Stanford University

AA228/CS238: Decision Making under Uncertainty
Autumn 2018

Prof. Mykel J. Kochenderfer • Durand 255 • email: *mykel@stanford.edu*

MIDTERM 3 **Due date: November 30**

You have until 5pm on November 30 to complete this take-home midterm. You may use any resources available to you (e.g., book, notes, internet, software). Although you may discuss the general concepts presented in this class with others, you must not consult with others inside or outside of the class on these specific questions; doing so would be a violation of the Stanford Honor Code. Please upload your responses to this quiz to `gradescope.com`.

Playing Catch Optimally

Mykel and Johnny like to play catch in the evenings. Johnny's utility is related to the distance of successful catches. However, he is uncertain about the relationship between the distances of their attempted catches and the probability of a successful catch, but he does know that the probability of a successful catch is the same regardless of whether he is throwing or catching. Johnny has a finite number of attempted catches to maximize his expected utility before he has to go home and get ready for bed. The following questions build upon each other. Make your code as clear as possible to enhance the probability of partial credit.

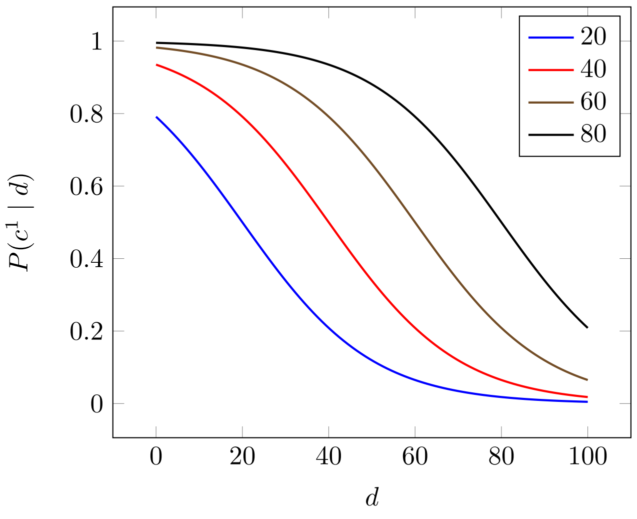
Question 1. Suppose Johnny models the probability of a successful catch at a distance d as follows:

$$P(c^1 \mid d) = 1 - \frac{1}{1+\exp(-\frac{d-\theta}{15})} \tag{1}$$

where θ is an unknown parameter. To keep things manageable, he assumes θ belongs to a discrete set $\Theta = \{20, 40, 60, 80\}$. Create a single plot where the horizontal axis is d from 0 to 100, the vertical axis is $P(c^1 \mid d)$, and different plot lines for each $\theta \in \Theta$. Please also show your code. [2pt]

Solution:

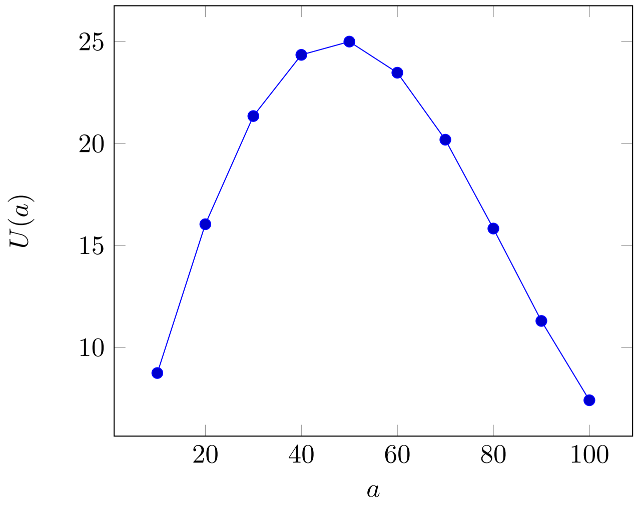
```julia
using LinearAlgebra
using Statistics
Θ = 20:20:80 # possible number of values for θ
prob_catch(d,θ) = 1 - 1/(1+exp(-(d-θ)/15))
plot(Axis([Plots.Linear(d->prob_catch(d,θ),(0,100),
    legendentry="$θ", style="thick") for θ in Θ],
    xlabel=L"d", ylabel=L"P(c^1\mid d)"))
```



