Topics in Reinforcement Learning: Rollout and Approximate Policy Iteration

ASU, CSE 691, Spring 2021

Links to Class Notes, Videolectures, and Slides at
http://web.mit.edu/dimitrib/www/RLbook.html

Dimitri P. Bertsekas
dbertsek@asu.edu

Lecture 12
Aggregation Methods

Outline

Aggregation within the Approximation in Value Space Framework

Approximate minimization

First Step "Future"

$$\min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha \tilde{J}(j)\big)$$

Approximations:

Replace $E\{\cdot\}$ with nominal values (certainty equivalence)

Adaptive simulation

Monte Carlo tree search

Computation of $\tilde{J}$:

Problem approximation

Rollout

Approximate PI

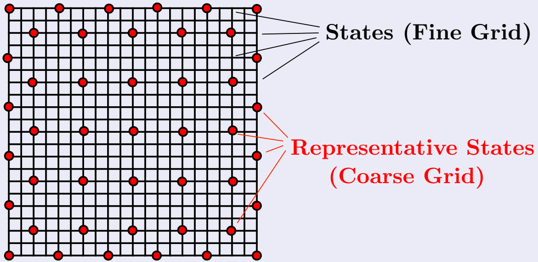
Parametric approximation

Aggregation

- Aggregation is a form of problem approximation. We approximate our DP problem with a "smaller/easier" version, which we solve optimally to obtain $\tilde{J}$.
- Is related to feature-based parametric approximation (e.g., when $\tilde{J}$ is piecewise constant, the features are 0-1 membership functions).
- Can be combined with (global) parametric approximation (like a neural net) in two ways. Either use the neural net to provide features, or add a local parametric correction to a $\tilde{J}$ obtained by a neural net.
- Several versions: multistep lookahead, finite horizon, etc ...

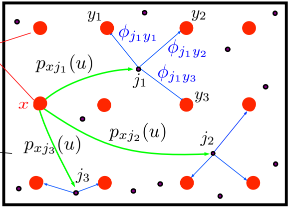
Illustration: A Simple Classical Example of Approximation

Approximate the state space with a coarse grid of states



- Introduce a “small" set of “representative" states to form a coarse grid.
- Approximate the original DP problem with a coarse-grid DP problem, called aggregate problem (need transition probs. and cost from rep. states to rep. states).
- Solve the aggregate problem by exact DP.
- “Extend" the optimal cost function of the aggregate problem to an approximately optimal cost function for the original fine-grid DP problem.
- For example extend the solution by a nearest neighbor/piecewise constant scheme (a fine grid state takes the cost value of the “nearest" coarse grid state).

Approximate the Problem by “Projecting" it onto Representative States



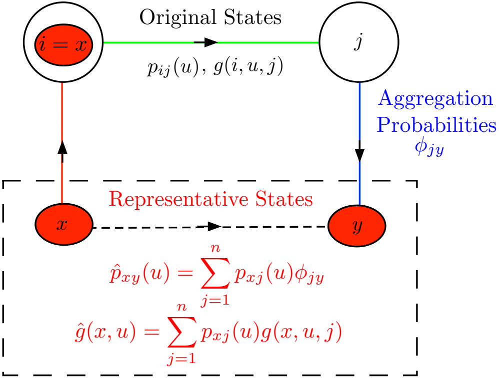
- Introduce a finite subset of “representative states" $\mathcal{A} \subset \{1, \ldots, n\}$. We denote them by x and y.
- Original system states j are related to rep. states $y \in \mathcal{A}$ with aggregation probabilities ϕ_{jy} (“weights" satisfying $\phi_{jy} \geq 0$, $\sum_{y \in \mathcal{A}} \phi_{jy} = 1$).
- Aggregation probabilities express “similarity" or “proximity" of original to rep. states.
- Aggregate dynamics: Transition probabilities between rep. states x, y

$$\hat{p}_{xy}(u) = \sum_{j=1}^{n} p_{xj}(u)\phi_{jy}$$

- Expected cost at rep. state x under control u:

$$\hat{g}(x,u) = \sum_{j=1}^{n} p_{xj}(u) g(x,u,j)$$

The Aggregate Problem

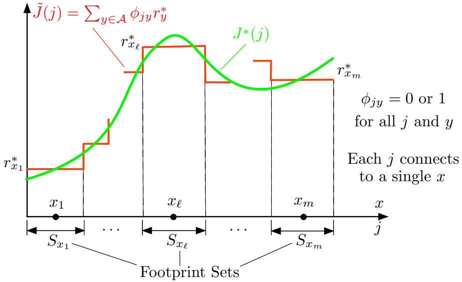


- If r_x^*, $x \in \mathcal{A}$, are the optimal costs of the aggregate problem, approximate the optimal cost function of the original problem by

$$\tilde{J}(j) = \sum_{y \in \mathcal{A}} \phi_{jy} r_y^*, \quad j = 1, \ldots, n, \qquad \text{(interpolation)}$$

