Symbolic Machine Learning

Filip Železný and Jiří Kléma

1 A General Framework

1.1 Percepts and Actions

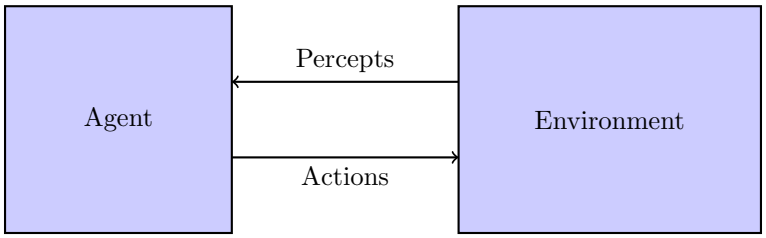

Figure 1: The basic situation under study.

- Discrete *time*
$$k = 1, 2, \ldots$$
- *Percepts*
$$\forall k : x_k \in X$$
- *Actions*
$$\forall k : y_k \in Y$$

X and Y are finite.

A *history* is a sequence of alternating percepts and actions, i.e,

$$x_1, y_1, x_2, y_2, \ldots, x_k, y_k$$

and is denoted as $xy_{\leq k}$. Similarly, $xy_{<k} = x_1, y_1, x_2, y_2, \ldots, x_{k-1}, y_{k-1}$. There is a probability distribution μ on histories

$$\mu(xy_{\leq k}) = \mu(x_1)\mu(y_1|x_1)\mu(x_2|x_1, y_1)\ldots\mu(x_k|xy_{<k})\mu(y_k|x_k, xy_{<k}) \quad (1)$$

After the initial 'kick-off' x_1 from the environment distributed according to $\mu(x_1)$, any percept x_k generated by the environment at time k depends on the entire preceding history $xy_{<k}$ according to

$$\mu(x_k|xy_{<k}) \tag{2}$$

Actions y_k are determined by agent's decision *policy* which also depends on the history as well as the current percept and are distributed according to $\mu(y_k|x_k, xy_{<k})$. We will assume that the policy is *deterministic*. Thus we identify the policy with function $\pi : (X \times Y)^* \times X \to Y$, so

$$y_k = \pi(xy_{<k}, x_k) \tag{3}$$

This means that $\mu(\pi(xy_{<k}, x_k)|xy_{<k}, x_k) = 1$.

The following diagram illustrates the influences between the introduced variables.

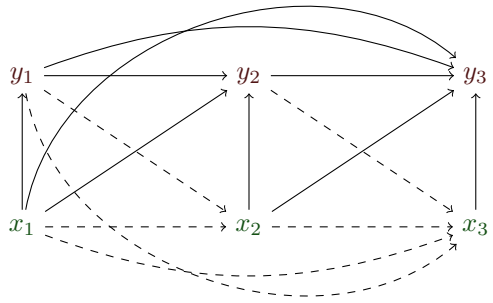

Figure 2: Influence diagram for actions y_k and percepts x_k for $1 \leq k \leq 3$ with full lines indicating deterministic influences (via π) and dashed lines showing probabilistic influences (via μ).

While we have yet to define what goals the agent should achieve through interaction with the environment, obviously some histories will be "better" than others in terms of the goal achievement. To maximize the probability (1) of good histories, the agent cannot influence the conditional probability (2), which is inherent to the environment, but it can follow a good policy (3). However, the effect of actions proposed by the policy depends on (2) which is generally not known to the agent. So the agent needs to recognize the environment by experimenting with it. This is formally reflected by (3) where action y_k depends not only on the current percept x_k but also on the history $xy_{<k}$. So the agent

will generally make different decisions $y_k \neq y_{k'}$ for $k > k'$ even if $x_k = x_{k'}$ because the experience $xy_{<k}$ at time k is larger than experience $xy_{<k'}$ at time k'. This is our first reflection of *learning*.

