a measure of their urgency;
  - needs an inverse model.

    Da coppia stato azione, mi dice i predecessori cappie stato azioni

Prioritized sweeping algorithm

Applico le tecniche di pianificazione in maniera strutturata.

Prioritized sweeping algorithm

Input: $\alpha > 0$, $\varepsilon > 0$, threshold $\theta > 0$ soglia che determina le coppie stato azione che hanno avuto un apprendimento importante
Output: q_*, π_*

Initialization

$Q(s,a) \leftarrow$ arbitrary, $\forall a \in \mathcal{A}(s), \forall s \in \mathcal{S}$; $Q(\text{terminal}, \cdot) \leftarrow 0$
Model$(s,a) \leftarrow \emptyset, \forall a \in \mathcal{A}(s), \forall s \in \mathcal{S}$
PQueue $\leftarrow \emptyset$ Una coda di priorità; da una seq, di elementi e una priorità di selezione degli elementi

Loop

$S \leftarrow$ current state
$A \leftarrow \varepsilon$-greedy(S, Q)
take action A and observe R, S'
Model$(s,a) \leftarrow S', R$
$P \leftarrow |R + \gamma \max_a Q(S', a) - Q(S, A)|$

Indice di priorità

Determino l'importanza dellìelemento di apprendimento; è l'ampiezza dell'aggiornamento. Se è grande la modifica è importante, al contrario se piccolo questo evento non è molto influente per l'apprendimento.

if $P > \theta$ **then**
insert S, A into PQueue with priority P

Pianificazione

while PQueue $\neq \emptyset$ **do**
$(S, A) \leftarrow$ first(PQueue) a priorità maggiore
$R, S' \leftarrow$ Model(S, A)
$Q(S,A) \leftarrow Q(S,A) + \alpha(R + \gamma \max_a Q(S', a) - Q(S,A))$

Modello inverso

for all $\bar{S}, \bar{A}$ predicted to lead to S **do** aggiorno coppie stato azione che mi hanno portato in quello stato
$\bar{R}, \bar{S}' \leftarrow$ Model$(\bar{S}, \bar{A})$
$\bar{P} \leftarrow |\bar{R} + \gamma \max_a Q(\bar{S}', a) - Q(\bar{S}, \bar{A})|$
if $\bar{P} > \theta$ **then**
insert $\bar{S}, \bar{A}$ into PQueue with priority $\bar{P}$

Priotirizzo le coppia (s,a) con priorità P che hanno subito un evento di apprendimento. Più P è grande più la coppia S,A ha bisongo di essere aggiornata.

Verifico la priorità anche per gli stati e le azioni precedenti

Trajectory sampling

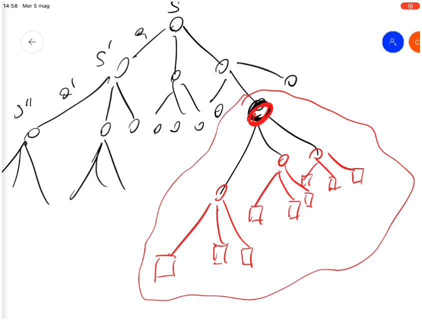
Proabilità di essere uno stato con la policy corrente

- Distribute updates according to the on-policy distribution
  - distribution observed when following the current policy;
  - simulate individual trajectories and perform updates at the state encountered along the way.
- States actually visited are updated more often.
- Uninteresting parts of the space are ignored.
- This is the same that happens in real time DP
  - can find a policy that is optimal on the relevant states without visiting every state infinitely often.

Planning at decision time

Non si risolve tutto il problema di ottimizzazione per tutti gli stati,ma ci si cocncentra su dove si è in quel momento. Invece di risolvere l'eq. di Bellman, si vedono tutti i successori dallo stato corrente; si trascurano i predecessori.

