The Effects of Large Disturbances on On-Line Reinforcement Learning for a Walking Robot

Erik Schuitema Wouter Caarls Martijn Wisse Pieter Jonker
Robert Babuška

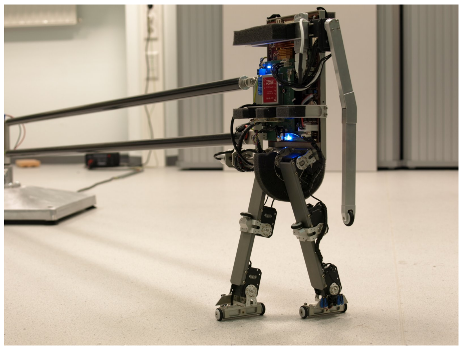
Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands

Abstract

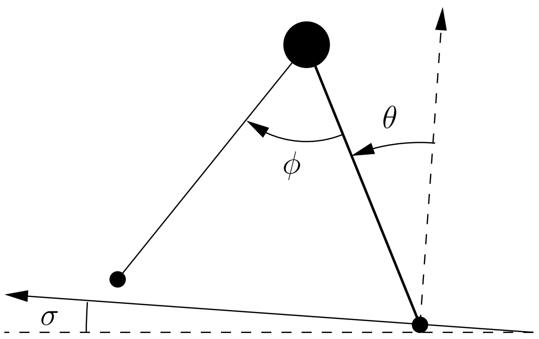
Reinforcement Learning is a promising paradigm for adding learning capabilities to humanoid robots. One of the difficulties of the real world is the presence of disturbances. In Reinforcement Learning, disturbances are typically dealt with stochastically. However, large and infrequent disturbances do not fit well in this framework; essentially, they are outliers and not part of the underlying (stochastic) Markov Decision Process. Therefore, they can negatively influence learning. The main reasons for such disturbances for a humanoid robot are sudden changes in the dynamics (such as a sudden push), sensor noise and sampling time irregularities. We investigate the effects of these types of outliers on the on-line learning process of a simple walking robot simulation. We propose to exclude the outliers from the learning process with the aim to improve convergence and the final solution. While infrequent sensor and timing outliers had a negligible influence, infrequent pushes heavily disrupted the learning process. By excluding the outliers from the learning process, performance was again restored.

1 Introduction

Reinforcement Learning (RL) is an attractive paradigm for adding autonomous learning capabilities to machines such as humanoid robots. By means of interaction with the real world, i.e., trial-and-error learning, the system is capable of learning complex behavior while receiving coarse feedback in terms of positive and negative rewards. An added complexity of real-world systems is the presence of *disturbances*. For a real-time dynamic system such as a humanoid robot (see Figure 1(a)), we have found the main sources of disturbance to be sensor noise, noise on the timing (i.e., sample duration) and changes in the dynamics such as a sudden push or height differences in the floor. Disturbances can be (virtually) instantaneous, such as sensor noise and timing hiccups, or lengthier such as wind, changing floor slope, or (temporary) sensor drift.



(a)



(b)

Figure 1: a. LEO: a 2D walking robot suitable for on-line Reinforcement Learning [8]. b. Simplest walker model, the most elementary model that describes walking behavior.

Within RL, the learning agent and its environment are usually modeled as a Markov Decision Process or MDP. It is common practice for RL algorithms to allow stochastic state transitions (and rewards) in the MDP [3, 6, 11]. Usually, averaging over sufficient experiences will allow the algorithm to find the optimal solution to such a stochastic MDP. However, not all disturbances can be considered to be part of the stochastic nature of the problem. Some disturbances, especially large and infrequent ones, should be considered as *outliers*. While some algorithms – particularly off-line ones – are robust against outliers, to our knowledge no in-depth study has been made of the effect of realistic outliers on learning a control policy for real-time dynamic systems.

We are primarily interested in on-line Temporal Difference (TD) learning, such as Q-learning and SARSA [10], because this class of algorithms requires a minimal amount of prior knowledge and does not restrict the space of possible solutions. This is important for creating learning techniques that make machines truly autonomous, i.e., not dependent on an engineer pre-programming a part of the solution.

