Machine Learning: Assignment #4

Due on Thursday, Dey 18th , 1393

Prof. Majid Nili Ahmad Abadi

Nima Mohammadi

Student ID: 610392043
(nima.mohammadi@ut.ac.ir)

Benchmark I: Herd Management

1. For this assignment[1], first we need to show how we are going to calculate $P^a_{ss'}$, which indicates the probability of going from state *s* to *s′* as a result of selecting action *a*.

To compute $P^a_{ss'}$ at first we must generate all the possible 'afterstates' and calculate the likelihood of transitions in form afterstate → state, while preserving the total number of cows in herd of each state. Then we take into account the possibility of states as the result of breeding cows producing infant (young) cows. Then we sum the probability of identical states and merge them.

The function `possible_states_prob(state)` does this operation. This function uses the another function called `burn_offspring(breeding, offspring)` which takes the number of breeding cows as its input parameter and recursively computes and returns the probability of newborn cows. Then these newborn cows are added to the number of youngster cows in previous substates and their probability are multiplied. Special attention is paid such that the states would not surpass the limit imposed on the total number of cows, that is twelve.

As these two functions, used in calculating in $P^a_{ss'}$, are called numerous times in the code, and as their computation is very time consuming, they have been 'cached', so to speak! This approach is called memoization. What we basically did is to use '`container.Map`' data structure to save the output of these functions and use the stored value in case the function is called with the same parameter again.

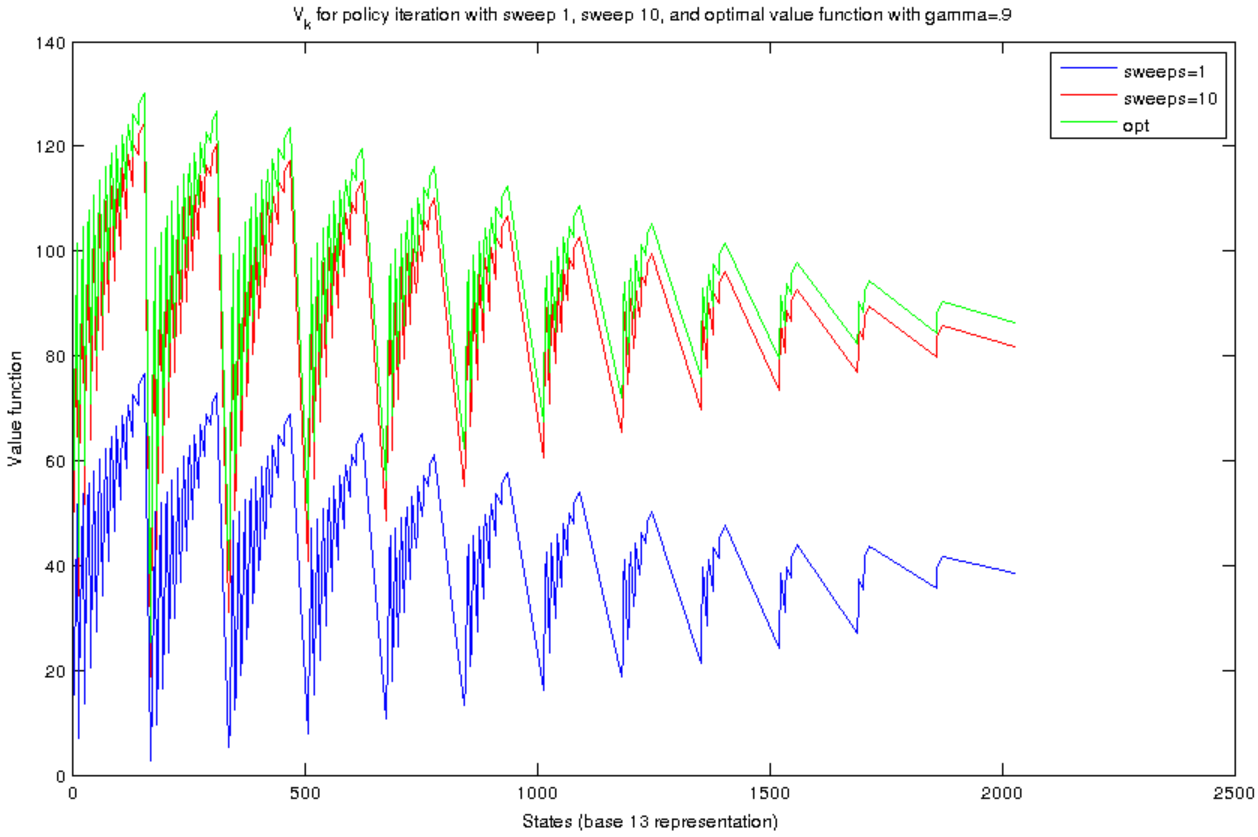
$R^a_{ss'}$, the received reward, is calculated via a function called get_reward(state, action). This function basically computes the reward as below:

