Heuristic Approach: Exploration Function

An alternative to GLIE speeds up convergence by promoting underexplored x, y pairs by changing (??) into

$$\widehat{U}(x) = r(x) + \gamma \max_{y \in Y} f\left(\sum_{x' \in X} p(x' \mid x, y)\widehat{U}(x'), N(x, y)\right)$$

where f is an **exploration function** and $N(x, y)$ is the number of times action y has been taken in state x.

We want f to give the utility expectation (w.r.t. the estimated probability p) only for pairs x, y such that y has been already tried on x a sufficient number of times.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Heuristic Approach: Exploration Function (cont'd)

For other (under-explored) pairs x, y, it should yield an *optimistic* utility value attracting the agent. This can be done with

$$f(u, v) = \begin{cases} \max R \text{ if } v < m \\ u \text{ otherwise} \end{cases}$$

Recall that R is bounded so $\max R$ is finite.

Observe that the GLIE approach changes (randomizes) a policy that has been computed through policy iteration or value iteration.

On the other hand, the exploration function is used only with value iteration because it is used to modify (??).

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Adaptive Dynamic Programming

An **adaptive dynamic programming** (ADP) agent is one that uses its estimated model p of P to calculate the state utilities and the policy by value or policy iteration.

(Both the iteration procedures are a form of dynamic programming.)

p is re-estimated on each new sample (x', x, y) and so the iteration procedures are also repeated on each time step.

This may be computationally costly.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Direct Utility Estimation

The costly procedures can be avoided by **direct utility estimation** (DUE)

- instead of estimating P from samples of x', x, y and then computing $U(x)$, $x \in X$ using the estimate
- estimate $U^\pi(x)$ directly from obtained rewards following some π

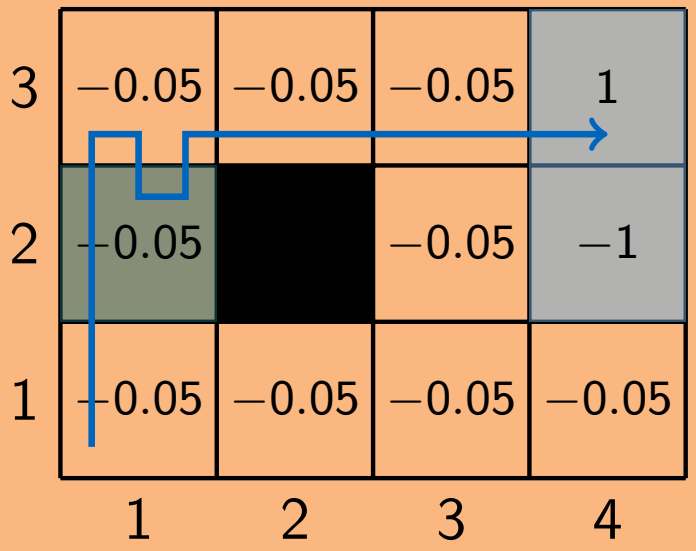
The estimate $\widehat{U}^\pi(x)$ of $U^\pi(x)$ is the average of **utility samples** for x. A utility sample (also called *reward-to-go*) for x is the sum of discounted rewards obtained on the path from x to a terminal state (end of episode).

$\widehat{U}^\pi(x)$ converges to $U^\pi(x)$. To make it converge to $U(x)$, exploration is needed, e.g., using GLIE (with $\widehat{U}^\pi(x)$ plugged in for $\widehat{U}(x)$ in (??)).