How does the agent know how well it is doing? This information comes from the environment through a specially distinguished part of the percepts, called *rewards*. The remaining part of each percept contains *observations*. Formally, $X = O \times R$, $o_k \in O$, $r_k \in R \subset \Re$, so $x_k = (o_k, r_k)$. Since X is assumed finite, it follows that rewards have their finite minimum and maximum.

The probability of x_k in (2) can be written in terms of the marginals μ_O and μ_R

$$\mu(x_k|xy_{<k}) = \mu(o_k, r_k|xy_{<k}) =$$
$$\mu_O(o_k|r_k, xy_{<k})\mu_R(r_k|xy_{<k}) = \mu_R(r_k|o_k, xy_{<k})\mu_O(o_k|xy_{<k})$$

which also makes it clear that o_k and r_k are in general not mutually independent, even if conditioned on $xy_{<k}$.

1.2 Nonsequential Cases

Scenarios where current percepts depend on the history of previous percepts and actions are called *sequential*. The framework described so far is maximally general in that dependence is assumed on the entire history from $k = 1$ on. On the other extreme are *nonsequential* scenarios. Here, observations are independent of the history as well as the current reward, i.e.

$$\mu_O(o_k|r_k, xy_{<k}) = \mu_O(o_k) \tag{4}$$

and thus $o_1, o_2, \ldots$ are mutually independent random variables sampled from the same distribution μ_O (they are "i.i.d.").

Rewards in the nonsequential case are assumed to depend only the immediately preceding observation and the action taken on it, i.e.

$$\mu_R(r_k|o_k, xy_{<k}) = \mu_R(r_k|o_{k-1}, y_{k-1}) \tag{5}$$

however, since y_{k-1} is functionally determined by the history $xy_{<k-1}$ and percept $x_{k-1} = (o_{k-1}, r_{k-1})$ through (3), we may rewrite (5) as

$$\mu_R(r_k|o_{k-1}, r_{k-1}, xy_{<k-1}) \tag{6}$$

which makes it clear that reward r_k depends on previous rewards, and thus rewards $r_1, r_2, \ldots$ are not i.i.d.. This is natural since if they were, it would mean the agent never improves its performance.

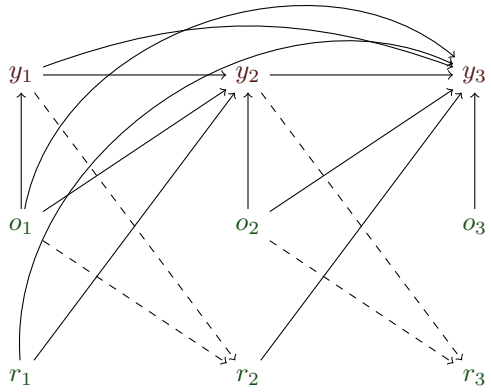

Figure 3: Influence diagram for actions y_k, observations o_k, and rewards r_k for $1 \leq k \leq 3$ with full lines indicating deterministic influences (via π) and dashed lines showing probabilistic influences (via μ) in the nonsequential case.

1.3 Batch Learning

We will also consider a specific yet important nonsequential case called *batch learning* consisting of two phases switching right after time K

- the *learning (training, exploration) phase* at $k = 1, 2, \ldots K$
- the *action (testing, exploitation) phase* taking place in $k = K+1, K+2, \ldots$

In the action phase, the agent no longer changes its decision making (for $k, k' > K$, if $x_k = x_{k'}$ then $y_k = y_{k'}$), and ignores rewards. So the action proposed by the policy depends only on the current observation and the history only up to time K. So for $k > K$, (3) changes here into

$$y_k = \pi(xy_{\leq K}, o_k) \tag{7}$$

and (5, 6) change into

$$\mu_R(r_k|o_{k-1}, y_{k-1}) = \mu_R(r_k|o_{k-1}, xy_{\leq K}) \tag{8}$$

because due to (7), y_{k-1} is determined by o_{k-1} and $xy_{\leq K}$. The observation o_{k-1} does not depend on rewards due to (4). So reward r_k does not depend on previous rewards $r_{k'}$, $k > k' > K$. Another way to say this is that rewards in the action phase are conditionally independent of each other, given the learning

phase history:

$$\mu_R(r_k, r_{k'}|xy_{<K}) = \mu_R(r_k|xy_{<K})\mu_R(r_{k'}|xy_{<K}) \tag{9}$$

The following figure illustrates the batch-learning situation.