Question 2. The reward for a successful catch is equal to the distance. If the catch is unsuccessful, then the reward is zero. Johnny's utility is equal to the sum of the rewards associated with the attempted throws. With each throw, Johnny chooses a distance from a discrete set $A = \{10, 20, \ldots, 100\}$. Initially, Johnny has a uniform distribution over θ. Plot the utility as a function of $a \in A$, assuming a one-step horizon and show your code. [2pt] What is the best action? [1pt]

Solution:

```julia
A = 10:10:100 # possible values for a
R(a, o) = a*o # reward function
b = fill(1/length(Θ), length(Θ)) # initial belief
one_step_value(b, a) = R(a, 1) * dot(b, [prob_catch(a, θ) for θ in Θ])
plot(Axis(Plots.Linear(A, map(a->one_step_value(b,a), A)), xlabel=L"a", ylabel=L"U(a)"))
```

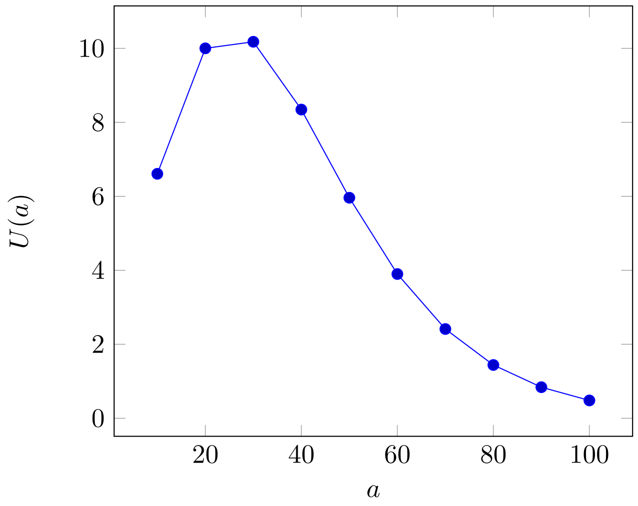


The best action is 50.

Question 3. Replot the one-step horizon from the previous questions, but assume that Johnny is certain that $\theta = 20$, and show your code. [2pt] What is the best action? [1pt]

Solution:

```julia
b = [1.0, 0.0, 0.0, 0.0] # initial belief
plot(Axis(Plots.Linear(A, map(a->one_step_value(b,a), A)), xlabel=L"a", ylabel=L"U(a)"))
```



The best action is 30.

Question 4. Write a function that will update our belief vector b (representing a distribution over the elements in the set Θ) given action a (the distance of the throw) and observation o (which indicates whether the throw was successful). [2pt] Use this function to print the numerical values of Johnny's belief vector after starting with a uniform prior and observing a single successful catch with $a = 30$. [1pt]

Solution:

```
b = fill(1/length(Θ), length(Θ))
prob_o(o, θ, a) = (o == 1) ? prob_catch(a, θ) : 1 - prob_catch(a, θ)
update_belief(b, a, o) = normalize([prob_o(o, θ, a) for θ in Θ] .* b, 1)
@show update_belief(b, 30, 1)
```

```
update_belief(b, 30, 1) = [0.119185, 0.232141, 0.309448, 0.339225]
```

Question 5. Continuing from the previous question, if we start with a uniform prior and then observe four successful catches with $a = 100$, what is the resulting belief vector? Show your code. [1pt]

Solution:

```
let
    b = fill(1/length(Θ), length(Θ));
    for i = 1:4
        b = update_belief(b, 100, 1)
    end
    @show b
end
```

```
b = [2.78782e-7, 5.47444e-5, 0.00931988, 0.990625]
```

Question 6. How many nodes are there in a 4-step policy tree for this problem? [1pt] How many possible 4-step plans are there? [1pt]

Solution: There are 15 nodes and 10^{15} plans.

