

Robot Learning

Winter Semester 2017/2018, Homework 3

Prof. Dr. J. Peters, D. Tanneberg, M. Ewerton



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Total points: 54 + 8 bonus

Due date: Wednesday, 17 January 2018 (before the lecture)

Name, Surname, ID Number

Problem 3.1 Optimal Control [20 Points]

In this exercise, we consider a finite-horizon discrete time-varying Stochastic Linear Quadratic Regulator with Gaussian noise and time-varying quadratic reward function. Such system is defined as

$$\mathbf{s}_{t+1} = \mathbf{A}_t \mathbf{s}_t + \mathbf{B}_t a_t + \mathbf{w}_t, \quad (1)$$

where \mathbf{s}_t is the state, a_t is the control signal, $\mathbf{w}_t \sim \mathcal{N}(\mathbf{b}_t, \Sigma_t)$ is Gaussian additive noise with mean \mathbf{b}_t and covariance Σ_t and $t = 0, 1, \dots, T$ is the time horizon. The control signal a_t is computed as

$$a_t = -\mathbf{K}_t \mathbf{s}_t + k_t \quad (2)$$

and the reward function is

$$\text{reward}_t = \begin{cases} -(\mathbf{s}_t - \mathbf{r}_t)^\top \mathbf{R}_t (\mathbf{s}_t - \mathbf{r}_t) - a_t^\top \mathbf{H}_t a_t & \text{when } t = 0, 1, \dots, T-1 \\ -(\mathbf{s}_t - \mathbf{r}_t)^\top \mathbf{R}_t (\mathbf{s}_t - \mathbf{r}_t) & \text{when } t = T \end{cases} \quad (3)$$

Note: the notation used in Marc Toussaint's notes "(Stochastic) Optimal Control" is different from the one used in the lecture's slides.

a) Implementation [8 Points]

Implement the LQR with the following properties

$$\begin{aligned} \mathbf{s}_0 &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}) & T &= 50 \\ \mathbf{A}_t &= \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix} & \mathbf{B}_t &= \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} \\ \mathbf{b}_t &= \begin{bmatrix} 5 \\ 0 \end{bmatrix} & \Sigma_t &= \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix} \\ \mathbf{K}_t &= \begin{bmatrix} 5 & 0.3 \end{bmatrix} & k_t &= 0.3 \\ H_t &= 1 & \mathbf{R}_t &= \begin{cases} \begin{bmatrix} 100000 & 0 \\ 0 & 0.1 \end{bmatrix} & \text{if } t = 14 \text{ or } 40 \\ \begin{bmatrix} 0.01 & 0 \\ 0 & 0.1 \end{bmatrix} & \text{otherwise} \end{cases} & \mathbf{r}_t &= \begin{cases} \begin{bmatrix} 10 \\ 0 \end{bmatrix} & \text{if } t = 0, 1, \dots, 14 \\ \begin{bmatrix} 20 \\ 0 \end{bmatrix} & \text{if } t = 15, 16, \dots, T \end{cases} \end{aligned}$$

Execute the system 20 times. Plot the mean and 95% confidence (see "68-95-99.7 rule" and `matplotlib.pyplot.fill_between` function) over the different experiments of the state \mathbf{s}_t and of the control signal a_t over time. How does the system behave? Compute and write down the mean and the standard deviation of the cumulative reward over the experiments. Attach a snippet of your code.