Dealing with outliers has two main aspects. The first step is to *detect* the outlier. Once it is detected, the system can *reject* the outlier, excluding it from the learning process. A possible further step is *correction*, i.e., to try and counter the disturbance while it is active, which is only possible when the disturbance has a significant duration. Appropriate correction requires *classification* of the outlier to predict its evolution. Deliberately *including* disturbances in the learning process might result in a solution that is more robust against disturbances. However, this is only possible if they occur often enough; in other words, if they are part of the stochastic nature of the system. The focus of this paper is on the effects of outliers and the added value of rejecting them.

Several techniques have been studied to make RL more robust against disturbances [9, 7, 4, 1]. While this work offers important and useful techniques, the difference in effects of various types of disturbances remains unknown. We explore the influence of several types of outliers on the learning process of a simple simulation[1] model of a walking robot – the simplest walker model [5] – in order to assess the need for further steps such as detection and classification. To illustrate the benefit of being able to detect outliers, we test the effect of excluding the outliers from the learning process – a relatively simple operation in TD learning. We do not look into the ability of the learning agent to withstand stochastic disturbances, e.g. normally distributed disturbances of moderate size; we only consider large, infrequent disturbances that can best be regarded as outliers. We also do not look into slow and permanent changes in the system or its environment, because this requires different properties of the learning algorithm: the ability to adapt to new situations.

This paper is organized as follows. In Section 2, we give a theoretical overview of the Reinforcement Learning framework and discuss the influence of large disturbances in Section 3. In Section 4, we describe the experimental setup (simulation). We present our results in Section 5 and our conclusions in Section 6.

2 Reinforcement learning

In this section, we briefly introduce the Markov Decision Process and the main on-line Temporal Difference learning algorithms Q-learning and SARSA. For a thorough introduction to RL, see, e.g., [10].

Markov Decision Process The common approach in RL is to model the learning system as a Markov Decision Process (MDP) with discrete time steps labeled $k = 0, 1, .. \in \mathbb{Z}$ with sampling period h. The dynamic systems that we are interested in - robotic systems - have a continuous state space S and a continuous action space A. The MDP is defined as the 4-tuple $\langle S, A, T, R\rangle$, where S is a set of states and A is a set of actions. The state transition probability function $T : S \times A \times S \to [0, \infty)$ defines the probability that the next state $s_{k+1} \in S$ belongs to a region $S_{k+1} \subset S$ as $\int_{S_{k+1}} T(s_k, a_k, s')ds'$, when executing action $a_k \in A$ in state $s_k \in S$. The reward function $R : S \times A \times S \to \mathbb{R}$ is real valued and defines the reward of a state transition as $r_{k+1} = R(s_k, a_k, s_{k+1})$. An MDP has the Markov property, which means that T and R only depend on the current state-action pair and not on past state-action pairs nor on information excluded from s.

The goal of the learner is to find the optimal control policy $\pi^* : S \to A$ that maps states to actions and that maximizes, from every initial state s_0, the long term sum of discounted rewards $\mathfrak{R}(s_0) = \sum_{k=0}^{\infty} \gamma^k r_{k+1}$ in which γ is the discount factor. The action-value function or Q-function $Q(s, a)$ gives the estimated return of choosing a_k in s_k and following the control policy afterwards: $Q(s_k, a_k) = \mathrm{E}\{r_{k+1} + \gamma\mathfrak{R}(s_{k+1})\}$.

[1]While in simulation one can choose when and how to apply a disturbance, on a real robot, this happens mostly involuntarily. Therefore, we use simulations throughout this paper to show the effects of several types of large disturbances.

Online RL implies that the system learns from interaction with the real world. In this paper, we assume that the state transition probability function T is unknown to the learning agent (model-free RL). Although there are multiple classes of online RL algorithms, in this paper we focus on the class of (online) Temporal Difference learning algorithms.