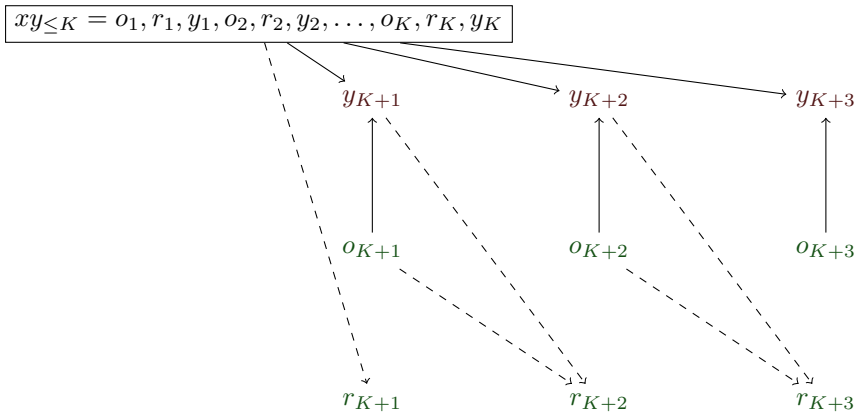

Figure 4: Influence diagram for actions y_k, observations o_k, and rewards r_k in the action phase ($k > K$) of batch learning with full lines indicating deterministic influences (via π) and dashed lines showing probabilistic influences (via μ). The top row indicates the influence of the learning phase on the agent's decisions in the action phase.

We can further express the distribution of r_k ($\forall k > K$) without conditioning on the observations, which are i.i.d. by (4)

$$\mu_R(r_k|xy_{\leq K}) = \sum_{o_{k-1} \in O} \mu_O(o_{k-1})\mu_R(r_k|o_{k-1}, xy_{\leq K}) \tag{10}$$

So rewards in the action phase are i.i.d. according to the above distribution conditioned only on the history of the learning phase.

1.4 Rewards and Goals

It has been obvious that the agent's goal is to maximize rewards. Here we formalize this goal. Since rewards come at each point of the history, we want the agent to maximize their sum up to a finite time *horizon* $m \in N$

$$r_1 + r_2 + \ldots + r_m$$

or, more generally, maximize the *discounted* sum

$$\sum_{k=1}^{\infty} r_k \gamma_k$$

where $\forall k : \gamma_k \geq 0$ and $\sum_{i=1}^{\infty} \gamma_i < \infty$, so the above sum converges.

But since rewards are probabilistic, the agent should choose a sequence $y_{\leq m}$ of actions leading to a high *expected* cumulative reward

$$\sum_{r_{\leq m}} \mu_R(r_{\leq m}|y_{\leq m})(r_1 + r_2 + \ldots + r_m)$$

or, in the discounted case

$$\lim_{m\to\infty} \sum_{r_{\leq m}} \mu_R(r_{\leq m}|y_{\leq m}) \sum_{k=1}^{m} r_k \gamma_k$$

where the first sum in both cases goes over all possible reward sequences $r_{\leq m}$ (since R and m are finite, there is a finite number of them).

However, for the specific case of *batch learning*, we establish a more appropriate learning goal. First, we do not care about maximizing rewards in the *learning phase* as the purpose of this phase is to probe the environment even at the price of possibly poor rewards. Second, in the *action phase* after time K, the rewards r_k, $k > K$ are sampled independently from the same distribution (10) so we can simply maximize their expectation with respect to this distribution

$$\sum_{r_k \in R} \mu_R(r_k|xy_{\leq K}) r_k \tag{11}$$

It is again obvious from the formula that the expected reward only depends on the learning phase history $xy_{\leq K}$, after which the agent no longer changes its action policy. Note also that the batch learning scenario allowed us to define an objective (11) without the need to choose the parameters m or γ_k ($k = 1, 2, \ldots$) needed in the sequential scenario.