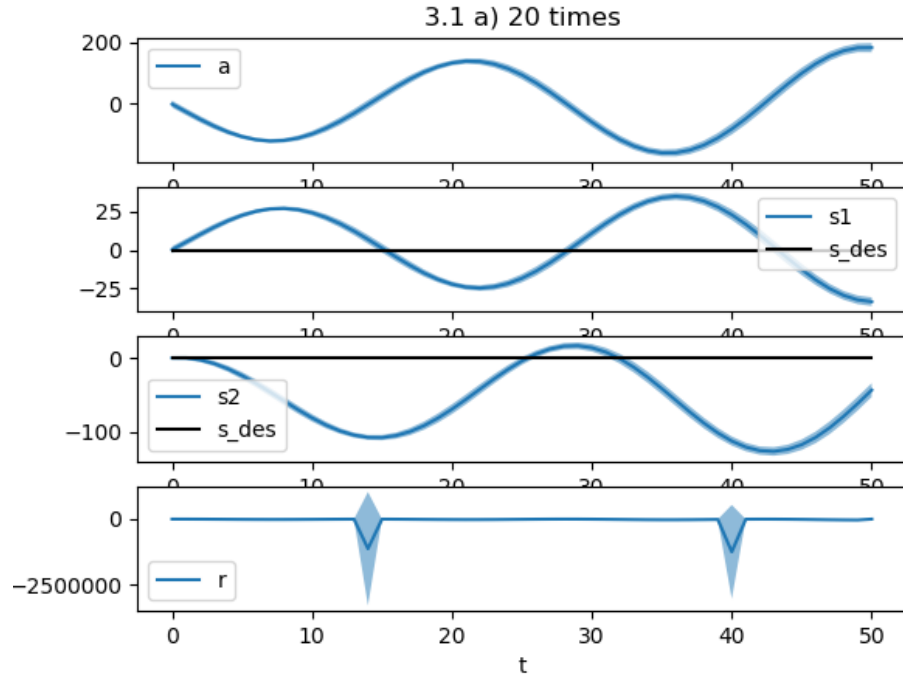


Figure 2: States and action of an instable controller. Mean and 95 % confidence level.

The system is instable cause the eigenvalues of the discrete system (derived from the A-BK matrix) are: $\lambda_i = 0.985 \pm i0.2231i$. The eigenvalues are outside the unit circle. For this reason the system is instable and oscillates (see state s1, s2).

The mean of the cummulative reward over the experiments: $\sum rw = -2717991.1$

The standard deviation of the cummulative reward is: 681486

b) LQR as a P controller [4 Points]

The LQR can also be seen as a simple P controller of the form

$$a_t = K_t (s_t^{\text{des}} - s_t) + k_t, \quad (4)$$

which corresponds to the controller used in the canonical LQR system with the introduction of the target s_t^{des} .

Assume as target

$$s_t^{\text{des}} = r_t = \begin{cases} \begin{bmatrix} 10 \\ 0 \end{bmatrix} & \text{if } t = 0, 1, \dots, 14 \\ \begin{bmatrix} 20 \\ 0 \end{bmatrix} & \text{if } t = 15, 16, \dots, T \end{cases} \quad (5)$$

Use the same LQR system as in the previous exercise and run 20 experiments. Plot in one figure the mean and 95% confidence (see “68–95–99.7 rule” and matplotlib.pyplot.fill_between function) of the first dimension of the state, for both $s_t^{\text{des}} = r_t$ and $s_t^{\text{des}} = 0$.

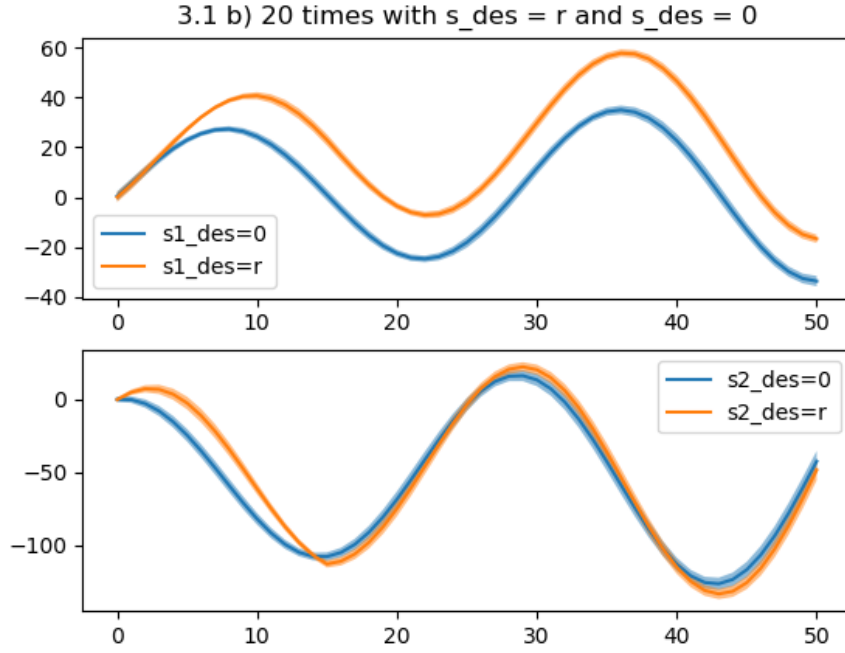


Figure 4: States of an instable controller. Mean and 95 % confidence level. Once with $s_des=0$ and once with $s_des=r$.