Temporal Difference learning Temporal Difference (TD) learning methods mostly aim at directly estimating the Q-function $Q(s,a)$, from which the policy is derived by selecting the control action for which $Q(s,a)$ is maximal. Popular on-line algorithms in this category are *Q-learning* and *SARSA*. After every state transition, $Q(s_k, a_k)$ is updated as follows

$$Q(s_k, a_k) \leftarrow Q(s_k, a_k) + \alpha\delta_{\mathrm{TD},k} \tag{1}$$

where α is the learning rate, γ the discount factor and δ_{TD} is the temporal difference (TD) error, which is calculated differently for SARSA and for Q-learning:

$$\begin{aligned} \delta_{\mathrm{TD_{SARSA}},k} &= r_{k+1} + \gamma Q(s_{k+1}, a_{k+1}) - Q(s_k, a_k) \\ \delta_{\mathrm{TD_Q},k} &= r_{k+1} + \gamma \max_{a'} Q(s_{k+1}, a') - Q(s_k, a_k) \end{aligned} \tag{2}$$

While SARSA uses the Q-value of the actually selected action in its update rule, Q-learning uses the maximum achievable Q-value, which does not depend on the executed policy. This makes SARSA an *on-policy* and Q-learning an *off-policy* algorithm.

To speed up convergence, SARSA and Q-learning can be combined with *eligibility traces*, thereby forming SARSA(λ) and Q(λ), respectively. With eligibility traces, learning updates are not just applied to the previously visited state-action pair (s_k, a_k), but also to pairs that were visited earlier in the episode. In this process, more recently visited (s,a)-pairs receive a stronger update than pairs visited longer ago. The update rules become

$$\begin{aligned} Q_{k+1}(s,a) &= Q_k(s,a) + \alpha\delta_{TD} e_k(s,a) \\ e_k(s,a) &= \begin{cases} 1 & \text{, if } s = s_k,\, a = a_k \\ \gamma\lambda e_{k-1}(s,a) & \text{, otherwise.} \end{cases} \end{aligned} \tag{3}$$

which are now executed for all $s \in S, a \in A$. At the start of a new episode, e is reset to 0. For Q(λ), the eligibility of preceding states is only valid while the greedy policy is being followed. Thus, for Q(λ), e must also be reset after an exploratory action.

For Q-learning and SARSA, the greedy policy selects actions $a_{k,\mathrm{greedy}}$ according to

$$a_{k,\mathrm{greedy}} = \arg\max_{a'} Q(s_k, a'). \tag{4}$$

A widely used action selection scheme that includes exploratory actions is the ϵ-greedy policy, which defines an exploration rate ϵ at which random actions are chosen (and greedy actions otherwise).

3 Large disturbances during Reinforcement Learning

For a real-time dynamic system such as a humanoid robot, the main sources of outliers are sudden changes in the dynamics, sensor and actuator noise and sampling time irregularities. We now discuss the effect of each of these categories of outliers on the Reinforcement Learning process, as well as disturbance rejection and detection.

1. Outliers due to sudden changes in the system dynamics or the environment An external disturbance such as an instantaneous push or a step-up or step-down in the floor can cause an outlier in the state transitions and bring the learning agent into an exotic state s_k^{exo} from which it needs to recover. This may not always be possible. See Figure 2a for a one-dimensional example. The agent will erroneously relate $Q(s_{k-1}, a_{k-1})$ to $Q(s_k^{exo}, a_k)$. Especially when the agent cannot recover from s_k^{exo}, states on good solution paths might be linked to the negative results of the disturbance. The agent cannot relate and thus not reward its behavior before the disturbance to the reward that it actually deserves.

When the agent almost never suffers from such disturbances, it probably does not have enough experience to recover from exotic states. However, when it is disturbed frequently, it is able to practice in these regions of the state space as well, so that it automatically learns to correct these outliers.