1.5 States

With the exception of the non-sequential scenario, our framework has been very general in that percepts x_k generally depend on entire histories $xy_{<k}$. In the real world, many histories may be equivalent, i.e. leading to the same probabilities of x_k conditioned on action y_{k-1}. This can be formalized through the notion of *environment state* $s_k^E \in S^E$ at time k. We will assume that the state is

probabilistically established by the preceding state and the last agent's action through the following state *update* distribution

$$\mathcal{S}^E(s_k^E|s_{k-1}^E, x_{k-1}, y_{k-1}) \tag{12}$$

and that percepts depend only on the state rather than the history, thus changing (2) into

$$\mu(x_k|s_k^E) \tag{13}$$

This modification does not lessen the generality of the framework if we allow S^E to be infinite as then there could simply exist a distinct state for each possible history (there is an infinite number of possible histories for unbounded k). Indeed, if one instantiates the distribution (12) to the functional dependence

$$s_k^E = s_{k-1}^E \parallel (x_{k-1}, y_{k-1})$$

where $\parallel$ denotes concatenation, s_k^E will simply collect the entire history and its occurrence in (13) would be just a different name for $xy_{<k}$ in (2). However, we will make the important assumption that

$$|S^E| < \infty \tag{14}$$

which will significantly simplify the framework. In practical tasks, there will be far fewer states than possible histories. Furthermore, we will make an intuitively motivated assumption, that the influence between environment states and the emitted percepts are only one-directional. In particular, the percepts depend on states by (13) but not vice versa, so we remove x_{k-1} from (12)

$$\mathcal{S}^E(s_k^E|s_{k-1}^E, x_{k-1}, y_{k-1}) = \mathcal{S}^E(s_k^E|s_{k-1}^E, y_{k-1}) \tag{15}$$

A similar reasoning applies to the agent, whose actions generally depend on the entire history as in (3). Again, many histories can lead to the same mapping from percepts to actions, for example because the agent has built the same *model* of the environment throughout the different histories. So analogically to the environmental states, we introduce the notion of *agent state* $s_k^A \in S^A$. Since we work with deterministic agents, we will assume that the state depends functionally on the preceding state and the current percept as

$$s_k^A = \mathcal{S}^A(s_{k-1}^A, x_k) \tag{16}$$

and instead of (3), we will assume that actions depend on the state rather than the history itself

$$y_k = \pi(s_k^A, x_k) \tag{17}$$

and again will postulate that

$$|S^A| < \infty \tag{18}$$

The formalization using states results in the agent and environment structured depicted in Fig. 5. The diagram of variable influences is shown in Fig. 6.

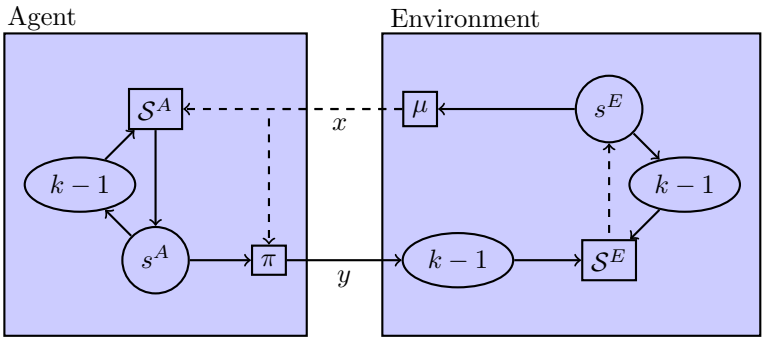

Figure 5: The state-based scheme of agent-environment interaction. Full and dashed lines denote functional and probabilistic influences, respectively. The $k-1$ nodes denote a one-step time lag.