- If $\phi_{jy} = 0$ or 1 for all j and y, $\tilde{J}(j)$ is piecewise constant. It is constant on each set

$$S_y = \{j \mid \phi_{jy} = 1\}, \quad y \in \mathcal{A}, \qquad \text{(called the footprint of } y)$$



The approximate cost function $\tilde J = \sum_{y\in\mathcal{A}} \phi_{jy} r_y^*$ is constant within $S_y = \{j \mid \phi_{jy} = 1\}$.

Approximation error for the piecewise constant case ($\phi_{jy} = 0$ or 1 for all j, y)

Consider the footprint sets

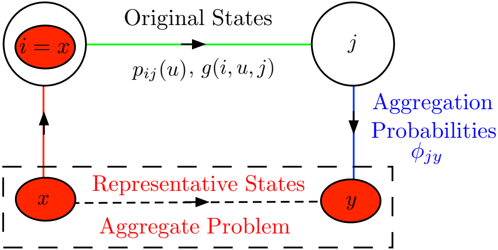
$$S_y = \{j \mid \phi_{jy} = 1\}, \qquad y \in \mathcal{A}$$

The $(J^* - \tilde J)$ error is small if J^* varies little within each S_y. In particular,

$$\left| J^*(j) - \tilde J(j) \right| \le \frac{\epsilon}{1-\alpha}, \qquad j \in S_y,\ y \in \mathcal{A},$$

where $\epsilon = \max_{y\in\mathcal{A}} \max_{i,j\in S_y} \left| J^*(i) - J^*(j) \right|$ is the max variation of J^* within the S_y.

Solution of the Aggregate Problem



Data of aggregate problem (it is stochastic even if the original is deterministic)

$$\hat{p}_{xy}(u) = \sum_{j=1}^{n} p_{xj}(u)\phi_{jy}, \quad \hat{g}(x,u) = \sum_{j=1}^{n} p_{xj}(u)g(x,u,j), \qquad \tilde{J}(j) = \sum_{y\in\mathcal{A}} \phi_{jy} r_y^*$$

Exact methods

Once the aggregate model is computed (i.e., its transition probs. and cost per stage), any exact DP method can be used: VI, PI, optimistic PI, or linear programming.

Model-free simulation methods - Needed for large n, even if model is available

Given a simulator for the original problem, we can obtain a simulator for the aggregate problem. Then use an (exact) model-free method to solve the aggregate problem.

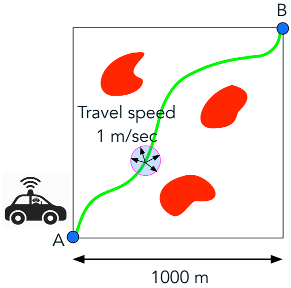
Extension: Continuous State Space - POMDP Discretization

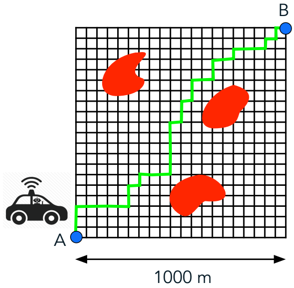
Continuous state space

- The rep. states approach applies with no modification to continuous spaces discounted problems.
- The number of rep. states should be finite.
- The cost per stage should be bounded for the “good"/contraction mapping-based theory to apply to the original DP problem.
- A simulation/model-free approach may still be used for the aggregate problem.
- We thus obtain a general discretization method for continuous-spaces discounted problems.

Discounted POMDP with a belief state formulation

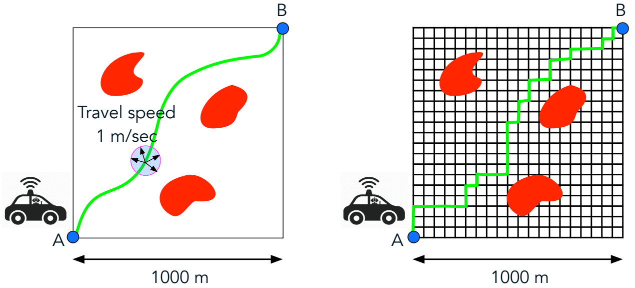
- Discounted POMDP models with belief states, fit neatly into the continuous state discounted aggregation framework.
- The aggregate/rep. states POMDP problem is a finite-state MDP that can be solved for r^* with any (exact) model-based or model-free method (VI, PI, etc).
- The optimal aggregate cost r^* yields an approximate cost function $\tilde{J}(j) = \sum_{y \in \mathcal{A}} \phi_{jy} r_y^*$, which defines a one-step or multistep lookahead suboptimal control scheme for the original POMDP.