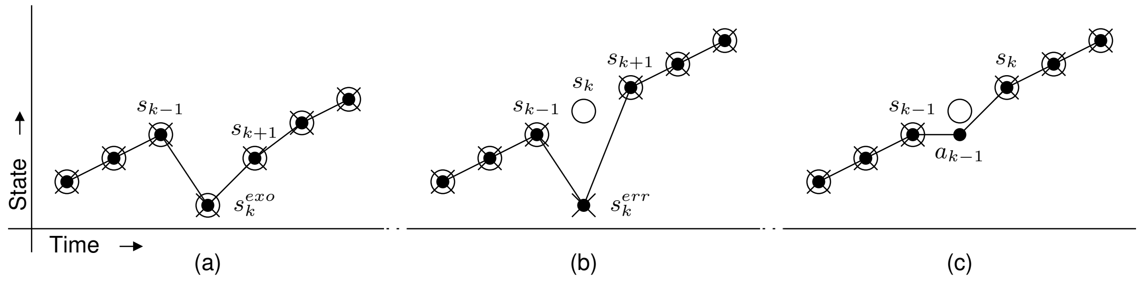


Figure 2: Effect of disturbances. ◯ is the actual state, $\times$ is the measured state, and $\bullet$ is the state on which the effective action at that timepoint based. (a) Instantaneous push. (b) Erroneous sensor reading. (c) Sample time irregularity.

The effect of actuator noise is similar. However, actuator noise is likely to bring the system in a state that it also could have visited under normal conditions, e.g. due to exploration. Therefore, this type of outlier is less interesting and we do not further look into it.

In summary, the learning problem becomes larger because a larger part of the state space is visited. Furthermore, with every outlier, parts of the state space are connected that are in principle not related, which can negatively influence learned behavior in frequently visited parts of the state space.

2. Outliers due to sensor noise In case of an outlier in the sensor reading, the system will perceive a state s_k^{err} that is different from the actual s_k. Because of this, it will experience two erroneously perceived state transitions; $(s_{k-1}, a_{k-1}) \to (s_k^{err}, r_k^{err})$ and $(s_k^{err}, a_k) \to (s_{k+1}, r_{k+1}^{err})$, with $r_k = R(s_{k-1}, a_{k-1}, s_k)$. See Figure 2b for a one-dimensional example. The effect on the learning agent is that $Q(s_{k-1}, a_{k-1})$ will be related to $Q(s_k^{err}, a_k)$ – the latter most probably having a Q-value unrelated to the problem with s_k^{err} being an outlier. Next, $Q(s_k^{err}, a_k)$ is related to $Q(s_{k+1}, a_{k+1})$. When using eligibility traces however, it is likely that the subsequent TD updates (see (2)) largely cancel each other out. When using SARSA, for example, the two subsequent errors in the TD updates are:

$$\begin{aligned} \Delta\delta_{\text{TD,k-1}} &= \delta_{\text{TD,k-1}}^{err} - \delta_{\text{TD,k-1}}^{correct} &&= r_k^{err} - r_k + \gamma(Q(s_k^{err}, a_k) - Q(s_k, a_k)) \\ \Delta\delta_{\text{TD,k}} &= \delta_{\text{TD,k}}^{err} - \delta_{\text{TD,k}}^{correct} &&= r_{k+1}^{err} - r_{k+1} + Q(s_k, a_k) - Q(s_k^{err}, a_k) \end{aligned} \tag{5}$$

When using eligibility traces and when s_k^{err} is not terminal, *both* incorrect TD updates are applied to all $Q(s, a)$ with $e(s, a) \neq 0$. The net error function $\Delta(s, a)$ then becomes (also see (3)):

$$\begin{aligned} \Delta(s,a) &= \alpha(\Delta\delta_{\text{TD,k-1}} e_{k-1}(s,a) + \Delta\delta_{\text{TD,k}} e_k(s,a)) = \alpha(\Delta\delta_{\text{TD,k-1}} + \Delta\delta_{\text{TD,k}}\gamma\lambda) e_{k-1}(s,a) \\ &= \alpha(r_k^{err} - r_k + \gamma\lambda(r_{k+1}^{err} - r_{k+1}) + \gamma(1-\lambda)(Q(s_k^{err}, a_k) - Q(s_k, a_k))) e_{k-1}(s,a) \end{aligned} \tag{6}$$

with $e_k(s,a) = \gamma\lambda e_{k-1}(s,a)$. $\Delta(s,a)$ is maximal for (s_{k-1}, a_{k-1}) because $e_{k-1}(s_{k-1}, a_{k-1}) = 1$. If the reward function R is simple (e.g., sparse with possibly a time penalty) and $s_{k,err}$ is not a goal state, it often holds that $r_k^{err} = r_k$ and $r_{k+1}^{err} = r_{k+1}$, in which case the remaining error goes to 0 when λ approaches 1.