Since $t=40$ the state $s1_des=r$ oscillates around 20. Whereas the state $s1_des = 0$ oscillates a little bit lower.

c) Optimal LQR [8 Points]

To compute the optimal gains K_t and k_t , which maximize the cumulative reward, we can use an analytic optimal solution. This controller recursively computes the optimal action by

$$a_t^* = -\left(H_t + B_t^T V_{t+1} B_t\right)^{-1} B_t^T (V_{t+1} (A_t s_t + b_t) - v_{t+1}), \quad (6)$$

which can be decomposed into

$$K_t = -\left(H_t + B_t^T V_{t+1} B_t\right)^{-1} B_t^T V_{t+1} A_t, \quad (7)$$

$$k_t = -\left(H_t + B_t^T V_{t+1} B_t\right)^{-1} B_t^T (V_{t+1} b_t - v_{t+1}). \quad (8)$$

where

$$M_t = B_t \left(H_t + B_t^T V_{t+1} B_t\right)^{-1} B_t^T V_{t+1} A_t \quad (9)$$

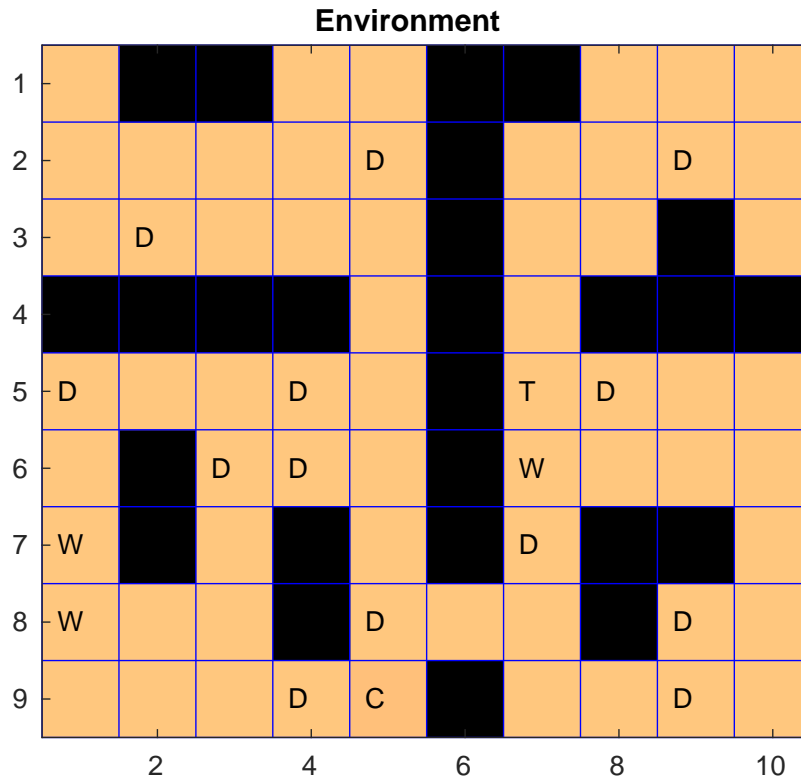
$$V_t = \begin{cases} R_t + (A_t - M_t)^T V_{t+1} A_t & \text{when } t = 1 \dots T-1 \\ R_t & \text{when } t = T \end{cases} \quad (10)$$

$$v_t = \begin{cases} R_t r_t + (A_t - M_t)^T (v_{t+1} - V_{t+1} b_t) & \text{when } t = 1 \dots T-1 \\ R_t r_t & \text{when } t = T \end{cases} \quad (11)$$

Run 20 experiments with $s_t^{\text{des}} = \mathbf{0}$ computing the optimal gains K_t and k_t . Plot the mean and 95% confidence (see “68–95–99.7 rule” and `matplotlib.pyplot.fill_between` function) of both states for all three different controllers used so far. Use one figure per state. Report the mean and std of the cumulative reward for each controller and comment the results. Attach a snippet of your code.