$$reward = \sum_{i=1}^{3} [(a_i c_i) + r_i (s_i - a_i)]$$

Where and r_i are c_i the utility and payoff for each type of cow, respectively.

1 1. Disclaimer: I've had many questions regarding this assignment, which although asked by email from the TA, have not been answered! Fortunately I've found the original assignment from E0293 course of Balaraman Ravindran. I wonder why the very helpful hints of the original assignment have not been included in our assignment description! For example, what 'utility' actually is? (milking the cows!). Does the offspring probability show the number of offsprings each single breeding cow may born?

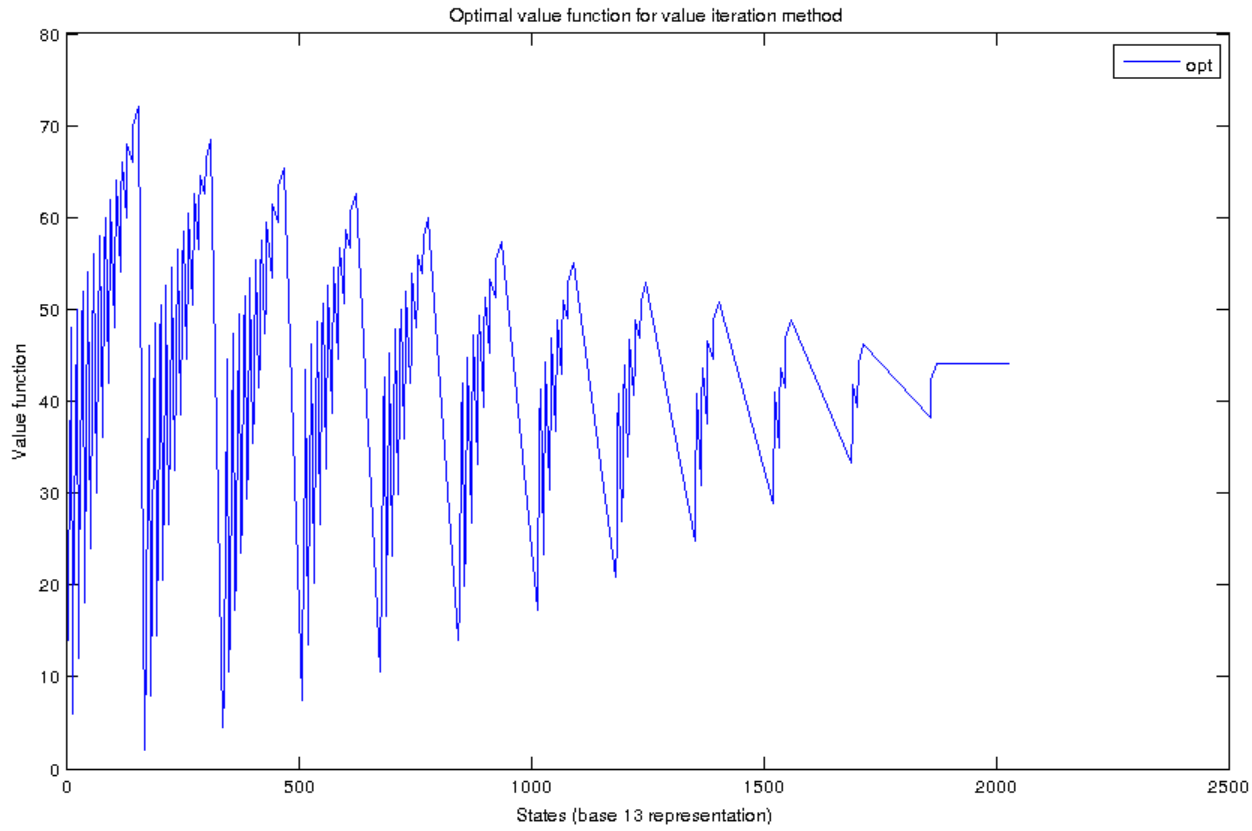
(a) The plot below depicts the value function we got for herd management problem using policy iteration algorithm. The three curves represent the value function for different states obtained using deferent number of sweeps (namely 1, 10 and infinite).