However, the agent will base a_k on $Q(s_{k,err}, *)$, which results in a suboptimal action. This action can remove the agent from its optimal path - but not worse than when an exploratory action was chosen. Unlike with a dynamics disturbance, the agent has a fair chance to successfully continue its trial and receive the reward it deserves based on its behavior before the disturbance.

In summary, the most prominent effect of a sensor outlier are two learning updates with an erroneous TD-error (with eligibility traces, the effect is usually very small) and one extra random action.

3. Outliers due to sampling time irregularities When the sampling time is suddenly disrupted, e.g. due to calculations taking longer than normal, the dynamics of the system evolve longer (or shorter) than normal. Depending on the type of action that the agent executes, the action itself will be shortened or extended, which results in a different resulting state. This is true for actions like motor torque or voltage when they are maintained until other actions overwrite them. However, unless the disruption lasts a large multiple of the sampling time, the effect is expected to be limited. See Figure 2c for a 1-dimensional example.

In summary, an outlier in sampling time is expected to have a limited effect, unless it lasts a large multiple of the sampling time.

Disturbance rejection and detection For on-line TD-learning, simply skipping the learning update for state transitions that include outliers is enough to exclude the outlier from the learning process. When eligibility traces are used, the traces can simply be cleared once an outlier is detected. This skips the faulty learning update. Note that clearing the eligibility trace can slow down the learning process.

In this paper, we do not focus on the subject of detection of outliers. In our simulations, we simply signal the learning algorithm that an outlier was detected at the moment we apply the disturbance. For on-line disturbance detection on robotics systems, a state transition model of the robot and its environment is needed, preferably learned on-line. Model learning techniques that might be appropriate for such systems are Locally Weighted Learning (LWL) [2], a local linear regression technique, and SmartSifter [12]. Once a model is available, every state transition can be compared to the expected state transition based on the model. If the prediction differs significantly from the measurement, the measurement can be labeled as an outlier. Prediction intervals can serve as a significance measure and are easily calculated for LWL.

4 Experimental set-up

In order to evaluate the effect of several types of outliers on the learning process and the efficacy of skipping the learning update, we have performed simulations of a simple two-dimensional system - the simplest walker - that learns to walk. To simulate a disturbance, the walking system was severely perturbed for a single time step. Because we do not focus on the outlier detection aspect, we simply signal the learning algorithm of the presence of the outlier when the disturbance was applied. To reject the outlier, the learning update is not performed and the eligibility trace is cleared. We tested three types of disturbances:

1. An instantaneous push, which is a change in the dynamics with a duration of one time step.
2. An erroneous sensor reading (spike noise).
3. A sampling time irregularity, resulting in a sample that takes longer to acquire.

The relevant parameters are a learning rate α of 0.4, exploration rate $\epsilon = 0.05$ (discounted such that it is 0.01 after 30 simulated hours), time discounting factor $\gamma = 0.99$ and trace decay rate $\lambda = 0.92$. The time step was $0.2s$. We keep the learning rate constant, which is realistic for a learning robot; it can then continuously adapt to possible slow changes in the environment or its own dynamics, such as changes in friction due to wear and tear of the system.

The simplest walker The simulated system is a compass walker [5] consisting of two rigid legs of unit length connected by a frictionless hinge at the hip (Figure 1(b)). A mass of unit size is located in the hip of the walker. The legs are massless, while the feet contain an infinitesimally small mass. This results in pendulum-like behavior of the swing leg. We allow the swing foot to be briefly below floor level during its swing, which is inevitable for a walker without knees; the second time the swing foot is at floor level height, the walker makes a step and the swing leg becomes the new stance leg. The system is described by the following equations of motion:

$$\begin{bmatrix} \ddot{\theta} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} \sin(\theta - \sigma) \\ \sin(\phi)(\dot{\theta}^2 - \cos(\theta - \sigma)) + \sin(\theta - \sigma) \end{bmatrix} \tag{7}$$

in which θ is the angle between the stance leg and the floor normal and ϕ is the relative hip angle. We used 4th order Runge-Kutta to integrate (7) with a time step of 0.0125s. At heel strike, the collision with the floor causes the system to lose energy, and the swing leg becomes the new stance leg and vice versa. The impact is modeled as an instantaneous velocity change from the pre-collision state(–) to the post-collision state (+) by:

$$\begin{bmatrix} \dot{\theta}^+ \\ \dot{\phi}^+ \end{bmatrix} = \begin{bmatrix} \cos(2\theta) \\ \cos(2\theta)(1 - \cos(2\theta)) \end{bmatrix} \dot{\theta}^- \tag{8}$$