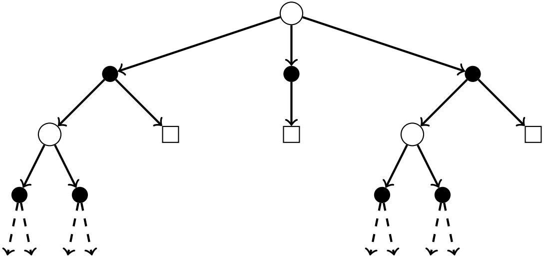
- Planning executed at current state.
- The values and policy are specific to the current state.
- The values and policy created are discarded after being used.
- Useful in applications in which fast responses are not required



Mi concentro solo sul sottoalbero e vado in avanti. Faccio una scelta per i successori e per il principip di ottimalità la policy ottimalità la soluzione che verrà fuori sarà sempre ottima.

Heuristic search

- Go through the tree of possible continuations.
- Use a model of the sub-MDP starting from *now*.
- Focused on state/actions that immediately follow.
- Build a search tree with S_t at its root.



Rollout algorithms

- Heuristic search guided by MC simulation.
- Simulate episodes from now with the model.
- Average returns of simulated trajectories that start with each action and then follow rollout policy
  - simulate K episodes following first action a and then policy π

$$S_t, a, R_{t+1}^k, S_{t+1}^k, A_{t+1}^k, R_{t+2}^k, \ldots, R_T^k, S_T^k;$$

  - evaluate actions by mean return

$$Q(S_t, a) = \frac{1}{K}\sum_{k=1}^{K} G_t^k;$$

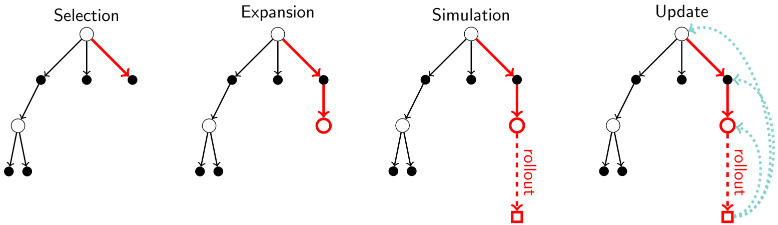
  - take action that maximize Q

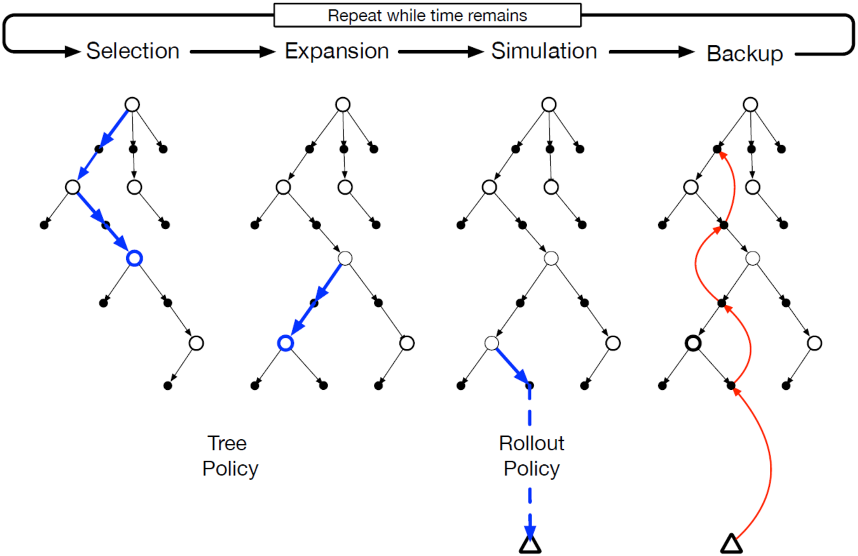
$$A_t = \arg\max_a Q(S_t, a).$$

Monte Carlo tree search

- Use rollout method by modifying policy.
- Record the values of Q in the search tree.
- In the tree, we pick actions to maximize Q (*e.g.*, ε-greedy).
- Outside the tree use a default policy.
- MC control applied to simulated experience.
- Expand the part of the tree that looks promising.
- MC can be substituted by TD.

Monte Carlo tree search logic




Repeat while time remains
Selection
Expansion
Simulation
Backup
Tree
Policy
Rollout
Policy