I used a function called `encode_base_13(state)` to transform the vectors representing each state to a scalar. This scalar has been used as the 'key' in my variables of datatype

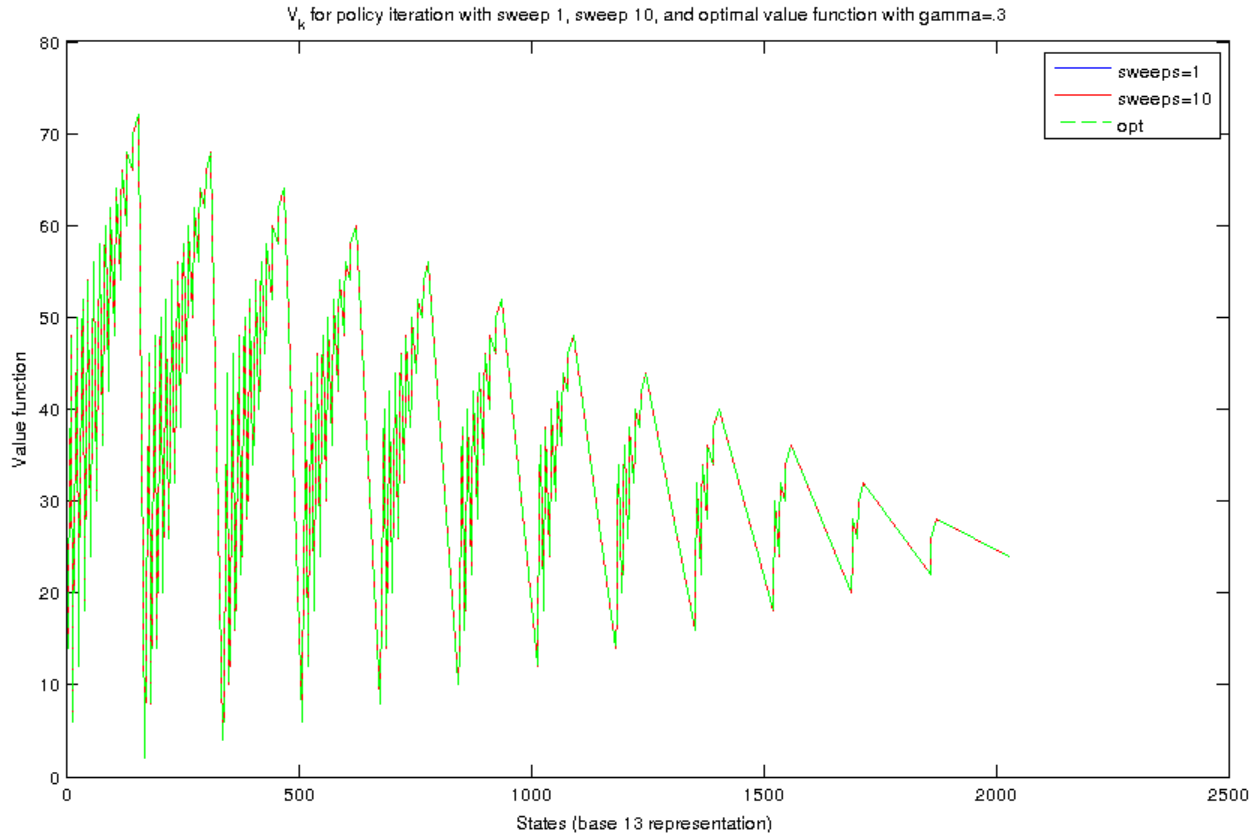
`container.Map`. It has also been used for plotting the value function. In this coding, the young cows represent the most significant digit. The sudden jumps in the plot show the non-existent states, that is the 13-based numbers that do not correspond to a state. You can see that state **(0, 12, 0)** has the highest value, which seems a logical case. Having 12 breeding cows which have highest payoff and utility, naturally is the best situation.