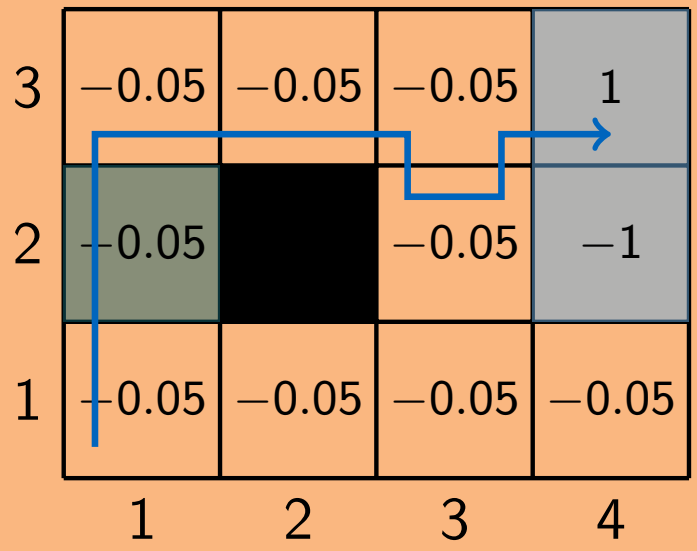
FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Example: Direct Utility Estimation

Estimate $\widehat{U}((1,2))$ with $\gamma = 1$ after two episodes



Episode 1: 2 samples
From 1st time in (2,1):
$6 \cdot (-0.05) + 1 = 0.7$
From 2nd time in (2,1):
$4 \cdot (-0.05) + 1 = 0.8$



Episode 2: 1 sample
$6 * (-0.05) + 1 = 0.7$

$$\widehat{U}((1,2)) = (0.7 + 0.8 + 0.7)/3 \approx 0.73$$

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Temporal Difference Learning

DUE ignores the strong correlation of utilities between adjacent states. The **temporal difference learning** method makes an explicit use of it.

First assume that x' always follows x under π ($x, x' \in X$, x non-terminal). That is, $P^\pi(x' \mid x) = 1$. Then the 2nd case of (??) changes to

$$U^\pi(x) = r(x) + \gamma U^\pi(x')$$

If this relation btw. $U^\pi(x)$ and $U^\pi(x')$ is not observed for the estimates $\widehat{U}^\pi(x)$ and $\widehat{U}^\pi(x')$ on a transition from x to x', i.e.,

$$\widehat{U}^\pi(x) \neq r + \gamma \widehat{U}^\pi(x')$$

(where r is the reward received in x) then the value of $\widehat{U}^\pi(x)$ must be 'corrected'.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Temporal Difference Learning (cont'd)

This is done by setting

$$\widehat{U}^{\pi}(x) := \widehat{U}^{\pi}(x) + \alpha_k \left(r + \gamma \widehat{U}^{\pi}(x') - \widehat{U}^{\pi}(x) \right) \tag{1}$$

moving $\widehat{U}^{\pi}(x)$ closer to the correct value $U^{\pi}(x)$ with speed given by the *learning rate* $\alpha_k \in \mathbb{R}$.

(1) is used as well when x can be followed by different states x' with non-zero probabilities. The higher $P^{\pi}(x' \mid x)$, the more times (1) is applied for x, x' during the interaction.

Lemma 1

Iterating (1), $\widehat{U}^{\pi}(x)$ *converges to* $U^{\pi}(x)$ *for all* $x \in X$ *if* $\sum_{k=1}^{\infty} \alpha_k = \infty$ *and* $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$.

OF ELECTRICAL ENGINEERING CTU IN PRAGUE

DUE vs. TD vs. ADP

DUE tends to converge slower to the true state utilities than ADP in terms of time steps (k). But convergence may be faster in terms of runtime as the costly value or policy iteration is avoided.

TD is a middle ground between DUE and ADP.

Although DUE and TD do not need a model p of P for computing utilities, they still need it for evaluating the policy through (??) (greedy case) or (??) (GLIE case).

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Q-Learning

Consider an alternative method **Q-Learning** which does not need p at all.