The agent state s^A has a very natural interpretation as it corresponds to the agent's model of the environment at time k, whereas π is the the interpreter of the model. For example, s^A may encode a set of logical rules, and π may be a logical prover deriving actions as logical consequences of the rules. Since the model description has to fit in a finitely bounded memory, there can be only a finite number of different models, aka states. Therefore, the assumption in (18) is well justified.

In light of this state-model interpretation, it may seem that the model update in (16) should also include previous percepts $x_{k-1}, x_{k-2}, \ldots$ and actions $y_{k-1}, y_{k-2}, \ldots$ as arguments because the history of percepts and actions (in combination with the current percept) is obviously informative for updating the model. However, note that this is not necessary as the update function $\mathcal{S}^A$ in (16) can always be made to store any finite number of percepts, and previous states in the memory, as part of the evolving state, because they are inputs to the update step (16). But also any historical action $y_{k'}$, $k' < k$ can be retrieved by first retrieving $s^A_{k'}$ and $x_{k'}$ from the memory and then using (16). This is possible because π is deterministic and can be simulated by $\mathcal{S}^A$.

Just like in the state-free case, also with the state-based formulation the situation simplifies a lot in the *nonsequential case*. Here, the environment has no states (i.e., memory) at all. Observations are again generated i.i.d from (4) and rewards depend as in (5) on the last observation and action taken on it by the agent. However, the non i.i.d. agent states are still in place and cause the mutual dependence of successive actions and rewards. This is shown in Fig. 7.

A further simplification comes in the special *batch-learning* scenario of the nonsequential case. In the action phase of the latter, the agent never updates its

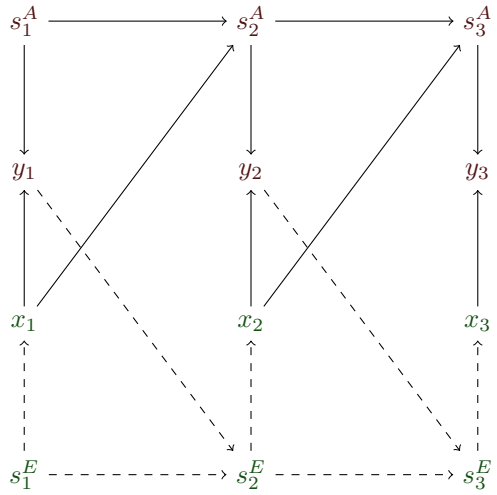

Figure 6: Influence diagram for states s_k^A, actions y_k and percepts x_k for $1 \leq k \leq 3$ with full lines indicating deterministic influences (via π and $\mathcal{S}^A$) and dashed lines showing probabilistic influences (via μ and $\mathcal{S}^E$).

state (model) and generates actions using the last state s_K^A of the learning phase. This is captured again in Fig. 4 with the only change that the boxed top row in it will contain s_K^A.

1.6 Prior Knowledge

- *Implicit*: the setting of S^A ("hard bias") and $\mathcal{S}^A$ ("soft bias")
- *Explicit*: the setting of s_1^A ("background knowledge")

1.7 State (Model) Representations

See Fig. 8

1.8 Learning Scenarios

1. on-line concept learning
2. batch concept learning

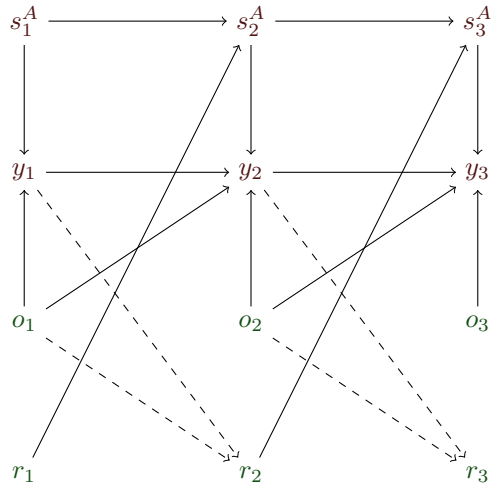

Figure 7: Influence diagram for states s_k^A, actions y_k, observations o_k, and rewards r_k for $1 \leq k \leq 3$ with full lines corresponding to deterministic influences (via π and $\mathcal{S}^A$) and dashed lines showing probabilistic influences (via μ and $\mathcal{S}^E$) in the nonsequential case.