Question 7. Johnny considers a 4-step policy where the root action is $a = 50$. If an action results in success, then the next action is $a + 10$, otherwise the next action is $a - 10$. Compute the alpha vector associated with this policy and show your code. [2pt]

Solution:

```
function alpha_component(θ, a, k)
    if k == 0
        return 0.0
    end
    p = prob_catch(a, θ)
    return p*(a + alpha_component(θ, a + 10, k - 1)) + (1-p)*alpha_component(θ, a - 10, k - 1)
end
α = [alpha_component(θ, 50, 4) for θ in Θ]
@show α
```

```
α = [31.5621, 69.1415, 121.952, 179.386]
```

Question 8. If Johnny is certain that $\theta = 80$, then what is his expected utility if he follows the 4-step policy in the previous question for 4 steps? [1pt] What if his belief is uniform? [1pt]

Solution: If $\theta = 80$:

```
@show last(α)
```

```
last(α) = 179.38638822051192
```

If uniform:

```
@show mean(α)
```

```
mean(α) = 100.51053280020167
```

Question 9. Use one-step lookahead with the alpha vector computed from his 4-step plan from the previous question to choose an action from an initial belief state $b = [0.5, 0.5, 0, 0]$. Show your code and specify the action. [1pt]

Solution:

```julia
function belief_action_value(b, a, α)
    p = dot([prob_catch(a, θ) for θ in Θ], b) # prob of catch
    b0 = update_belief(b, a, 0)
    b1 = update_belief(b, a, 1)
    return p*(a + dot(b1, α)) + (1-p)*(dot(b0, α))
end
function one_step_lookahead(b, α)
    return argmax(Dict(a=>belief_action_value(b, a, α) for a in A))
end
b = [0.5, 0.5, 0.0, 0.0]
@show one_step_lookahead(b, α)
```

```
one_step_lookahead(b, α) = 30
```

Question 10. Anna goes outside and tells Johnny that he has to come in after exactly 4 throws. She calculates the optimal strategy assuming a uniform prior over Θ. Compute the expected utility of this strategy and show your code. [2pt]

Solution:

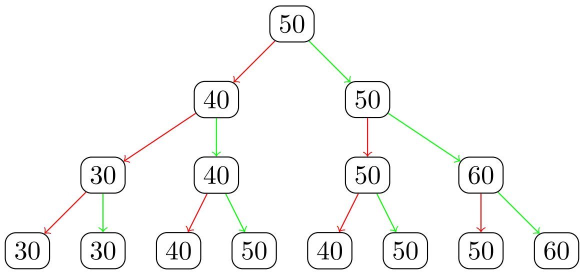
```julia
function belief_action_value(b, a, d)
    if d == 0
        return 0.0
    end
    p = dot([prob_catch(a, θ) for θ in Θ], b) # prob of catch
    b0 = update_belief(b, a, 0)
    b1 = update_belief(b, a, 1)
    v0 = belief_value(b0, d-1)
    v1 = belief_value(b1, d-1)
    return p*(a+v1) + (1-p)*v0
end
belief_value(b, d) = maximum(belief_action_value(b, a, d) for a in A)
b = fill(1/length(Θ), length(Θ))
@show belief_value(b, 4)
```

```
belief_value(b, 4) = 103.28237451968727
```

Question 11. Draw the policy recommended by Anna represented as a policy tree, still assuming an initial uniform prior. [2pt extra credit]

Solution:

```julia
using TikzGraphs
using LightGraphs
function plot_node(b, d, G, nodes, edge_styles)
    a = argmax(Dict(a=>belief_action_value(b, a, d) for a in A))
    push!(nodes, string(a))
    if d < 2
        return
    end
    self = length(nodes)
    add_edge!(G, (self, length(nodes)+1))
    edge_styles[(self, length(nodes)+1)] = "red"
    plot_node(update_belief(b, a, 0), d - 1, G, nodes, edge_styles)
    add_edge!(G, (self, length(nodes)+1))
    edge_styles[(self, length(nodes)+1)] = "green"
    plot_node(update_belief(b, a, 1), d - 1, G, nodes, edge_styles)
end
function plot_policy(b, d)
    G = DiGraph(2^d-1)
    edge_styles = Dict()
    nodes = String[]
    plot_node(b, d, G, nodes, edge_styles)
    TikzGraphs.plot(G, nodes, node_style="draw, rounded corners", edge_styles=edge_styles)
end
p = plot_policy(b, 4)
using TikzPictures
save(PDF("policy"), p)
```



Question 12. Write a little rhyme about what you learned in this class. A selection of clever ones will be compiled and shared with the class. By default they will be shared anonymously, but if you would like your name associated with it, please indicate. [0pt]