Lecture Note 19

1 Overview

In the previous lecture, we studied constraint sampling as a generic approach to dealing with the large number of constraints in the approximate LP, and showed that, by sampling a number of constraints that is polynomial on the number of variables in the LP, it is possible to ensure that almost all constraints will be satisfied, with high probability. The hope that an LP with a large number of constraints and a small number of variables may be solved efficiently — either exactly or approximately — stems from the fact that many of the constraints should be redundant; in particular, it is known that only a number of constraints equal to the number of variables is binding at the optimal solution. This gives hope that, at least in certain problem-specific situations, other approaches besides constraint sampling may be used for dealing with the large number of constraints. In particular, the following approaches may be possible:

- We may be able to replace the original constraints $T\Phi r \geq \Phi r$ with an equivalent set of constraints $A_i r \geq b_i, i = 1, \ldots, N$ where N is "small;"
- Constraint Generation We may be able to solve the LP exactly without including all constraints by solving it incrementally, as follows:
  - start with small subset of constraints
  - solve smaller LP
  - add one or more violated constraints
  - repeat until no violated constraints can be found.

Both approaches can be found in the literature; e.g., Morrison and Kumar [2] replace the exponentially many constraints in the approximate LP with a manageable number of constraints in problems involving queueing networks, and Grötschel and Holland [1] solve travelling salesman problems involving up to 2^{60} constraints by doing constraint generation. In today's lecture, we will study *factored MDPs*, a reasonably general class of MDPs that lends itself well to both approaches.

2 Factored MDPs

The underlying idea in factored MDPs is that many high-dimensional MDPs are actually generated by systems with many parts that are weakly interconnected. Each part i has an associated state variable X_i, so that the full state of the system is described by a vector $(X_1, \ldots, X_n)$. We assume that costs are *factored*, i.e.,

$$g(x) = \sum_j g_j(X_{Z_j}), \tag{1}$$

where $Z_j \subset \{1, \ldots, n\}$ and X_{Z_j} indicates a (hopefully small) subset of the state variables. Moreover, we also assume that transition probabilities are *factored*, i.e.,

$$P_a(X_i(t+1)|X(t)) = P_a(X_i(t+1)|X_{Z_i}(t)) \ \forall i,$$

where once again $Z_i \subset \{1, \ldots, n\}$ and X_{Z_i} indicates a (hopefully small) subset of the state variables. In words, one way of viewing this assumptions is that costs are mostly local to the various parts of the system, and dynamics are also mostly local, with each state variable being affected only the subset of state variables it interacts with. Note that, in the long run, if all state variables are directly or indirectly interconnected, the evolution of a particular state variable may still be affected by all others.

A common way of representing factored MDPs is through a dynamic Bayesian network, as shown in Figure 1. The nodes at the left and right represent the state variables in subsequent time steps, and arcs indicate the dependencies between state variables across time steps. This figure may be generalized to include dependencies within the same time step.

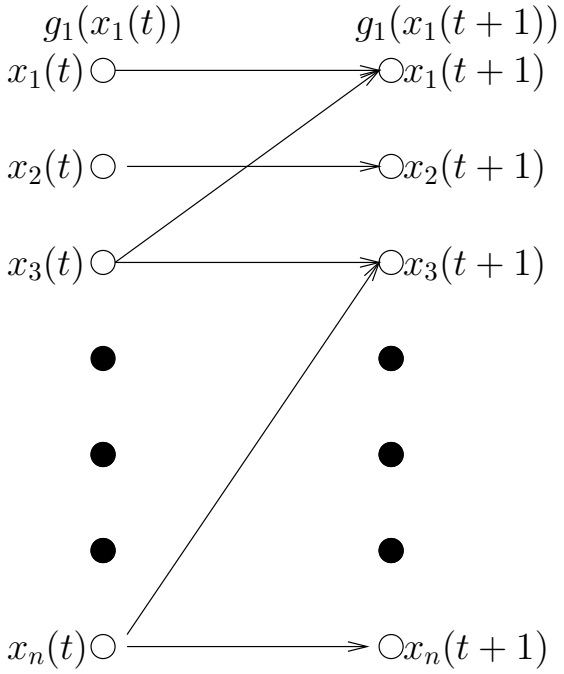

Figure 1: Each state is influenced by a small subset of states in every time stage