Problem 3.2 Reinforcement Learning [34 Points + 8 Bonus]

You recently acquired a robot for cleaning you apartment but you are not happy with its performance and you decide to reprogram it using the latest AI algorithms. As a consequence the robot became self-aware and, whenever you are away, it prefers to play with toys rather than cleaning the apartment. Only the cat has noticed the strange behavior and attacks the robot. The robot is about to start its day and its current perception of the environment is as following



The black squares denote extremely dangerous states that the robot must avoid to protect its valuable sensors. The reward of such states is set to $r_{\text{danger}} = -10^5$ (NB: the robot can still go through these states!). Moreover, despite being waterproof, the robot developed a phobia of water (W), imitating the cat. The reward of states with water is $r_{\text{water}} = -100$. The robot is also afraid of the cat (C) and tries to avoid it at any cost. The reward when encountering the cat is $r_{\text{cat}} = -3000$. The state containing the toy (T) has a reward of $r_{\text{toy}} = 1000$, as the robot enjoys playing with them. Some of the initial specification still remain, therefore the robot receives $r_{\text{dirt}} = 35$ in states with dirt (D).

State rewards can be collected at every time the robot is at that state. The robot can perform the following actions: down, right, up, left and stay.

In our system we represent the actions with the an ID (0, 1, 2, 3, 4), while the grid is indexed as {row, column}. The robot can't leave the grid as it is surrounded with walls. A skeleton of the gridworld code and some plotting functions are available at the webpage. For all the following questions, always attach a snippet of your code.

a) Finite Horizon Problem [14 Points]

In the first exercise we consider the finite horizon problem, with horizon $T = 15$ steps. The goal of the robot is to maximize the expected return

$$J_\pi = \mathbb{E}_\pi \left[\sum_{t=1}^{T-1} r_t(s_t, a_t) + r_T(s_T) \right], \quad (12)$$

according to policy π , state s , action a , reward r , and horizon T . Since rewards in our case are independent of the action and the actions are deterministic, Equation (12) becomes

$$J_\pi = \sum_{t=1}^T r_t(s_t). \quad (13)$$

Using the Value Iteration algorithm, determine the optimal action for each state when the robot has 15 steps left. Attach the plot of the policy to your answer and a mesh plot for the value function. Describe and comment the policy: is the robot avoiding the cat and the water? Is it collecting dirt and playing with the toy? With what time horizon would the robot act differently in state (9,4)?

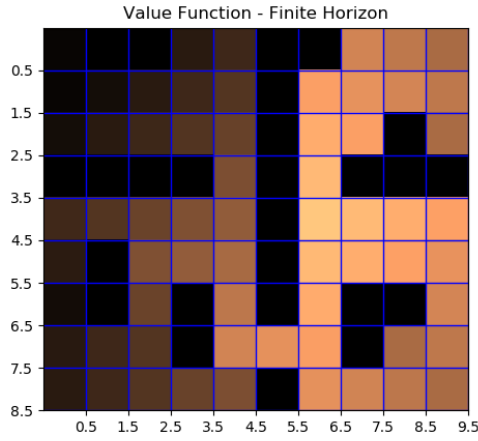


Figure 8: T=15, finite horizon Value Function

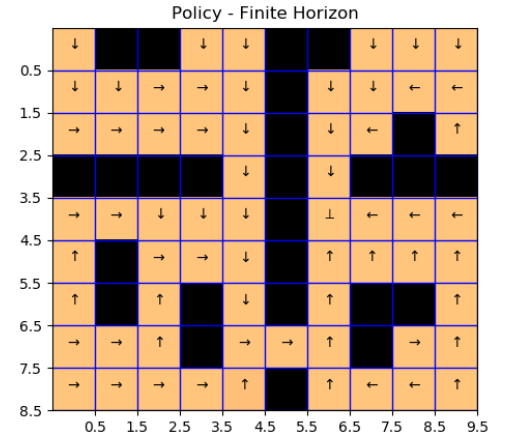


Figure 9: T=15, finite horizon Policy

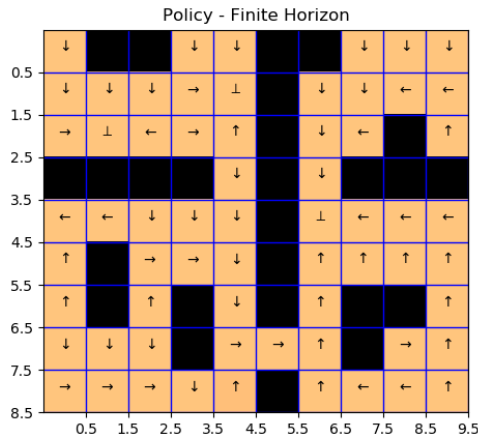


Figure 10: T=10, finite horizon Policy

Policy of the robot:

The robot is trying to avoid the water and the cat (for $T=15$) if possible (see for example state(8,1) the robot is not moving to state 7,1 as this state is punished cause of its water). The robot is moving from all start states to the max. reward position (the toy at 5,7) avoiding all dangerous states. Thereby the robot is not so afraid of small punishments by the cat or water and moves also over these fields in order to reach the toy! Note: At the toy's position the robot earns a reward as long as it stays at this position. For this reason the robot does not move around the cat but moves over the cat's position directly to the toy in order to play with the toy as long as possible. If it is worth it the robot also collects dirt but this is only a minor aim of the robot.