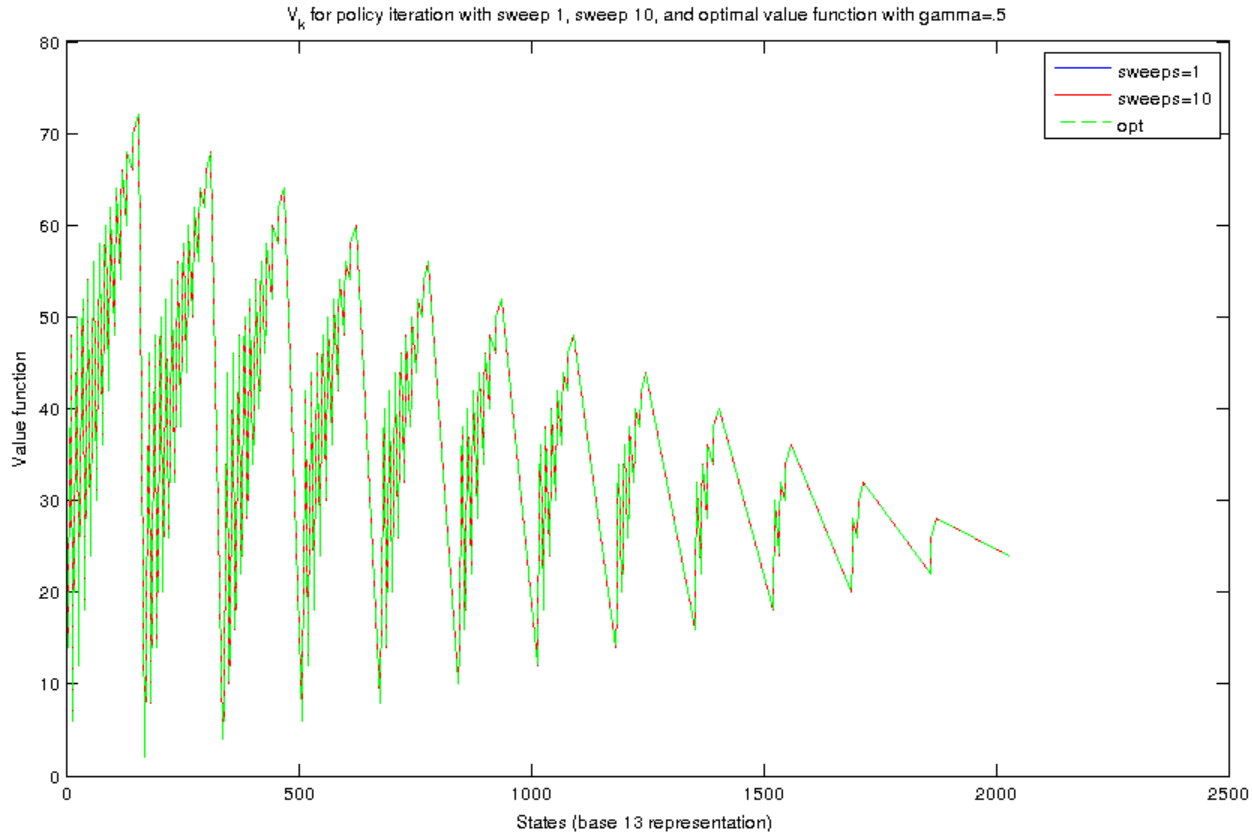
(b) The plot below shows the value function obtained by value iteration method using γ=0.9:



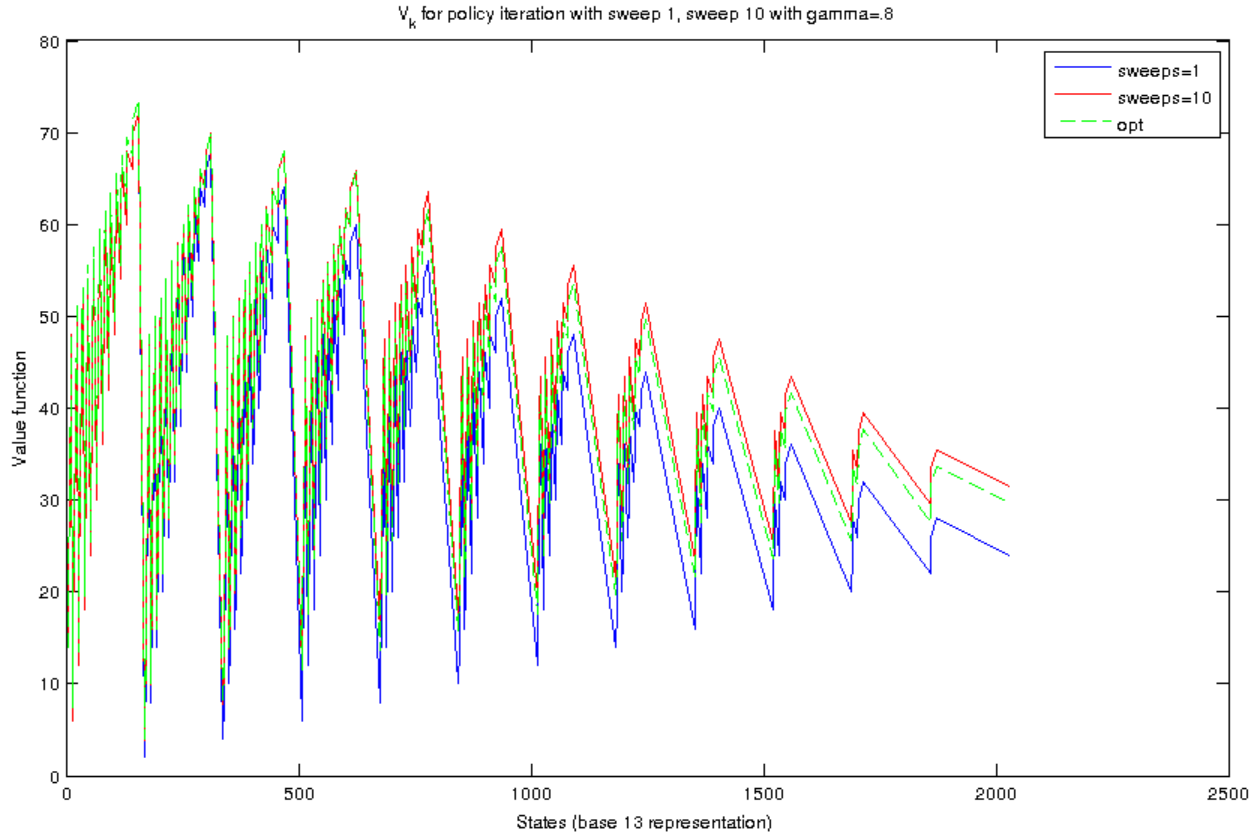
You can see in both plots that having more young cows (the left side of the plot) is better than having the equal number of old cows instead, which seems reasonable.

(c) Both the algorithms with γ=0.3 and γ=0.5 converge to V*(s), even with a single sweep.





It is also interesting that the number of iterations in increases with discount factor. The γ=0.3 and γ=0.5 cases only need one iteration while γ=0.9 requires three iterations. You can say that higher value of discount factor requires more sweeps. To confirm this I also included the results for γ=0.8:



You can see that $V^{10}(s)$ is almost equal to the optimal value function for γ=0.8.

This graph, compared to the graphs of γ=0.3, γ=0.5 and γ=0.9 depicts how the number of necessary sweeps increases with the value of discount factor.

The table below shows the optimal next action selected with optimal value functions learned with different values of γ. From the plots you can see that γ=0.9 gives the highest optimal value function which in turn gives the best next action for these states:

States	**(4, 7, 1)**	**(1, 3, 6)**	**(9, 2, 1)**
γ			
0.9	(0, 3, 1)	(0, 0, 2)	(0, 1, 1)
0.5	(4, 7, 1)	(1, 3, 6)	(9, 2, 1)
0.3	(4, 7, 1)	(1, 3, 6)	(9, 2, 1)