Discretizing Continuous Motion

- A self-driving car wants to drive from A to B through obstacles. Find the fastest route.
- Car speed is 1 m/sec in any direction.
- We discretize the space with a fine square grid; restrict directions of motion to horizontal and vertical.
- We take the discretized shortest path solution as an approximation to the continuous shortest path solution.
- Is this a good approximation?



Discretizing Continuous Motion

- The discretization is FLAWED.
- Example: Assume all motion costs 1 per meter, and no obstacles.
- The continuous optimal solution (the straight A-to-B line) has length $\sqrt{2}$ kilometers.
- The discrete optimal solution has length 2 kilometers regardless of how fine the discretization is.
- Here the state space is discretized finely but the control space is not.
- This is not an issue in POMDP (the control space is finite).

From Representative States to Representative Features

The main difficulty with rep. states/discretization schemes:

- It may not be easy to find a set of rep. states and corresponding piecewise constant or linear functions that approximate well J^*.
- Too many rep. states may be required for good approximate costs $\tilde{J}(j)$.

Suppose we have a good feature vector $F(i)$: We discretize the feature space

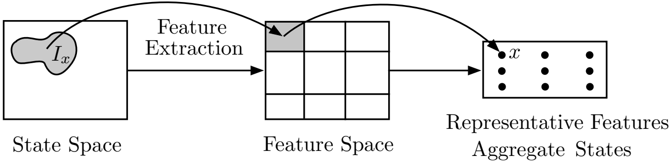
- We introduce representative features that span adequately the feature space

$$\mathcal{F} = \{F(i) \mid i = 1, \ldots, n\}$$

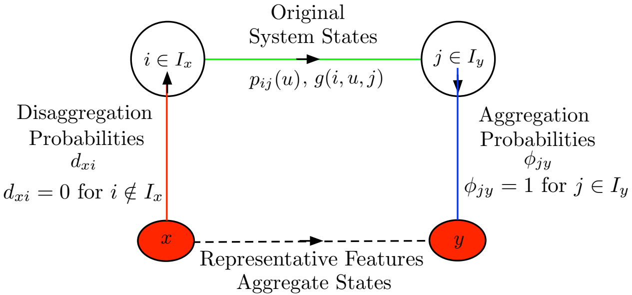
- We aim for an aggregate problem whose states are the rep. features.
- We associate each rep. feature x with a subset of states I_x that nearly map onto feature x, i.e.,

$$F(i) \approx x, \qquad \text{for all } i \in I_x$$

- This is done with the help of weights d_{xi} (called disaggregation probabilities) that are 0 outside of I_x.
- As before, we associate each state j with rep. features y using aggregation probabilities ϕ_{jy}.
- We construct an aggregate problem using d_{xi}, ϕ_{jy}, and the original problem data.

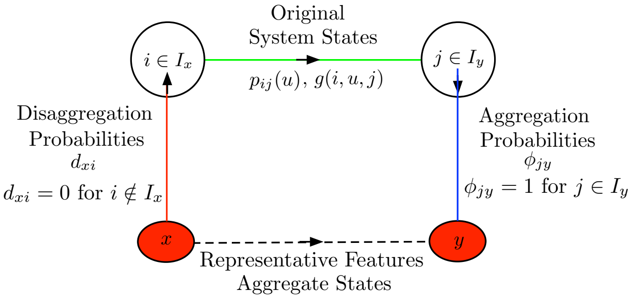


Representative feature formation



Transition diagram for the aggregate problem

Working Break: Feature Formation Methods in Aggregation



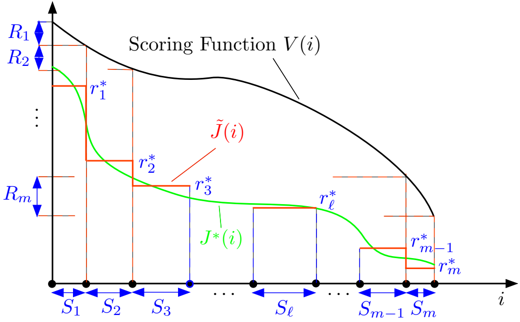
Question 1: Why is the rep. states model a special case of the rep. features model?