This unactuated system is able to passively and stably walk down a slope in a gravity force field with unit magnitude. The slope angle σ is 0.004rad. For this passive walking gait, a footstep takes around 4 seconds (20 time steps). Its stability and walking speed can be increased by adding actuation. Because the legs are virtually massless, the action consists of an acceleration of the swing leg instead of a torque. This acceleration has no effect on the movement of the stance leg, only on the swing leg. The agent can choose its action from the range $[-1.2, 1.2]\text{ms}^{-2}$ in 15 uniformly spaced steps. The state space of the learning agent is spanned by θ, $\dot{\theta}$, ϕ and $\dot{\phi}$. At the start of a trial, the walker is randomly set to an initial condition that is known to contain enough energy to start walking (but not necessarily leads to a stable walking pattern). A trial ends when the walker fell down or after 100 seconds. The rewards are -1 for every action, 50 per meter at every footstep and -50 when it falls. By rewarding traveled meters and punishing time, the walker will optimize towards maximum walking speed. In most gaits we observed, a footstep took around 1.6 seconds.

Test scenarios We analyzed three types of large disturbances. In each test, we added a specific type of disturbance to the system at random time intervals with an average of one disturbance every 50 time steps.

The first type of large disturbance is a physical perturbation of the system in the form of an instantaneous push, which leads to an outlier in the state transitions (Figure 2(a)). In our simulation this was effected by applying an instantaneous change in angular velocity of the stance leg, distributed uniformly over the ranges $[-0.044, -0.038]$rad/s and $[0.038, 0.044]$rad/s. This corresponds to a change in its velocity of rougly 25%-100% depending on the moment of application.

The second type of large disturbance is sensor spike noise. Potentiometers commonly applied to measure joint angles can lose contact, resulting in noise spikes. Such a disturbance only changes the state reported to the learning module, while the physical system itself remains unchanged (Figure 2b). In our simulation, an outlier was implemented by replacing the sensor reading of the hip angle by a random value distributed uniformly over the range $[\frac{-\pi}{2}, \frac{\pi}{2}]$rad. In addition, we changed the hip angular velocity as if it resulted from differentiating the faulty position signal.

The third type of large disturbance is sampling time irregularity. While the motor control policy of most robots runs on a proper real-time operating system, samples may still be delayed or lost due to communication errors or algorithmic calculations taking occasionally longer than normal. This was simulated by omitting samples at random, causing the last chosen action to take twice as long.

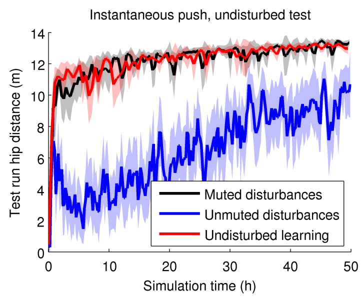
5 Results and Discussion

We present our test results as learning performance graphs, showing the average traveled distance of the walker versus simuation time. After every 20 episodes, a series of 11 test runs – each 100 seconds long – was performed. By measuring traveled distance, an increase in performance in terms of faster walking shows an increase in traveled distance. On the other hand, equally fast but unstable walking results in the walker falling easily, resulting in on average a decrease in traveled distance. Each point in the graph is an average of 20 tests and includes the 95% confidence interval of the average. Usually, the walker quite quickly learns to walk, after which it (slowly) continues to optimize for walking speed.

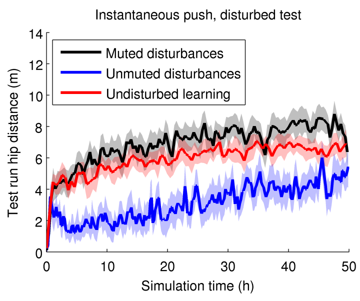
The performance of learning to walk using the SARSA algorithm without disturbances is shown as baseline in, e.g., Figure 3(f). We see that the walker learned to walk after about 1.5 hours, after which it slowly keeps increasing its average walking speed, which increases its traveled distance during the 100 second test runs. We now compare this result to our different disturbance scenarios.

