# INDR 450/550

Spring 2022

Lecture 26: Dynamic Programming Approximations (2)

May 25, 2022

**Fikri Karaesmen**

# Announcements

- Class Exercise at the end of lecture today. If you are participating online, please upload your document under Course Contents/Class Exercises
- Project long description due on May 27
- HW 4 extended due date May 25
- **Please complete online course evaluation**

# The Problem

- Introduction to Stochastic Dynamic Programming in the context of Capacity Allocation with Multi-class Demand
- Optimal policy by recursion
- Extracting the optimal actions: optimal policy
- Approximating the value function, obtaining approximately optimal actions
  - Tail approximations of the value function
  - Combining tail approximations with short-term look-ahead

# Some Ideas from Reinforcement Learning

- Local policy improvement: Multi-step look ahead

- We can get possibly get more accurate estimators of the value function if we look ahead by more than one period.
- The optimal action at time $t-1$ is given by:

$$\max\{p_2 + \hat{w}_{t-2}(x-1), \hat{w}_{t-2}(x)\}$$

We can now get a more accurate (or updated) estimate for $\hat{w}_{t-1}(x)$.

$$\begin{aligned}\hat{w}'_{t-1}(x) \quad = \quad & \hat{q}_{1,t-1}\max\{p_1 + \hat{w}_{t-2}(x-1), \hat{w}_{t-2}(x)\} \\ & +\hat{q}_{2,t-1}\max\{p_2 + \hat{w}_{t-2}(x-1), \hat{w}_{t-2}(x)\} \\ & +\hat{q}_{3,t-1}\hat{w}_{t-2}(x)\end{aligned}$$

# Some Ideas from Reinforcement Learning

- Local policy improvement: Multi-step look ahead

  - And eventually possibly a better optimal action for time $t$
  - The optimal action at time $t$ is now given by:

$$\max\left\{p_2 + \hat{w}'_{t-1}(x-1), \hat{w}'_{t-1}(x)\right\}$$

  - We can also look multiple periods ahead and therefore combine the more accurate short term forecasts with more rough cut long term approximations of the value function.

# Some Ideas from Reinforcement Learning

- If we can observe multiple sample paths of the process and the rewards generated, we can approximate an initial value function.
- We can then improve the policy for the next run an obtain better estimates to the value function.
- We can then keep on estimating the value function, improving the policy, estimating the value function, improving the policy…
- $\varepsilon$-greedy algoritm: does this but not to get trapped in local optima, also takes some suboptimal actions with a small probability $\varepsilon$.
- If the ‘games’ are easily repeated (by simulation for instance), this gives us a general tool. This is not always easy to do in operations practice.

# Some Ideas from Reinforcement Learning

- Approximating the tail of the expected revenue function: $v_{t\text{-}1}(\mathrm{x})$.

- Approximating the tail of the profit function is critical to run a data-dependent dynamic policy. One idea is to use a deterministic approximation.
- Assume that with $t$ periods to go and $x$ seats remaining our point estimates for the total demand are $\hat{d}_1$ and $\hat{d}_2$.
- We can argue that the optimal dynamic policy will find a smart way to satisfy all of the demand from class 1 if possible and some of the demand from class 2 if capacity remains.

# Some Ideas from Reinforcement Learning

- Approximating the tail of the expected revenue function:

  - Here's a deterministic approximation:

$$\tilde{v}_t(x) \approx p_1 \min\{x, \hat{d}_1\} + p_2 \min\{(x - \hat{d}_1)^+, \hat{d}_2\}$$

  - We can now look forward using the current estimates of future demand available. As before the optimal action would be given by:

$$\max\{p_2 + \tilde{v}_{t-1}(x-1), \tilde{v}_{t-1}(x)\}$$

# Some Ideas from Reinforcement Learning

- Here's the approximation:

| | A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **q1** | **q2** | **p1** | **p2** | **q3** | | | | | | | | |
| 2 | 0.2 | 0.7 | 500 | 100 | 0.1 | | | | | | | | |
| 3 | | | | | | | | | | | | | |
| 4 | | | | | | 160 | 250 | 340 | 400 | 400 | | | |
| 5 | **v(x,t)** | | | | | | | | | | | | |
| 6 | **x↓ t→** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 8 | 1 | 0 | 170 | 260 | 340 | 420 | 500 | 500 | 500 | 500 | 500 | 500 | |
| 9 | 2 | 0 | 170 | 340 | 440 | 520 | 600 | 680 | 760 | 920 | 920 | 1000 | |
| 10 | 3 | 0 | 170 | 340 | 510 | 620 | 700 | 780 | 860 | 1020 | 1020 | 1100 | |
| 11 | 4 | 0 | 170 | 340 | 510 | 680 | 800 | 880 | 960 | 1120 | 1120 | 1200 | |
| 12 | 5 | 0 | 170 | 340 | 510 | 680 | 850 | 980 | 1060 | 1220 | 1220 | 1300 | |
| 13 | 6 | 0 | 170 | 340 | 510 | 680 | 850 | 1020 | 1160 | 1320 | 1320 | 1400 | |
| 14 | 7 | 0 | 170 | 340 | 510 | 680 | 850 | 1020 | 1190 | 1420 | 1420 | 1500 | |
| 15 | 8 | 0 | 170 | 340 | 510 | 680 | 850 | 1020 | 1190 | 1520 | 1520 | 1600 | |
| 16 | 9 | 0 | 170 | 340 | 510 | 680 | 850 | 1020 | 1190 | 1530 | 1530 | 1700 | |
| 17 | 10 | 0 | 170 | 340 | 510 | 680 | 850 | 1020 | 1190 | 1530 | 1530 | 1700 | |
| 18 | 11 | 0 | 170 | 340 | 510 | 680 | 850 | 1020 | 1190 | 1530 | 1530 | 1700 | |
| 19 | 12 | 0 | 170 | 340 | 510 | 680 | 850 | 1020 | 1190 | 1530 | 1530 | 1700 | |
| 20 | 13 | 0 | 170 | 340 | 510 | 680 | 850 | 1020 | 1190 | 1530 | 1530 | 1700 | |
| 21 | 14 | 0 | 170 | 340 | 510 | 680 | 850 | 1020 | 1190 | 1530 | 1530 | 1700 | |
| 22 | 15 | 0 | 170 | 340 | 510 | 680 | 850 | 1020 | 1190 | 1530 | 1530 | 1700 | |
| 23 | 16 | 0 | 170 | 340 | 510 | 680 | 850 | 1020 | 1190 | 1530 | 1530 | 1700 | |
| 24 | 17 | 0 | 170 | 340 | 510 | 680 | 850 | 1020 | 1190 | 1530 | 1530 | 1700 | |
| 25 | 18 | 0 | 170 | 340 | 510 | 680 | 850 | 1020 | 1190 | 1530 | 1530 | 1700 | |
| 26 | 19 | 0 | 170 | 340 | 510 | 680 | 850 | 1020 | 1190 | 1530 | 1530 | 1700 | |
| 27 | 20 | 0 | 170 | 340 | 510 | 680 | 850 | 1020 | 1190 | 1530 | 1530 | 1700 | |

# Some Ideas from Reinforcement Learning

- Class Exercise from May 23:

| v(x,t) | | | | | | |
|---|---|---|---|---|---|---|
| x↓ t→ | 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 170 | 260 | 340 | 420 | 500 |
| 2 | 0 | 170 | 340 | 440 | 520 | 600 |
| 3 | 0 | 170 | 340 | 510 | 620 | 700 |
| 4 | 0 | 170 | 340 | 510 | 680 | 800 |
| 5 | 0 | 170 | 340 | 510 | 680 | 850 |
| 6 | 0 | 170 | 340 | 510 | 680 | 850 |

Figure 1: The approximate value function $\tilde{v}_t(x)$

- This is long-term rough approximation but we can use it to determine optimal actions.
- We can do better by using short-term look ahead in combination with the above approximation.

# Some Ideas from Reinforcement Learning

The look-ahead approximation:

(b) Perform a two-step look-ahead to compute the optimal policy with 5 periods to go and 4 items remaining (first find an updated value function $\tilde{v}'_4(x)$ using $\tilde{v}_3(x)$ and then use the updated value function $\tilde{v}'_4(x)$ to find an optimal action with 5 periods to go).

*Solution:* We first find

$$\begin{aligned} \tilde{v}'_4(4) &= q_1 \max(p_1 + \tilde{v}_3(3), \tilde{v}_3(4)) + q_2 \max(p_2 + \tilde{v}_3(3), \tilde{v}_3(4)) + q_3\tilde{v}_3(4)) \\ &= (0.2)(500 + 510) + (0.7)(100 + 510) + (0.1)510 \\ &= 680 \end{aligned}$$

and

$$\begin{aligned} \tilde{v}'_4(3) &= q_1 \max(p_1 + \tilde{v}_3(2), \tilde{v}_3(3)) + q_2 \max(p_2 + \tilde{v}_3(2), \tilde{v}_3(3)) + q_3\tilde{v}_3(3)) \\ &= (0.2)(500 + 440) + (0.7)(100 + 440) + (0.1)510 \\ &= 617 \end{aligned}$$

We can now estimate the update bid price with 5 periods to go and 4 items remaining as: $\tilde{v}'_4(4) - \tilde{v}'_4(3) = 680 - 617 = 63$. Therefore, it is optimal to admit a customer from Class 2.

# Some Ideas from Reinforcement Learning

| v(x,t) | | | | | | |
|---|---|---|---|---|---|---|
| x↓ t→ | 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 170 | 260 | 340 | 420 | 500 |
| 2 | 0 | 170 | 340 | 440 | 520 | 600 |
| 3 | 0 | 170 | 340 | 510 | 620 | 700 |
| 4 | 0 | 170 | 340 | 510 | 680 | 800 |
| 5 | 0 | 170 | 340 | 510 | 680 | 850 |
| 6 | 0 | 170 | 340 | 510 | 680 | 850 |

Figure 1: The approximate value function $\tilde{v}_t(x)$

- We are using some long term approximation but improving on it by using short term dynamics. We expect that we have some accurate estimation of the model parameters for the short term.
- We can look ahead multiple-periods (more than 2)
  - This combines the recursive solution with a tail approximation of the value function for the future.

# Some Ideas from Reinforcement Learning

- But it’s not always to possible to get a simple deterministic approximation (we may not know the optimal policy and not understand the non-trivial tradeoffs).
- It’s then useful to try feature-based approximations (for instance, non-linear regressions)

- Inputs: $a_1,\ a_2,\ a_3, \ldots$

$$v_t(x) = \beta_0 + f(a_1, a_2..) + \epsilon_t$$

- For instance:

$$v_t(x) = \beta_0 + \beta_1 a_1 + \beta_2 a_2 + \beta_3 a_1^2 + \beta_4 a_1 a_2 + \beta_5((a_2 - I)^+)^2 + \epsilon_t$$

# Some Ideas from Reinforcement Learning

- For instance, for the RM example:

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | **q1** | **q2** | **p1** | **p2** | **q3** |
| 2 | 0.2 | 0.7 | 500 | 100 | 0.1 |
| 3 | | | | | |

- We can test the following features:
- $x$, $t$, $p_1$,$p_2$, $q_1t$, $q_2t$, $p_1q_1t$, $p_2q_2t$ .

# Some Ideas from Reinforcement Learning

- We generate a training sample using some different values for $p_1$,$p_2$, $q_1$, $q_2$ .
- Compute (or estimate the value function) for the training sample
- Fit a non-linear regression to the value functions using (for $p_1$,$p_2$, $q_1$, $q_2$) and some additional derived predictors.

# Some Ideas from Reinforcement Learning

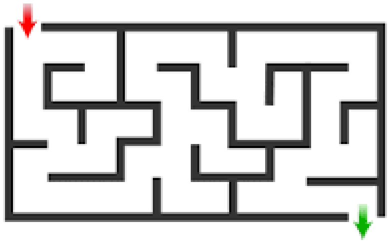
- Here's an example of a fit:

| | | | |
|---|---|---|---|
| **Dep. Variable:** | v | **R-squared:** | 0.821 |
| **Model:** | OLS | **Adj. R-squared:** | 0.818 |
| **Method:** | Least Squares | **F-statistic:** | 378.6 |
| **Date:** | Wed, 18 May 2022 | **Prob (F-statistic):** | 6.21e-152 |
| **Time:** | 17:07:10 | **Log-Likelihood:** | -3091.8 |
| **No. Observations:** | 420 | **AIC:** | 6196. |
| **Df Residuals:** | 414 | **BIC:** | 6220. |
| **Df Model:** | 5 | | |
| **Covariance Type:** | nonrobust | | |

| | **coef** | **std err** | **t** | **P>\|t\|** | **[0.025** | **0.975]** |
|---|---|---|---|---|---|---|
| **Intercept** | -0.0483 | 0.009 | -5.453 | 0.000 | -0.066 | -0.031 |
| **x** | 55.8989 | 3.091 | 18.083 | 0.000 | 49.822 | 61.975 |
| **t** | -19.1327 | 13.234 | -1.446 | 0.149 | -45.147 | 6.882 |
| **p1** | 0.1044 | 0.214 | 0.488 | 0.626 | -0.316 | 0.525 |
| **p2** | -4.8348 | 0.887 | -5.453 | 0.000 | -6.578 | -3.092 |
| **q1t** | -30.6307 | 21.179 | -1.446 | 0.149 | -72.262 | 11.000 |
| **q2t** | 0.0092 | 0.002 | 4.260 | 0.000 | 0.005 | 0.013 |
| **p1q1t** | 0.8376 | 0.071 | 11.873 | 0.000 | 0.699 | 0.976 |
| **p2q2t** | 0.9213 | 0.216 | 4.260 | 0.000 | 0.496 | 1.346 |

$$\hat{v}_t(x) = -0.0483 + 55.8989x - 19.1327t + 0.1044p_1 + \cdots$$

# Maze



| | | |
|---|---|---|
| 7 | 8 | 9 |
| 6 | 5 | 4 |
| 1 | 2 | 3 |

- Starting from the lower left corner, we would like to leave the maze at the upper right corner.
- We can choose to go up, down, left or right. If we choose one of these movement actions, we incur a cost.
- We can choose to stay still (no movement) and incur zero cost.
- There is a reward $r$ at the upper right corner. When we reach there, we receive the reward and are sent back to the beginning.
- Find the optimal policy to maximize the total reward over $T$ periods.

# Maze

- Easy shortest path problem if there is no randomness.
- More complicated if our movements are random.
- We’ll assume that when we choose a movement action, we may end up at any direction with some non-negative probability.
- For instance, if we take action ‘right’ at position 1, we can end up at position 6 but also at positions 1 (left or down) and 2 (right).

# Maze

- We'll count time backward as usual.
- $v_0(x)=0$ (no future reward if no time remains).
- $v_t(9)=10 + v_{t-1}(1)$ (at state 9, we collect a reward of $r$ and then are sent back to the beginning).
- For the other states, we'll have to write the recursion carefully.
- Five actions (0,up, down, left, right):
  - Assume that if on boundaries we bounce back to the interior
- Let's look at two example cases:

# Maze

- $$\begin{aligned} v_t(8) \quad = \quad & max\{v_{t-1}(8), \\ & -c + (q_{u|u} v_{t-1}(8) + q_{d|u} v_{t-1}(5) + q_{l|u} v_{t-1}(7) + q_{r|u} v_{t-1}(9), \\ & -c + (q_{u|d} v_{t-1}(8) + q_{d|d} v_{t-1}(5) + q_{l|d} v_{t-1}(7) + q_{r|d} v_{t-1}(9), \\ & -c + (q_{u|l} v_{t-1}(8) + q_{d|l} v_{t-1}(5) + q_{l|l} v_{t-1}(7) + q_{r|l} v_{t-1}(9), \\ & -c + (q_{u|r} v_{t-1}(8) + q_{d|r} v_{t-1}(5) + q_{l|r} v_{t-1}(7) + q_{r|r} v_{t-1}(9) \quad \} \end{aligned}$$

If the ‘up’ action is selected and the movement is realized, we bounce back to where we are.
But we can move to all other directions.

| | | |
|---|---|---|
| 7 | 8 | 9 |
| 6 | 5 | 4 |
| 1 | 2 | 3 |

# Maze

- $$
\begin{aligned}
v_t(5) \quad = \quad & max\{v_{t-1}(5), \\
& -c + (q_{u|u} v_{t-1}(8) + q_{d|u} v_{t-1}(2) + q_{l|u} v_{t-1}(6) + q_{r|u} v_{t-1}(4), \\
& -c + (q_{u|d} v_{t-1}(8) + q_{d|d} v_{t-1}(2) + q_{l|d} v_{t-1}(6) + q_{r|d} v_{t-1}(4), \\
& -c + (q_{u|l} v_{t-1}(8) + q_{d|l} v_{t-1}(2) + q_{l|l} v_{t-1}(6) + q_{r|l} v_{t-1}(4), \\
& -c + (q_{u|r} v_{t-1}(8) + q_{d|r} v_{t-1}(2) + q_{l|r} v_{t-1}(6) + q_{r|r} v_{t-1}(4) \quad \}
\end{aligned}
$$

| | | |
|---|---|---|
| 7 | 8 | 9 |
| 6 | 5 | 4 |
| 1 | 2 | 3 |

# Maze

- Here’s a numerical solution

| | | |
|---|---|---|
| 7 | 8 | 9 |
| 6 | 5 | 4 |
| 1 | 2 | 3 |

| pup | pdown | pleft | pright | c | r | qup | qdown | qleft | qright |
|---|---|---|---|---|---|---|---|---|---|
| 0.5 | 0.6 | 0.6 | 0.8 | -1 | 10 | 0.166667 | 0.133333 | 0.133333 | 0.066667 |
| | | | | | | | | | |
| | | | | | 0.373333 | 0.36963 | 0.413395 | 0.398644 | 0.377102 |

v(x,t)

| x↓ t→ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0.296 | 1.070923 | 1.730451 | 2.208092 | 2.567806 | 2.916533 |
| 2 | 0 | 0 | 0 | 0 | 1.466667 | 2.315556 | 2.953963 | 3.404756 | 3.747985 | 4.096104 | 4.59566 |
| 3 | 0 | 0 | 0 | 1 | 1.666667 | 2.272222 | 2.816481 | 3.258994 | 3.679968 | 4.178726 | 4.681769 |
| 4 | 0 | 0 | 4 | 4.666667 | 4.944444 | 5.346296 | 5.655679 | 6.052354 | 6.654812 | 7.212353 | 7.685243 |
| 5 | 0 | 0 | 0 | 3.166667 | 3.511111 | 4.382963 | 4.794951 | 5.122962 | 5.481072 | 6.028445 | 6.54263 |
| 6 | 0 | 0 | 0 | 0 | 1.84 | 2.685185 | 3.367358 | 3.8034 | 4.125087 | 4.459138 | 4.964452 |
| 7 | 0 | 0 | 0 | 4.6 | 5.586667 | 6.28 | 6.529057 | 6.697904 | 6.980192 | 7.577107 | 8.171073 |
| 8 | 0 | 0 | 7 | 7.466667 | 8.015556 | 8.140889 | 8.25359 | 8.541973 | 9.214261 | 9.829396 | 10.3288 |
| 9 | 0 | 10 | 10 | 10 | 10 | 10 | 10.296 | 11.07092 | 11.73045 | 12.20809 | 12.56781 |
| 10 | 0 | | | | | | | | | | |

| x↓ t→ | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3.396362 | 3.913104 | 4.419312 | 4.882591 | 5.315484 | 5.76551 | 6.232471 | 6.710311 | 7.1828 | 7.644134 |
| 2 | 5.117864 | 5.622305 | 6.079204 | 6.507733 | 6.960398 | 7.429907 | 7.909304 | 8.380955 | 8.840635 | 9.299457 |
| 3 | 5.169155 | 5.631536 | 6.076532 | 6.536006 | 7.004094 | 7.476704 | 7.945231 | 8.40683 | 8.868391 | 9.331551 |
| 4 | 8.111015 | 8.524605 | 8.994589 | 9.481606 | 9.96388 | 10.42936 | 10.8804 | 11.33858 | 11.80396 | 12.27399 |
| 5 | 7.038627 | 7.481598 | 7.905263 | 8.368498 | 8.841118 | 9.322085 | 9.790265 | 10.24677 | 10.70647 | 11.16917 |
| 6 | 5.489472 | 5.995818 | 6.449799 | 6.874287 | 7.326906 | 7.797162 | 8.278077 | 8.749923 | 9.208786 | 9.667163 |
| 7 | 8.683483 | 9.102759 | 9.490449 | 9.945352 | 10.43316 | 10.9279 | 11.39957 | 11.84857 | 12.30164 | 12.7639 |
| 8 | 10.72375 | 11.09628 | 11.56247 | 12.06103 | 12.56044 | 13.02839 | 13.47095 | 13.92313 | 14.38721 | 14.86127 |
| 9 | 12.91653 | 13.39636 | 13.9131 | 14.41931 | 14.88259 | 15.31548 | 15.76551 | 16.23247 | 16.71031 | 17.1828 |
| 10 | | | | | | | | | | |

# Maze

- And the optimal actions:

| | | |
|---|---|---|
| 7 | 8 | 9 |
| 6 | 5 | 4 |
| 1 | 2 | 3 |

| a(x,t): | actions | 0,u,d,l,r | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| x↓ t→ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 1 | 0 | 0 | 0 | 0 | 0 | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |
| 2 | 0 | 0 | 0 | 0 | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |
| 3 | 0 | 0 | 0 | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |
| 4 | 0 | 0 | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |
| 5 | 0 | 0 | 0 | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |
| 6 | 0 | 0 | 0 | 0 | r | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |
| 7 | 0 | 0 | 0 | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |
| 8 | 0 | 0 | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | |
|---|---|---|
| 7 | 8 | 9 |
| 6 | 5 | 4 |
| 1 | 2 | 3 |

# Maze

- Why is the maze easier for ‘learning’ than a typical operational problem?
  - If the environment is stationary, it is easier to estimate probabilistic parameters and simulate the ‘game.’
  - There is a stationary optimal policy, for $t$ large enough we take the same actions at each position.

| | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | r | r | r | r | r | r | r | r | r | r | r |
| | u | u | u | u | u | u | u | u | u | u | u |
| | u | u | u | u | u | u | u | u | u | u | u |
| | u | u | u | u | u | u | u | u | u | u | u |
| | u | u | u | u | u | u | u | u | u | u | u |
| | u | u | u | u | u | u | u | u | u | u | u |
| | r | r | r | r | r | r | r | r | r | r | r |
| | r | r | r | r | r | r | r | r | r | r | r |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Final Comments

- Many interesting operational problems to solve using reinforcement learning.
- Challenges:
  - Usually non-stationary environment,
  - complicated dependence on predictors,
  - the predictors may be evolving randomly
  - Real life simulation is usually impossible.,
  - Tail approximations of the value function are crucial and require domain knowledge (we have to guess the approximate behaviour of the optimal policy)

# Final Comments

- Spectacular results are obtained using reinforcement learning
- But these are usually for ‘games’ that can be simulated easily
- Or use rich domain expertise (i.e. Robotics, movement on paths etc.)

- There is a need to strenghen reinforcement learning with domain expertise in operations.