Assume the following general principle for feature-based aggregation:

Choose features so that states i with similar features $F(i)$ have similar $J^*(i)$, i.e., $J^*(i)$ changes little within each of the “footprint" sets $I_x = \{i \mid d_{xi} > 0\}$ and $S_y = \{j \mid \phi_{jy} > 0\}$.

Question 2: Can you think of examples of useful features for aggregation schemes?

Feature Formation Using Scoring Functions



Idea: Suppose that we have a scoring function $V(i)$ with $V(i) \approx J^*(i)$. Then group together states with similar score.

- We partition the range of values of V into m disjoint intervals $R_1, \ldots, R_m$.
- We define a feature vector $F(i)$ according to

$$F(i) = \ell, \qquad \text{all } i \text{ such that } V(i) \in R_\ell, \quad \ell = 1, \ldots, m$$

- Defines a partition of the state space into the footprints $S_\ell = I_\ell = \{ i \mid F(i) = \ell \}$.

Examples of Scoring Functions

- Cost functions of heuristics or policies.
- Approximate cost functions produced by neural networks.

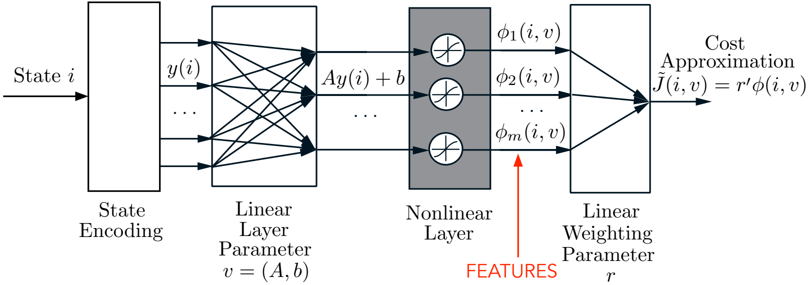
Let the scoring function be the cost function J_μ of a policy μ

Let's compare with rollout:

- Rollout uses as cost approximation $\tilde{J} = J_\mu$.
- Score-based aggregation uses J_μ as scoring function to form features. The resulting $\tilde{J}$ is a "nonlinear function of J_μ" that aims to approximate J^*.
- If the scoring function quantization were so fine as to have a single feature value per interval R_ℓ, we would have $\tilde{J} = J^*$ (much better than rollout).
- Score-based aggregation can be viewed as a more sophisticated form of rollout.
- Score-based aggregation is more computation-intensive, less suitable for on-line implementation.

It is possible to use multiple scoring functions to generate more complex feature maps.

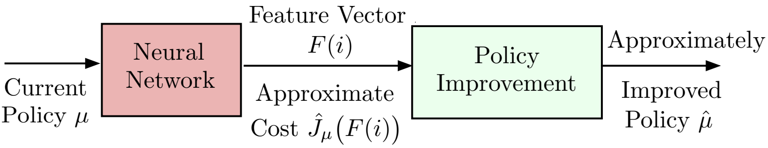
Feature Formation Using Neural Networks

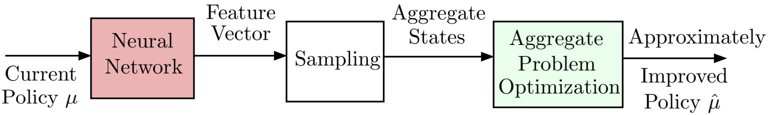


Suppose we have trained a NN that provides an approximation $\hat{J}(i) = r'\phi(i, v)$

- Features from the NN can be used to define rep. features.
- Training of the NN yields lots of state-feature pairs.
- Rep. features and footprint sets of states can be obtained from the NN training set data, perhaps supplemented with additional (state,feature) pair data.
- NN features may be supplemented by handcrafted features.
- Feature-based aggregation yields a nonlinear function $\tilde{J}$ of the features that approximates J^* (not $\hat{J}$).

Policy Iteration with Neural Nets, and Feature-Based Aggregation

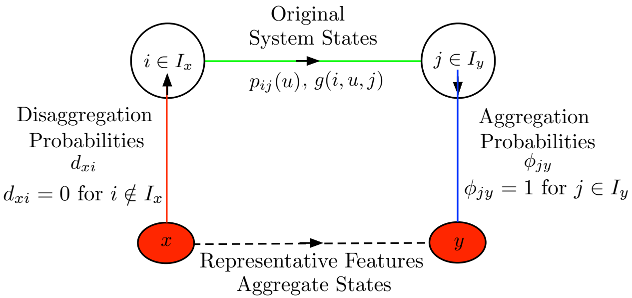




Several options for implementation of mixed NN/aggregation-based PI

- The NN-based feature construction process may be performed multiple times, each time followed by an aggregate problem solution that constructs a new policy.
- Alternatively: The NN training and feature construction may be done only once with some “good" policy.
- After each cycle of NN-based feature formation, we may add problem-specific handcrafted features, and/or features from previous cycles.
- Note: Deep NNs may produce fewer and more sophisticated final features

A Simple Version of the Aggregate Problem



Patterned after the simpler rep. states model.