Push We first compared a learning process disturbed with random pushes – but undisturbed during the test runs – to undisturbed learning. This allows us to ascertain how well we do on learning the underlying undisturbed problem. In Figure 3(a) we can see that by muting the learning algorithm during an outlier as described in Section 3, we do indeed learn the optimal policy. Without rejection however, the performance is significantly worse. Apparently, Q-values of states that regularly visited during walking are severely affected. Figure 3(b) shows the more realistic scenario we get when we include disturbances in the test runs. Again, undisturbed learning and outlier rejection perform similarly (but worse than testing without disturbances, of course), while regular learning in the disturbed system has a lower performance. This indicates that the SARSA algorithm is unable to treat the outliers in the dynamics as stochastic variations. Learning without disturbances while testing with disturbances performs slightly worse than learning with and rejecting outliers. This can be explained by the fact that the disturbed learner visits a larger part of the state space (i.e., more exotic states), from which it learns to recover, than the undisturbed learner. Note that it might be expected that *not* rejecting outliers could eventually improve performance over rejection, because the system could possibly learn a more cautious walking gait. In our simulations this was not the case.

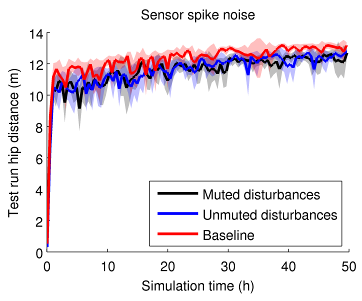
Although we motivated our choice of keeping the learning rate constant (see Section 4), we tested the effect of slowly reducing the learning rate over time (results not shown here). The low learning rate at the end of the runs allowed regular SARSA to perform enough averaging to endure the disturbances and reach the same level of performance as outlier rejection. However, it converged more slowly. We also compared the performance of Q-learning and SARSA, but could not find a significant difference between the two, indicating that outlier rejection is equally applicable to both algorithms.



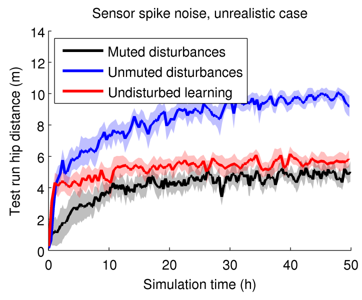
(a) Undisturbed test (pushes). Disturbance rejection (muting) restores optimal performance over regular disturbed learning.



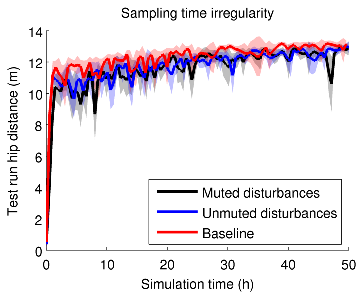
(b) Disturbed test (pushes). Disturbance rejection performs better than undisturbed learning and regular disturbed learning.



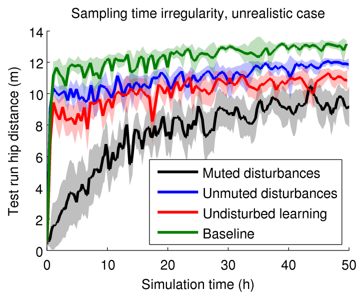
(c) Disturbed test (sensor noise). There is almost no difference between this scenario and the baseline.



(d) Disturbed test (sensor noise), unrealistic case. Rejecting outliers has a *negative* effect.



(e) Disturbed test (timing). There is almost no difference between this scenario and the baseline.



(f) Disturbed test (timing), unrealistic case. Outlier rejection has the worst performance.

Figure 3: Performance comparison for the push scenario (a, b), sensor spike scenario (c, d) and sampling time irregularity scenario (e, f) using SARSA.