For $T=10$ the robot is not moving to the toy's position, as it can not play long enough with the toy and earn such a high reward that compensates moving over the cat's position. (see Figure 10).

The Algorithmic Description of Value Iteration slide can be found in the lecture. The equation on this slide is used to code the current problem.

b) Infinite Horizon Problem - Part 1 [4 Points]

We now consider the infinite horizon problem, where $T = \infty$. Rewrite Equation (12) for the infinite horizon case adding a discount factor γ . Explain briefly why the discount factor is needed.

Theory:

In the infinite time horizon case (when you live forever), the time index is not part of the state. The optimal policy is time-independent: $\pi_t^*(a|s) = \pi^*(a|s)$. The reward and the transition model can no longer be time-dependent. There is a single stationary value function for all times. There are two different approaches to learning:

- Value Iteration: Same as before just choose a large T for which the value function converges (this is what we use)
- Policy Iteration: Learn a value function for the current policy. Update this policy and learn a new value function. Repeat.

Optimal policy for infinite horizon: $\pi^* = \operatorname{argmax}_{\pi} J_{\pi}, \quad J_{\pi} = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$

In this case here:

$$J_{\pi} = \sum_{t=1}^{\infty} \gamma^t r_t(s_t). \quad (15)$$

The discount factor $0 \leq \gamma < 1$ trades-off long term vs. immediate reward. Without a discount factor many or all policies have infinite expected reward. For this reason a discount factor is required. Thereby future rewards $r_t(s, a)$ are discounted by γ per time step.

c) Infinite Horizon Problem - Part 2 [6 Points]

Calculate the optimal actions with the infinite horizon formulation. Use a discount factor of $\gamma = 0.8$ and attach the new policy and value function plots. What can we say about the new policy? Is it different from the finite horizon scenario? Why?

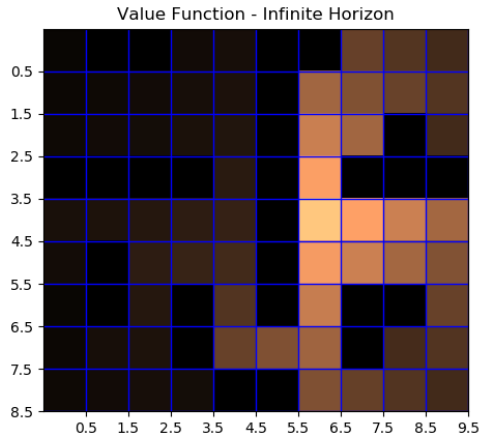


Figure 13: T=100, gamma=0.8 infinite horizon Value Function

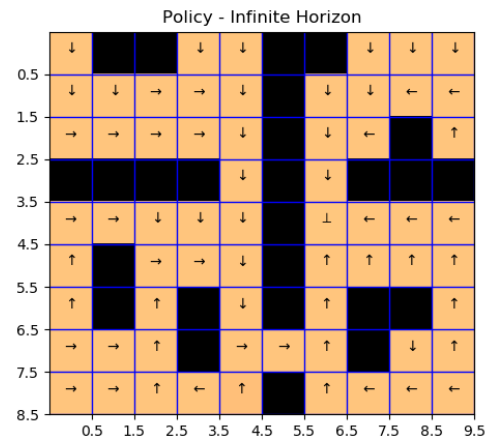


Figure 14: T=100, gamma=0.8, infinite horizon Policy

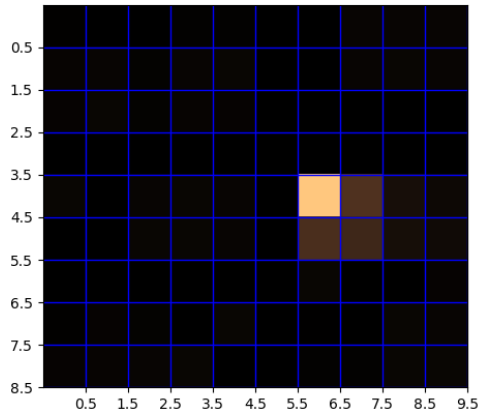
If the robot lives forever it avoids the cat to reach the toy! This is the only huge difference compared to the finite horizon scenario. The reason for this policy is the discount factor. It weights future rewards smaller than immediate rewards. The smaller the discount factor (e.g 0.1) the higher immediate rewards are weighted. The higher the discount factor the more attractive is the toy to the robot.