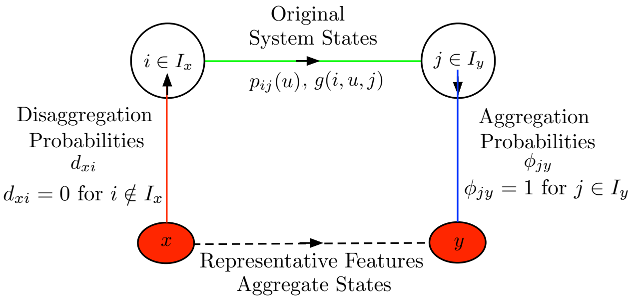
Aggregate dynamics and costs

- Aggregate dynamics: Transition probabilities between rep. features x, y

$$\hat{p}_{xy}(u) = \sum_{i \in I_x} d_{xi} \sum_{j=1}^{n} p_{ij}(u)\phi_{jy}$$

- Expected cost per stage:

$$\hat{g}(x,u) = \sum_{i \in I_x} d_{xi} \sum_{j=1}^{n} p_{xj}(u) g(x,u,j)$$

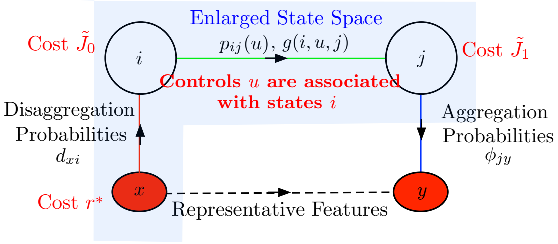


There is an implicit assumption in the aggregate dynamics and cost formulas

$$\hat{p}_{xy}(u) = \sum_{i \in I_x} d_{xi} \sum_{j=1}^{n} p_{ij}(u)\phi_{jy}, \qquad \hat{g}(x,u) = \sum_{i \in I_x} d_{xi} \sum_{j=1}^{n} p_{xj}(u) g(x,u,j)$$

For a given rep. feature x, the same control u is applied at all states i in the footprint I_x.

So the simple aggregate problem is legitimate, but the approximation $\tilde{J}$ of J^* may not be very good. We will address this issue in the next lecture.



Bellman equations for the enlarged problem

$$r_x^* = \sum_{i=1}^n d_{xi}\tilde{J}_0(i), \qquad x \in \mathcal{A},$$

$$\tilde{J}_0(i) = \min_{u \in U(i)} \sum_{j=1}^n p_{ij}(u)\big(g(i,u,j) + \alpha \tilde{J}_1(j)\big), \qquad i = 1, \ldots, n,$$

$$\tilde{J}_1(j) = \sum_{y \in \mathcal{A}} \phi_{jy} r_y^*, \qquad j = 1, \ldots, n$$

r^* solves uniquely the composite Bellman equation $r^* = Hr^*$:

$$r_x^* = (Hr^*)(x) = \sum_{i=1}^n d_{xi} \min_{u \in U(i)} \sum_{j=1}^n p_{ij}(u)\left(g(i,u,j) + \alpha \sum_{y \in \mathcal{A}} \phi_{jy}\, r_y^*\right), \qquad x \in \mathcal{A}$$

Error Bound

Approximation error for the piecewise constant case ($\phi_{jy} = 0$ or 1 for all j, y)

Consider the footprint sets

$$S_y = \{ j \mid \phi_{jy} = 1 \}, \qquad y \in \mathcal{A}$$

The $(J^* - \tilde{J})$ error is small if J^* varies little within each S_y. In particular,

$$\left| J^*(j) - r_y^* \right| \le \frac{\epsilon}{1 - \alpha}, \qquad j \in S_y, \; y \in \mathcal{A},$$

where $\epsilon = \max_{y \in \mathcal{A}} \max_{i,j \in S_y} \left| J^*(i) - J^*(j) \right|$ is the max variation of J^* within S_y.

Implication

Choose representative features x so that J^* varies little over the footprint of x.

This is a generally valid qualitative guideline

Holds for the more general piecewise linear interpolation case.