3. query-based learning
4. reinforcement learning
5. universal learning

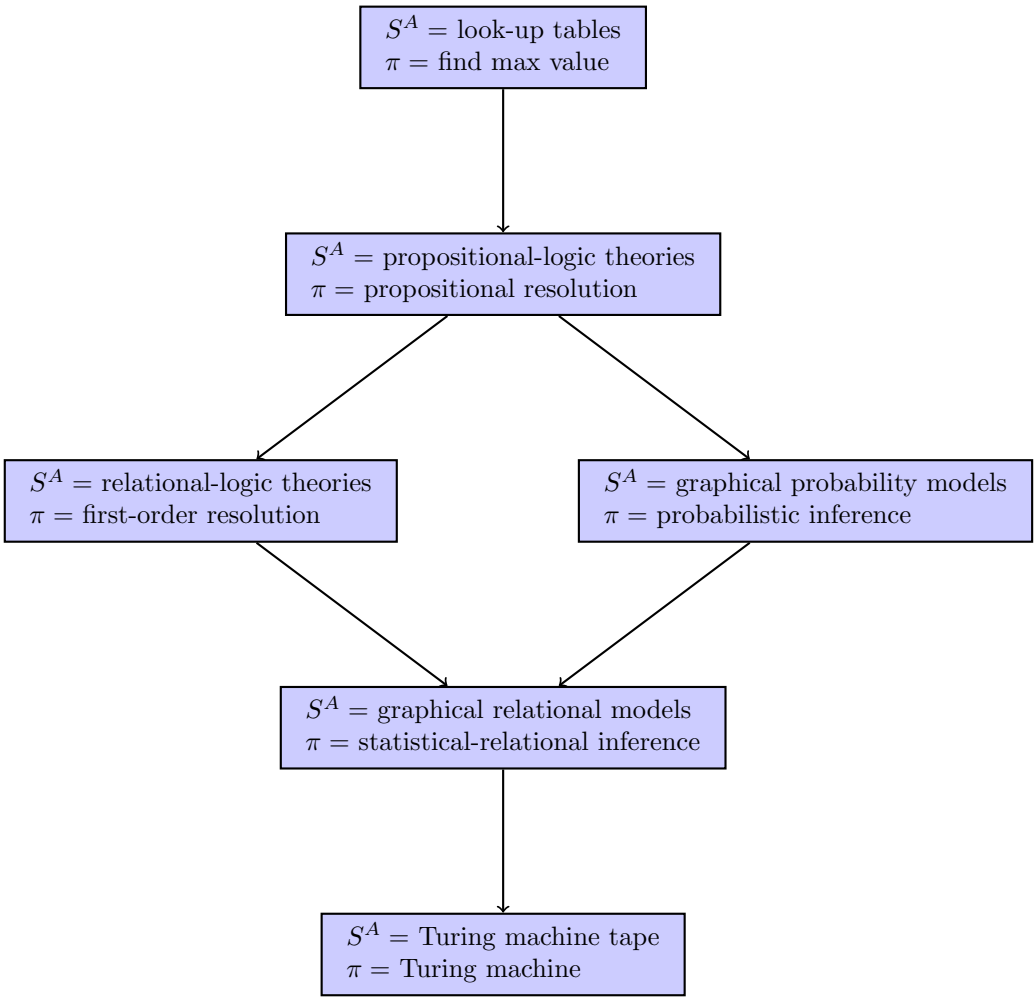

Figure 8: State (model) representations and their corresponding policy classes (interpreters) considered in this course.