Example 1 *Consider the queueing network represented in Figure 2. With our usual choice of costs* $g_a(x) = \sum_i x_i$*, corresponding to the total number of jobs in the system, it is clear that stage costs are factored. Moreover, transition probabilities are also factored; for instance, we have*

$$P_a(x_2(t+1)|x(t)) = P_{a_1,a_2}(x_2(t+1)|x_1(t), x_2(t)),$$

since the number of jobs in queue 2 in the next time step is determined exclusively by potential departures from queue 1 — which depend only on $x_1(t)$ *and* $a_1(t)$ *— and potential departures from queue 2 — which depend only on* $x_2(t)$ *and* $a_2(t)$*.*

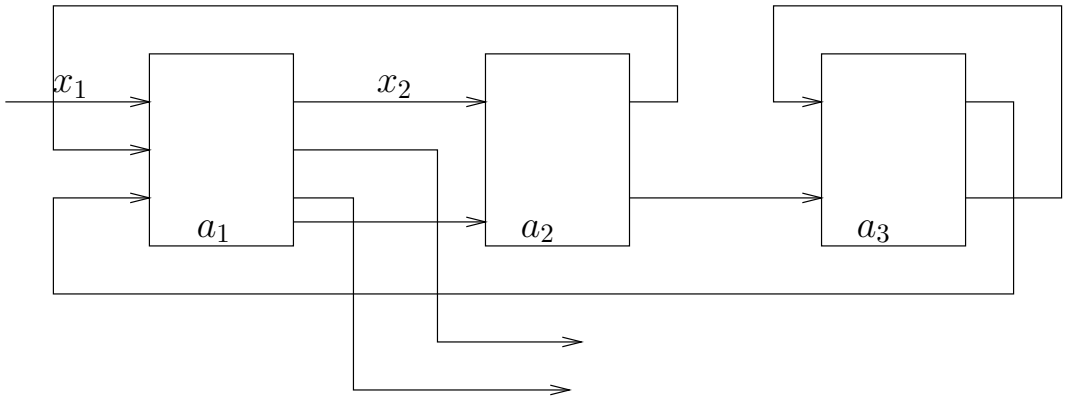

Figure 2: An queueing network

We consider approximating J^* with a factored representation:

$$J^*(x) \approx \sum \phi_i(x_{w_i}) r_i \triangleq \tilde{J}(x),\ W_i \subset \{1, \ldots, n\}.$$

Note that, in general, the optimal cost-to-go function J^* does not have an exact factored representation. However, factored approximations are appealing both because, if the system is indeed only loosely interconnected, we can expect J^* to be roughly factored, and perhaps most importantly, factored approximations give rise to decentralized policies. Indeed, note that Q factors associated with a factored approximation $\tilde{J}$ are also factored:

$$\begin{aligned}
Q(x,a) &= g_a(x) + \alpha \sum_y P_a(x;y)\tilde{J}(y) \\
&= g_a(x) + \alpha \sum_{y_1,\ldots,y_n} P_a(x_1,\ldots,x_n;y_1,\ldots,y_n) \sum_i \phi(y_{W_i}) r_i \\
&= g_a(x) + \alpha \sum_i \sum_{y_{W_i}} P_a(x_1,\ldots,x_n;y_{W_i}) \phi(y_{W_i}) r_i \\
&= g_a(x) + \alpha \sum_i \sum_{y_{W_i}} P_a(x_{Z_i};y_{W_i}) \phi(y_{W_i}) r_i \\
&= g_a(x) + f(x_{Z_i};r)
\end{aligned}$$

since $y_{W_i}(t+1)$ is only influenced by a subset $X_{Z_i}(t)$ of $X(t)$.