Simulation-Based Asynchronous Value Iteration for the Aggregate Problem

A sampled version of VI for solving $r^* = Hr^*$: $r^{k+1} \approx (1 - \gamma^k) r^k + \gamma^k H(r^k)$ with

$$(Hr)(x) = \sum_{i=1}^{n} d_{xi} \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + \alpha \sum_{y \in \mathcal{A}} \phi_{jy}\, r_y \right), \qquad x \in \mathcal{A}$$

Note that H is a contraction.

At time k iterate for a single rep. feature x_k, and keep all other r_x^k unchanged:

$$r_{x_k}^{k+1} = (1 - \gamma^k) r_{x_k}^k + \gamma^k \min_{u \in U(i)} \sum_{j=1}^{n} p_{i_k j}(u) \left(g(i_k, u, j) + \alpha \sum_{y \in \mathcal{A}} \phi_{jy} r_y^k \right)$$

where i_k is a sample from I_{x_k} selected according to $d_{x_k i}$, and γ^k is a stepsize.

Convergence result [Tsitsiklis and Van Roy (1995)]

With $\gamma^k \to 0$ and other technical conditions, this iteration converges to the unique solution r^*. Some similarity with (exact) Q-learning proofs.

Simulation-Based Policy Iteration

Uses policy evaluation/policy improvement to generate policy/cost pairs $\{(\mu^k, r^k)\}$.
Converges monotonically ($r^{k+1} \leq r^k$) and finitely ($r^k = r^*$ for sufficiently large k).

Policy evaluation of current policy μ^k

Solve the (linear) composite Bellman equation $r^k = H_{\mu^k} r^k$ for μ^k, where

$$(H_{\mu^k} r)(x) = \sum_{i=1}^{n} d_{xi} \sum_{j=1}^{n} p_{ij}\big(\mu^k(i)\big) \left(g\big(i, \mu^k(i), j\big) + \alpha \sum_{y \in \mathcal{A}} \phi_{jy}\, r_y \right), \qquad x \in \mathcal{A}$$

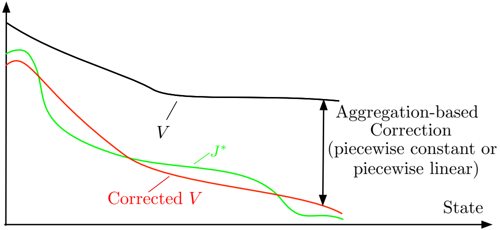
Two possibilities:

- Iteratively: Using a sampled version of VI with sampling for both i and for j.
- By matrix inversion: Write the equation $r^k = H_{\mu^k} r^k$ in matrix form as $r^k = A^k r^k + b^k$. Evaluate A^k and b^k by simulation, and set $r^k = (I - A^k)^{-1} b^k$.

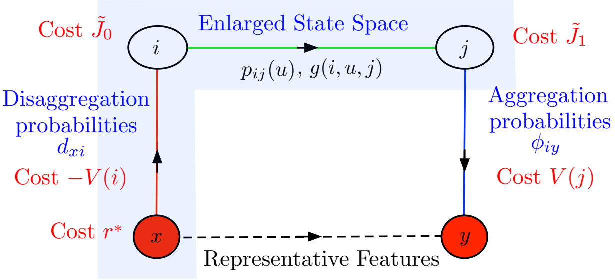
Policy improvement by one-step lookahead

$$\mu^{k+1}(i) = \arg \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u) \left(g(i, u, j) + \alpha \sum_{y \in \mathcal{A}} \phi_{jy} r_y^k \right), \qquad i = 1, \ldots, n$$

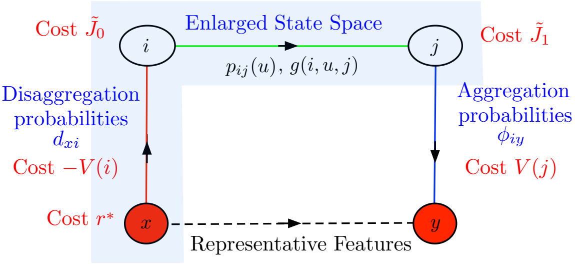
Biased Aggregation - Suppose we Know a Good Approximation $V \approx J^*$; How do we Correct it?