Another small difference is that if the robot lives forever it collects a little bit more Dirt on its way to the toy position. That is why the policy of state 8,9 changes to a down arrow to collect the Dirt on 9,9 on its way to the toy position.

d) Finite Horizon Problem with Probabilistic Transition Function [10 Points]

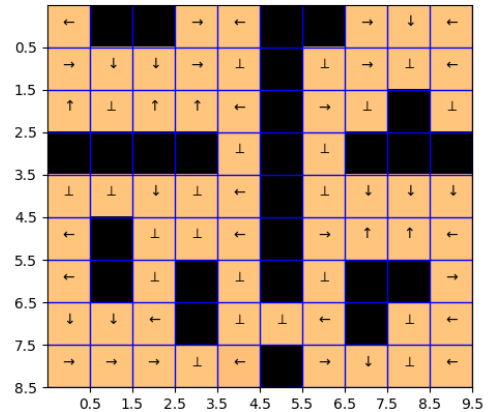
After a fight with the cat, the robot experiences control problems. For each of the actions up, left, down, right, the robot has now a probability 0.7 of correctly performing it and a probability of 0.1 of performing another action according to the following rule: if the action is left or right, the robot could perform up or down. If the action is up or down, the robot could perform left or right. Additionally, the action can fail causing the robot to remain on the same state with probability 0.1. Using the finite horizon formulation, calculate the optimal policy and the value function. Use a time horizon of $T = 15$ steps as before. Attach your plots and comment them: what is the most common action and why does the learned policy select it?

Value Function - Finite Horizon with Probabilistic Transition

Figure 17: $T=15$, finite horizon Value Function

One of the more common action's is to stay at a reward position not risking a wrong movement to a punished position (like Cat or dangerous field). One example is the state at 9,4. At this position the robot does not risk to move wrong in the right direction (into the cat). The learned policy rather let's the robot wait at a certain position than risking to move into a punished state. One reason for that is the quite high probability of moving wrong ($0.1+0.1$).

Policy - Finite Horizon with Probabilistic Transition

Figure 18: $T=15$, finite horizon Policy

e) Reinforcement Learning - Other Approaches [8 Bonus Points]

What are the two assumptions that let us use the Value Iteration algorithm? What if they would have been not satisfied? Which other algorithm would you have used? Explain it with your own words and write down its fundamental equation.

Some Theory:

Reinforcement Learning:

Contains a bunch of methods of machine learning. An agent(robot) learns a policy (Strategie) in order to increase its reward(Belohnung). At some particular states the agent gets a reward. With the knowledge of these rewards the agent optimizes a cost-function in order to learn its optimal actions to increase its reward.

TD-Learning:

Temporal Difference Learning is a method of reinforcement learning. Compared to reinforcement learning the agent adapts after each action its policy according to an estimated reward. In reinforcement learning the agent adapts its policy after it collected some rewards in order to maximize its rewards.

Value Iteration Assumptions:

Assume that the agent(robot) accurately knows the transition function(How the robot can move at each state) and the reward(R matrix) for all states in the environment. This knowledge however can be learned with a method called Q-Learning.

Q-Learning:

Q-Learning is a specific method of TD-Learning in reinforcement learning. Q-learning is a form of model-free learning, meaning that an agent does not need to have any model of the environment; it only needs to know what states exist and what actions are possible in each state. The way this works is as follows: we assign each state an estimated value, called a Q-value. When we visit a state and receive a reward, we use this to update our estimate of the value of that state. (Since our rewards might be stochastic, we may need to visit a state many times.)

Update Equations for learning the Q-Functions:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha \left(r_t + \gamma Q_t(s_{t+1}, a_{\gamma}) - Q_t(s_t, a_t) \right) \quad (19)$$

With:

$$Q-Learning : a_{\gamma} = \operatorname{argmax}_a Q_t(s_{t+1}, a) \quad (20)$$

Sidenote at SARSA:

$$SARSA : a_{\gamma} = a_{t+1} \quad (21)$$

See also: https://de.wikipedia.org/wiki/Temporal_Difference_Learning