Instead of state utility, define the utility of a *state-action pair* called the **Q value** as

$$Q(x,y) = \begin{cases} r(x) \text{ if } x \text{ is terminal} \\ r(x) + \gamma \sum_{x' \in X} P(x' \mid x, y) \max_{y' \in Y} Q(x', y') \quad \text{otherwise} \end{cases} \tag{2}$$

Note: unlike $U^\pi(x)$, $Q(x,y)$ does not depend on a policy π.

Given an estimate $\widehat{Q}$ of Q, instead of (??), the greedy policy is given by

$$\pi(x) = \arg\max_{y \in Y} \widehat{Q}(x,y) \tag{3}$$

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Q-Learning

Similarly to TD, the estimate $\widehat{Q}$ is iterated on each transition from x to x' where y is the action that taken on x and r is the reward in x:

$$\widehat{Q}(x,y) := \widehat{Q}(x,y) + \alpha_k \left(r + \gamma \max_{y' \in Y} \widehat{Q}(x',y') - \widehat{Q}(x,y) \right) \tag{4}$$

ultimately converging to $Q(x,y)$.

Exploration can be promoted by GLIE (??), using $\widehat{Q}(x,y)$ instead of (??) or by changing (3) into

$$\pi(x) = \arg\max_{y \in Y} f\left(\widehat{Q}(x,y), N(x,y) \right)$$

where f is an exploration function and $N(x,y)$ is the number of times x has occurred with y taken on it.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

SARSA Learning

In (4), the action maximizing $\widehat{Q}(x', y')$ is $\pi(x')$, i.e., the one chosen by the greedy policy (3) on state x'. So with a greedy policy, (4) simplifies into

$$\widehat{Q}(x, y) := \widehat{Q}(x, y) + \alpha_k \left(r + \gamma \widehat{Q}(x', y') - \widehat{Q}(x, y) \right) \tag{5}$$

where y' is the action taken on state x'.

The iteration (5) uses the *s*tate - *a*ction - *r*eward - *s*tate - *a*ction quintuplet

$$x, y, r, x', y'$$

hence this modification of Q-learning is called **SARSA**. It is used even with non-greedy policies.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Q vs. SARSA

The iteration (4) in Q-Learning is *less* dependent on the policy employed than (5) in SARSA because y' in the latter is determined by the policy.

For this reason, Q-Learning is called an **off-policy** strategy whereas SARSA is called an **on-policy**.

In Q-Learning, $\widehat{Q}$ tends to converge faster to Q even with a policy far from optimal.

In SARSA, it tends to converge faster with a *partially enforced* policy, i.e. where $\pi(x)$, $x \in X'$ are fixed for some $X' \subset X$. (E.g., in state $(1, 1)$ the agent must go up independently of state utilities.)

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Representation of $\widehat{U}$ and $\widehat{Q}$

If $\widehat{U}, \widehat{Q}$ are represented as arrays (one cell for each $x \in X$, they require at least $\mathcal{O}(|X|)$ resp. $\mathcal{O}(|X| \cdot |Y|)$ memory and time.

This would not scale to large state spaces X in real-life problems, e.g.,

- Backgammon or Chess: $|X|$ somewhere btw. 10^{20} and 10^{45}
- No way to capture in an array, let alone do policy/value iteration

A more compact ('generalized') model for

$$\widehat{U} : X \to \mathbb{R} \text{ or } \widehat{Q} : X \times Y \to \mathbb{R}$$

is needed that allows learning (updating) from $[x, y, r, x']$ or $[x, y, r, x', y']$ samples.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Feature-Based Representation of $\widehat{U}$

Consider learning $\widehat{U}$ with the DUE agent.

A simple option is to define a set of features, i.e., mappings $\phi^i : X \to \mathbb{R}$ informative of the utility, and use a *regression model*.

$$\widehat{U}(\mathbf{w}, x) = \sum_{i=1}^{n} w^i \phi^i(x)$$

and adapt the parameters $\mathbf{w} = [w^1, w^2, \ldots, w^n]$ at each episode's end to reduce the squared error

$$E_j(\mathbf{w}, x) = \frac{1}{2}\left(\widehat{U}(\mathbf{w}, x) - u_j(x)\right)^2$$

where $u_j(x)$ is the utility sample for x available at the end of episode $j = 1, 2, \ldots$ (when a terminal state is reached).