3 Efficient handling of constraints in factored MDPs

We will now show that the approximate LP constraints

$$g_a(x) + \alpha \sum_y P_a(x,y)\phi(y) r \geq \phi(x) r$$

can be dealt with efficiently when we consider factored MDPs with factored cost-to-go function approximations. For simplicity, let us denote each state-action pair (x, a) by a vector-valued variable $t = (t_1, t_2, \ldots, t_m) = (x_1, \ldots, x_n, a_1, \ldots, a_p)$. Then we are interested in dealing with a set of constraints

$$\sum_i f_i(t_{W_i}, r) \leq 0, \, \forall \, t. \tag{2}$$

The main difficulty is that t can take on an unmanageably large number of values — as many as the number of state-action pairs in the system. We will show that these constraints can be replaced by a smaller, equivalent subset. Moreover, we will show identifying a violated constraint can be done efficiently, which allows for using constraint generation schemes.

The first step is to rewrite (2) as

$$\max_t \sum_i f_i(t_{W_i}, r) \leq 0.$$

Consider solving the maximization problem above for a fixed value of r. The naive approach is to enumerate all possible values of t and take the one leading to maximum value of the objective. However, since each of the terms $f_i(t_{W_i}, r)$ depends only on a subset t_{W_i} of the components of t, the problem can be solved more efficiently via *variable elimination*. We illustrate the procedure through the following example.

Example 2 *Consider*

$$\max_{t_1,t_2,t_3,t_4} f_1(t_1, t_2) + f_2(t_2, t_3) + f_3(t_2, t_4) + f_4(t_3, t_4).$$

For simplicity, assume that $t_i \in \{0, 1\}$, for $i = 1, 2, 3, 4$. If we solve the optimization problem above by enumerating all possible solutions, there are on the order of $O(2^4)$ operations. Consider optimizing over one variable at a time, as follows:

1. *Eliminate variable t_2: For each possible value of t_2,t_3, we find*

$$e_1(t_2, t_3) = \max_{t_4} f_3(t_2, t_4) + f_4(t_3, t_4),$$

   and rewrite the problem as

$$\max_{t_1,t_2,t_3} f_1(t_1, t_2) + f_2(t_2, t_3) + e_1(t_2, t_3).$$

2. *Eliminate variable t_3: For each possible value of t_2, we find*

$$e_2(t_2) = \max_{t_3} f_1(t_2, t_2) + e_1(t_2, t_3),$$

   and rewrite the original problem as

$$\max_{t_1,t_2} f_1(t_1, t_2) + e_2(t_2).$$

3. *Eliminate variable t_2: For each possible value of t_1, we find*

$$e_3(t_1) = \max_{t_2} f_1(t_1, t_2) + e_2(t_2).$$

4. *Solve* $\max_{t_1} e_3(t_1)$.

Solving the problem via variable elimination requires on the order of $O(2^3)$ operations. More generally, variable elimination leads to exponential reduction in computational complexity relative to the naive approach.

The previous example suggests variable elimination as an efficient approach to verifying whether a candidate solution r is feasible for all constraints, and identifying a violated constraint if that is not the case. Therefore constraint generation can be implemented efficiently when we consider factored MDPs with factored cost-to-go function approximations. Moreover, the procedure described in the example can also be used to generate a smaller set of constraints, if we introduce new variables in the LP. Indeed, let $e_i(t_{Z_i})$ be each of the functions involved in the scheme (including the original functions f). Each function is given by

$$e_i(t_{Z_i}) = \max_{t_{j_i}} \sum_{k \in K_i} e_k(t_{Z_i}, t_{j_i}).$$

For each function e_i, we introduce a set of variables $u_{e_i}^{t^i}$, where each t^i corresponds to one possible assignment to variables t_{Z_i}; for instance, in the example above, we would have variables

$$u_{e_1}^{00}, u_{e_1}^{01}, u_{e_1}^{10}, u_{e_1}^{11},$$

associated with function $e_1(t_2, t_3)$ and all possible assignments for variables t_2 and t_3.

With this new definition, the original constraints can be replaced with

$$u_{f_i}^{t^i} = f_i(t^i, r), \forall t_i,$$

and

$$u_{e_i}^{t^i} \geq \sum_{k \in K_i} e_k(t^i, t_{j_i}), \forall t^i, t_{j_i}.$$

References

[1] M. Grötschel and O. Holland. Solution of large-scale symmetric travelling salesman problems. *Mathematical Programming*, 51:141–202, 1991.

[2] J.R. Morrison and P.R. Kumar. New linear program performance bounds for queueing networks. *Journal of Optimization Theory and Applications*, 100(3):575–597, 1999.