We add a "bias" function V to the cost structure of the enlarged aggregate problem

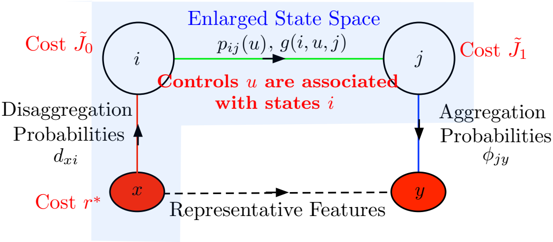


Some Results for Biased Aggregation



Let $(r^*, \tilde{J}_0, \tilde{J}_1)$ be the solution [note that $\tilde{J}_1(j) = V(j) + \sum_{y\in\mathcal{A}} \phi_{jy} r^*_y$]

- When $V = J^*$ then $r^* = 0$, $\tilde{J}_0 = \tilde{J}_1 = J^*$, and any optimal policy for the aggregate problem is optimal for the original problem.
- When $V = J_\mu$ for some policy μ, the policy produced by aggregation is a rollout policy based on μ, when there is a single rep. feature. Suggests that with multiple rep. features the aggregation/rollout policy should be much better than rollout.
- Error bounds similar to the ones for the case $V = 0$ suggest to choose rep. features and footprint sets within which $V - J^*$ varies little.
- We do not know J^*, but we may use $T^k V$ (k value iterations on V) as an approximation. Then use $V - T^k V$ as a scoring function to form rep. features.



How do VI and PI benefit from the problem being deterministic?

- VI form: $r_{x_k}^{k+1} = (1-\gamma^k) r_{x_k}^k + \gamma^k \min_{u \in U(i)} \sum_{j=1}^n p_{i_k j}(u) \big(g(i_k,u,j) + \alpha \sum_{y \in \mathcal{A}} \phi_{jy} r_y^k \big)$
- Policy evaluation: Solve the composite Bellman equation $r^k = H_{\mu^k} r^k$, where

$$(H_{\mu^k} r)(x) = \sum_{i=1}^n d_{xi} \sum_{j=1}^n p_{ij}(\mu^k(i)) \left(g(i,\mu^k(i),j) + \alpha \sum_{y \in \mathcal{A}} \phi_{jy} r_y \right), \qquad x \in \mathcal{A}$$

- Policy improvement: $\mu^{k+1}(i) = \arg\min_{u \in U(i)} \sum_{j=1}^n p_{ij}(u) \big(g(i,u,j) + \alpha \sum_{y \in \mathcal{A}} \phi_{jy} r_y^k \big)$
- How about using representative states? Possibility of multistep lookahead?

For a deterministic problem, the simulation-based VI and PI are simplified

- The sampled version of VI has the form

$$r_{x_k}^{k+1} = (1-\gamma^k) r_{x_k}^k + \gamma^k \min_{u \in U(i)} \left(g(i_k, u) + \alpha \sum_{y \in \mathcal{A}} \phi_{f(i_k,u)y} r_y^k \right)$$

- No expectation over j is required.
- If representative states are used, there is no need for sampling according to the probabilities $d_{x_k i}$ to obtain i_k (so $\gamma^k \equiv 1$).

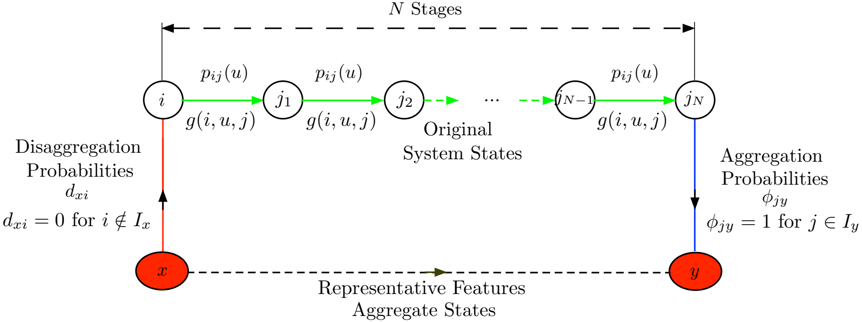
Given r^*, consider ℓ-step lookahead minimization

- At state i_0 we find

$$(u_0^*, \ldots, u_{N-1}^*) \in \arg \min_{(u_0, \ldots, u_{\ell-1})} \left(\sum_{k=0}^{\ell-1} \alpha^k g(i_k, u_k) + \alpha^\ell \sum_{y \in \mathcal{A}} \phi_{i_\ell y} r_y^* \right)$$

  and apply $\tilde{\mu}(i_0) = u_0^*$.
- This is a shortest path problem, and its solution on-line may be fast.

N-Step Feature-Based Aggregation



- The composite system consists of $N + 2$ stochastic Bellman equations.
- Simulation-based version of VI is hard to implement.
- Simulation-based version of PI is possible, but policies are multistep.

A simpler case: Deterministic problem and representative states (no features)

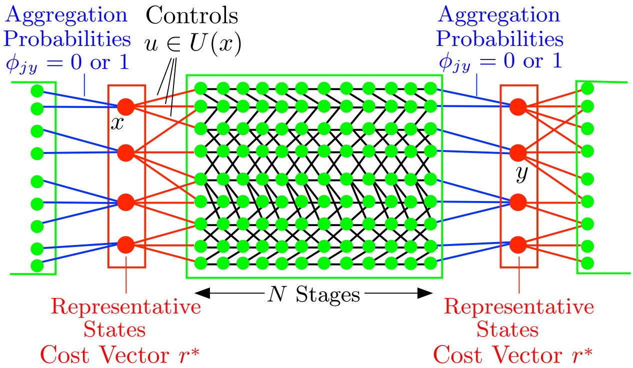
- Then each VI iteration involves solution of an N-stage deterministic DP (shortest path) problem: $r^{k+1} = H_N(r^k)$, where H_N is the N-stage DP operator.
- This algorithm embodies the idea of aggregation in both space and time.

Spatio-Temporal Aggregation - Compressing Space and Time

Plan 5-day auto travel from Boston to San Francisco - How would you do it?

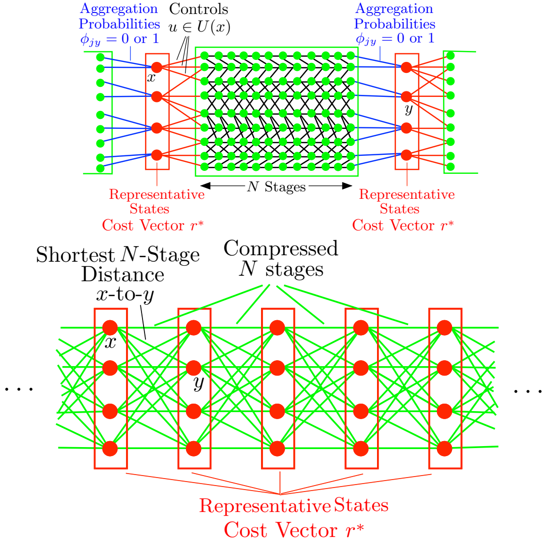
- Select major stops/cities (New York, Chicago, Salt Lake City, Phoenix, etc).
- Select major stopping times (times to stop for sleep, rest, etc).
- Decide on space and time schedules at a coarse level. Optimize the details later.
- We may view this as an example of reduction of a very large-scale shortest path problem to a manageable problem by spacio-temporal aggregation.

Deterministic Problems - N-Stage Aggregation with Representative States and Aggregation Probabilities $\phi_{jy} = 0$ or 1

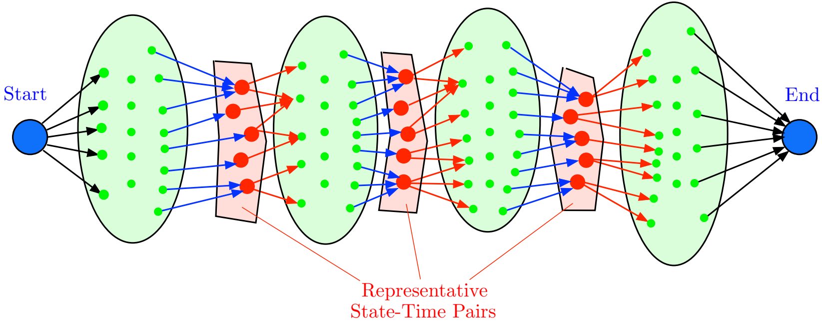


An example of spacio-temporal aggregation

- The infinite horizon discounted aggregate problem decomposes into a sequence of (identical) N-stage shortest path problems.
- Compute shortest path from each rep. state x to each rep. state y.
- Construct a low-dimensional deterministic infinite horizon DP problem (the states are just the representative states).



- Each N-stages block is "compressed" into an all-to-all shortest path problem.
- The compressed problem is a low-dimensional deterministic DP problem.



Deterministic shortest path and finite horizon extensions

- Consider the space-time tube of a deterministic shortest path problem.
- Introduce space-time barriers, i.e., subsets of representative state-time pairs that “separate past from future" (think of the Boston-San Francisco travel).
- “Compress" the portions of the space-time tube between two successive barriers into shortest path problems between each state-time pair of the left barrier to each state-time pair of the right barrier.
- Form a “master" shortest path problem of low dimension that involves only the representative state-time pairs.

About the Next and Final Lecture

WE WILL GIVE AN OVERVIEW OF THE ENTIRE COURSE