Numerical solution of finite state MDPs

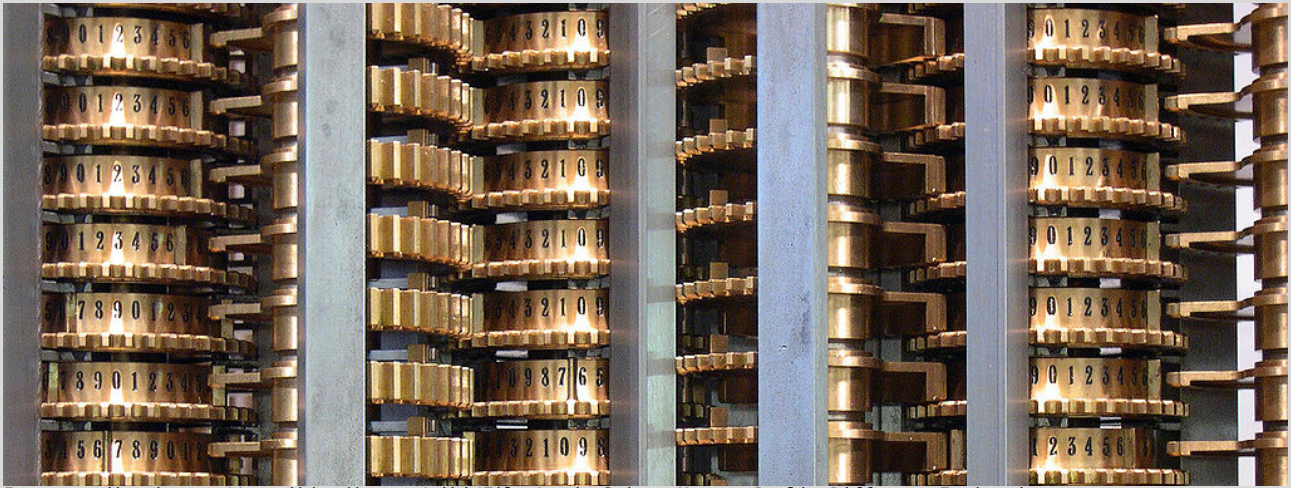

Image credit: https://en.wikipedia.org/wiki/File:LondonScienceMuseumsReplicaDifferenceEngine.jpg

Vectorized representation of finite state MDPs

Assumption (A1) The state space $\mathcal{X}$ and the action space $\mathcal{U}$ are finite. $|\mathcal{X}| = n$ and $|\mathcal{U}| = m$.

Assumption (A2) No constraints, i.e., $\mathcal{U}(x) = \mathcal{U}$.

Controlled transition matrix

$$\mathbf{P}(u) = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}_{n \times n} \quad \text{where } \mathbf{P}_{xy}(u) = \mathbb{P}(X_{t+1} = y \mid X_t = x, U_t = u).$$

$$\mathbf{P} = \begin{bmatrix} P(1) \\ \hline \cdots \\ \hline P(m) \end{bmatrix}_{nm \times n}$$

Cost matrix

$$\mathbf{c} = \begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}_{n \times m} \text{ and } \mathbf{c}_T = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix}_{n \times 1}.$$

Vectorized representation of DP recursion

DP Recursion

$$V_t(x) = \min_{u \in \mathcal{U}} \Big\{ c(x,u) + \beta\, \mathbb{E}[V_{t+1}(f_t(x,u,W_t))] \Big\}$$

$$= \min_{u \in \mathcal{U}} Q_t(x,u).$$

These updates are also written in operator form:

$$Q_t(\cdot, u) = \mathcal{B}_u V_{t+1}$$

$$V_t = \mathcal{B} V_{t+1}$$

where $\mathcal{B}$ is called Bellman operator.

MATLAB implementation

```
function [vUpdate, gOptimal] = bellmanUpdate(discount, c, P, v)
  Q = c + discount * reshape(P*v, size(c));
  [vUpdate, gOptimal] = min(Q, [], 2);
end
```

Solving finite horizon MDP

Dynamic Program

$$V_T = c_T$$

and for $t = T-1, \ldots, 1$

$$V_t = \mathcal{B} V_{t+1}$$

MATLAB implementation

```
function [v, g] = finiteHorizonMDP(c, P, cT, T)
  (n,m) = size(c);

  % Allocate space for value function and optimal policy
  v = zeros(n,T);
  g = zeros(n,T-1);

  % Initialization
  v(:,T) = cT;

  % Backward recursion
  for t = T-1:-1:1
    [v(:,t), g(:,t)] = bellmanUpdate(1, c, P, v(:, t+1));
  end
end
```

Solving infinite horizon discounted MDP

Dynamic program

Let V^* be the solution of the following fixed point equation.

$$V = \mathcal{B}V$$

Value iteration

- Initialize: $V^{(0)} = 0$
- Recursion: $V^{(n+1)} = \mathcal{B}V^{(n)}$
- Stopping condition: Need to find an approximation, V_ε such that $\|V_\varepsilon - V^*\| \leqslant \varepsilon$, where the sup-norm $\|W\|$ is $\max_{x \in \mathcal{X}} |W_x|$.

Stopping conditions

- Method 1 Stop when $\|\Delta V\| \leqslant \varepsilon(1-\beta)/2\beta$, where $\Delta V = V^{(n+1)} - V^{(n)}$. Set $V_\varepsilon = V^{(n)}$.
- Method 2 Stop when $\|\Delta V\|_{sp} \leqslant \varepsilon(1-\beta)/2\beta$ where the span-semi-norm $\|W\|_{sp}$ is $\max_{x \in \mathcal{X}} |W_x| - \min_{x \in \mathcal{X}} |W_x|$. Set $V_\varepsilon = V^{(n+1)} + M\beta/(1-\beta)$ where $M = \min_{x \in \mathcal{X}} \Delta V_x$. (For maximization problems, $M = \max_{x \in \mathcal{X}} \Delta V_x$.)

The second stopping condition leads to much quicker convergence than the first.

Solving infinite horizon discounted MDP

Value iteration algorithm

```
function [v, g] = valueIteration(discount, c, P, tolerance, iterations)
  (n,m) = size(c);

  scale = 1.0;
  if discount < 1
     scale = (1-discount)/discount;
  end
  scaledTolerance = scale * tolerance / 2.0;

  function [v,g] = update(v_previous)
    [v,g] = bellmanUpdate(discount, c, P, v_previous);
  end

  function m = spanNorm(v, w)
    z = v-w;
    m = max(z) - min(z);
  end

  v_previous = zeros(n);
  [v, g]  = update(v_previous);

  precision = Inf;
  iterationCount  = 1;
```

```
    while (precision > scaledTolerance && iterationCount < iterations)
        iterationCount = iterationCount + 1;

        v_previous = v;
        [v, g] = update(v_previous);

        precision = spanNorm(v, v_previous);
    end

    printf("Reached precision %e at iteration %d\n",
            2.0*precision/scale, iterationCount);

    % Renormalize v -- See Puterman 6.6.12 for details
    v = v + min(v - v_previous)/scale;
end
```