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Feature-Based Representation of $\widehat{U}$ (cont'd)

Going against the error gradient with learning rate $\alpha \in \mathbb{R}$:

$$w^i := w^i - \alpha \frac{\partial E_j(\mathbf{w}, x)}{\partial w^i} = w^i + \alpha_k \left(u_j(x) - \widehat{U}(\mathbf{w}, x) \right) \frac{\partial \widehat{U}(\mathbf{w}, x)}{\partial w^i}$$

Example: Let $[\phi^1(x), \phi^2(x)] = [x^1, x^2]$, i.e., the agent's coordinates in the grid environment and $\phi^3 \equiv 1$. Then

$$\widehat{U}(\mathbf{w}, x) = w^1 x^1 + w^2 x^2 + w^3$$

and the iterative update:

$$\begin{aligned} w^1 &:= w^1 + \alpha(u_j(x) - \widehat{U}(\mathbf{w}, x))x^1, \\ w^2 &:= w^2 + \alpha(u_j(x) - \widehat{U}(\mathbf{w}, x))x^2 \\ w^3 &:= w^3 + \alpha(u_j(x) - \widehat{U}(\mathbf{w}, x)) \end{aligned}$$

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Feature-Based Representation of $\widehat{U}$: Notes

1. Observe:

$$\frac{\partial \widehat{U}(\mathbf{w}, x)}{\partial w^i} = \phi^i(x)$$

So the derivative is simple even with non-linear features such as

$$\phi^i(x) = \sqrt{(x^1 - 4)^2 + (x^2 - 3)^2}$$

measuring the Euclidean ('air') distance to the terminal state $(4, 3)$.

2. Features allow to deal with a kind of *partial state observability* (similar to one considered here). If a fixed component of the state is not observable, features can be designed that do not use that component.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Feature-Based Representation of $\widehat{Q}$

A similar approach can be applied with the Q-Learning agent:

$$\widehat{Q}(\mathbf{w}, x, y) = \sum_{i=1}^{n} w^i \phi^i(x, y)$$

where ϕ^i are predefined features of state-action pairs.

Follow the gradient descent (again, $\frac{\partial \widehat{Q}(\mathbf{w},x,y)}{\partial w^i} = \phi^i(x, y)$) at each time k

$$w^i := w^i + \alpha \left(r + \gamma \max_{y'} \widehat{Q}(\mathbf{w}, x', y') - \widehat{Q}(\mathbf{w}, x, y) \right) \phi^i(x, y)$$

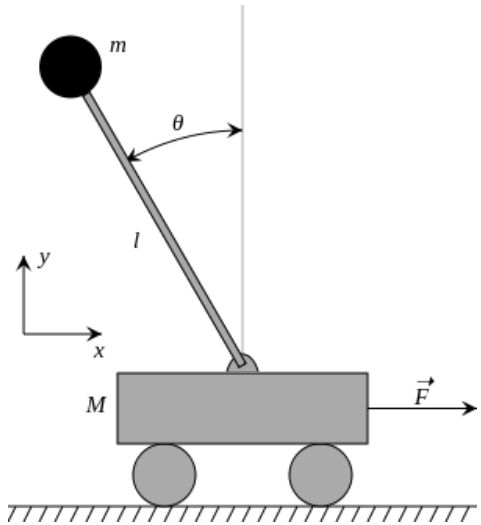
The principle is simple, the art is in designing good features ϕ^i.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Inverted Pendulum Demo

Real-valued features especially appropriate where environment is a dynamic physical system. Typical features are *positions* and *accelerations* of objects.

Example: inverted pendulum



Videos: Single (Experience Replay - see later), Triple (!).

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE