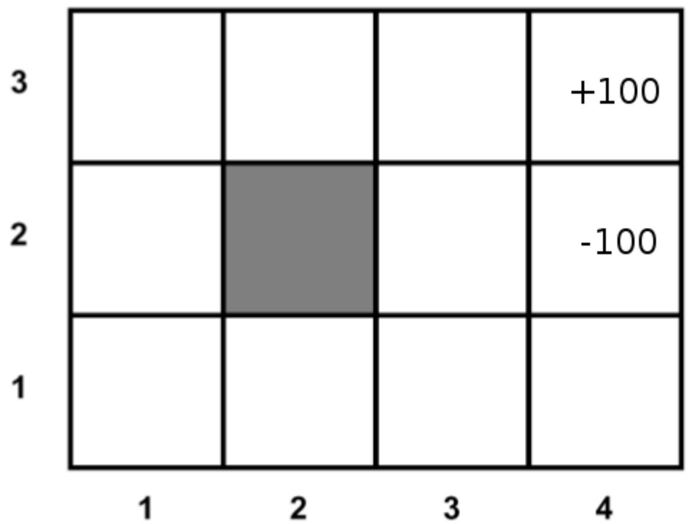
# HW4: Reinforcement Learning

**CS6300: Artificial Intelligence** **Jake Pitkin**
**University of Utah**

## 1 TD and Q in Blockworld

Consider the following gridworld:




Suppose that we run two episodes that yield the following sequences of (state, action, reward) tuples:

| **S** | **A** | **R** | **S** | **A** | **R** |
|---|---|---|---|---|---|
| (1,1) | up | -1 | (1,1) | up | -1 |
| (2,1) | left | -1 | (1,2) | up | -1 |
| (1,1) | up | -1 | (1,3) | right | -1 |
| (1,2) | up | -1 | (2,3) | right | -1 |
| (1,3) | up | -1 | (2,3) | right | -1 |
| (2,3) | right | -1 | (3,3) | right | -1 |
| (3,3) | right | -1 | (4,3) | exit | +100 |
| (4,3) | exit | +100 | (done) | | |
| (done) | | | | | |

1. **According to direct estimation, what are the values for every state in the grid?**

| State | Calculation | Value |
|---|---|---|
| (1,1) | (93 + 95 + 94)/3 | 94 |
| (1,2) | (96 + 95)/2 | 95.5 |
| (1,3) | (97 + 96)/2 | 96.5 |
| (2,1) | 94/1 | 94 |
| (2,2) | - | - |
| (2,3) | (98 + 97 + 98)/3 | 97.666 |
| (3,1) | - | - |
| (3,2) | - | - |
| (3,3) | (99 + 99)/2 | 99 |
| (4,1) | - | - |
| (4,2) | - | - |
| (4,3) | (100 + 100)/2 | 100 |

2. **According to model-based learning, what are the transition probabilities for every (state, action, state) triple. Don't bother listing all the ones that we have no information about.**

| s | a | s′ | T(s, a, s′) |
|---|---|---|---|
| (1,1) | up | (2,1) | 1/3 |
| (1,1) | up | (1,2) | 2/3 |
| (1,2) | up | (1,3) | 1 |
| (1,3) | up | (2,3) | 1 |
| (1,3) | right | (2,3) | 1 |
| (2,1) | left | (1,1) | 1 |
| (2,3) | right | (3,3) | 2/3 |
| (2,3) | right | (2,3) | 1/3 |
| (3,3) | right | (4,3) | 1 |
| (4,3) | exit | (done) | 1 |

3. **Suppose that we run Q-learning. However, instead of initializing all our Q values to zero, we initialize them to some large positive number ("large" with respect to the maximum reward possible in the world: say, 10 times the max reward). I claim that this will cause a Q-learning agent to initially explore a lot and then eventually start exploiting. Why should this be true? Justify your answer in a short paragraph.**

## 2 Policy Gradient

In order to do policy gradient, we need to be able to compute the gradient of the value function $J$ with respect to a parameter vector $\theta$: $\nabla_\theta J(\theta)$. By our algebraic magic, we expressed this as:

$$\nabla_\theta J(\theta) = \sum_a \pi_\theta(s_0, a) R(a) \underbrace{\nabla_\theta \log\left(\pi_\theta(s_0, a)\right)}_{g(s_0,a)} \tag{1}$$

If we us a linear function thrown through a soft-max as our stochastic policy, we have:

$$\pi_\theta(s, a) = \frac{\exp\left(\sum_{i=1}^n \theta_i f_i(s, a)\right)}{\sum_{a'} \exp\left(\sum_{i=1}^n \theta_i f_i(s, a')\right)} \tag{2}$$

**Compute a closed form solution for $g(s_0, a)$.**

We are given that:

$$g(s_0, a) = \nabla_\theta = log\left[\frac{\exp\left(\sum_{i=1}^{n} \theta_i f_i(s,a)\right)}{\sum_{a'} \exp\left(\sum_{i=1}^{n} \theta_i f_i(s,a')\right)}\right]$$

**Explain in a few sentences *why* this leads to a sensible update for gradience ascent (i.e., if we plug this in to Eq (1) and do gradient ascent, why is the derived form reasonable)?**