Sensor spike noise Our second testing scenario is sensor spike noise. In Figure 3(c) we see that one spike every 50 timesteps does not significantly reduce the performance from the baseline. As described in Section 3, the eligibility trace masks the disturbance. The situation changes when we increase the frequency of the spikes to once every 5 timesteps: overall performance drops, and only regular unmuted learning learns to deal with the disturbances (Figure 3(d)). Both outlier rejection and undisturbed learning plateau at a significantly lower performance level. Additionally, the convergence of muted learning is slowed by the clearing of the eligibility trace. This shows that for certain disturbances, outlier rejection can actually have a negative effect.

Sampling time irregularity The final scenario involved omitting samples. As can be seen in Figure 3(e), there is again no significant difference from the baseline. More interestingly, even in the unrealistic case of one lost sample every 5 timesteps, the regular unmuted learning process achieves only a slightly lower performance than the baseline (Figure 3(f)). This indicates that our walking system is quite robust against sampling time irregularity if it is allowed to learn to anticipate them. The frequency of the disturbance now allows it to be treated as stochastic noise.

6 Conclusions

Stochastic system behavior is part of the stochastic MDP framework and poses no problem for most learning algorithms, other than that it usually results in the need to average over more experience and thus longer learning times. However, the effect of large and infrequent disturbances – or outliers – is relatively unknown. Every real system will suffer from outliers to some degree. They can occur in sensor readings, timing or in the dynamics of the system or its environment. In this work, we evaluated the effects of outliers on a simple simulation model of a walking robot, which learned to walk using SARSA(λ). We tested the effects of three types of outliers: an instantaneous push, a sensor reading outlier, and a sampling time irregularity.

Pushing the walker at random moments, on average once in approx. 6 footsteps, had a dramatic effect on the learning time and system performance. A simple remedy – rejecting the outliers by excluding the faulty state transitions from the learning process – completely restored the performance of the walker. After an equal amount of practicing hours, the 'ignorant' walker performed only roughly half as good as the outlier rejecting walker. The introduction of random spike noise on the sensor reading of the hip angle, on average once every 50 measurements, had an undetectable effect on the learning agent. When spike noise was applied ten times more often (unrealistic), outlier rejection actually resulted in a *decrease* in learning speed. This can be explained by the fact that we excluded outliers in SARSA(λ) by clearing the eligibility traces, thus on average once in 5 samples, which slowed down learning. Doubling the sampling period randomly, on average every 50th sample, also had an undetectable effect on the learning agent. When this was done ten times more often (unrealistic), the effect became noticeable but was still surprisingly small. Again, rejecting outliers by clearing the eligibility traces led to a large drop in learning speed. The rejection process had a much more negative impact on the learning performance than the outliers themselves.

We can conclude that for this simple model, large disturbances in the dynamics have by far the largest influence on the learning process, compared to timing and sensor outliers. In the future, we will test whether these conclusions hold for a more complex model of our walking robot Leo.

Acknowledgements

This work was supported by the Dutch ministry of Economic Affairs under STW project number dwo7462.

References

[1] Charles W. Anderson, Peter Michael Young, Michael R. Buehner, J. N. Knight, K. A. Bush, and Douglas C. Hittle. Robust reinforcement learning control using integral quadratic constraints for recurrent neural networks. *IEEE Transactions on Neural Networks*, 18(4):993–1002, 2007.

[2] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial intelligence review*, 11(1):11–73, 1997.

[3] Dimitri P. Bertsekas. *Dynamic programming: deterministic and stochastic models*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1987.

[4] K. Doya. Robust reinforcement learning. *Advances in Neural Information Processing Systems*, pages 1061–1067, 2001.

[5] Mariano Garcia, Anindya Chatterjee, Andy Ruina, and Michael Coleman. The simplest walking model: Stability, complexity, and scaling. *ASME Journal of Biomechanical Engineering*, 120:281–288, 1998.

[6] Tommi Jaakkola, Michael I. Jordan, and Satinder P. Singh. Convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6:1185–1201, 1994.

[7] R. Matthew Kretchmar, Peter M. Young, Charles W. Anderson, Douglas C. Hittle, Michael L. Anderson, and Christopher C. Delnero. Robust reinforcement learning control with static and dynamic stability. *Intl. Journal of Robust and Nonlinear Control*, 2001:1469–1500, 2001.

[8] Erik Schuitema, Martijn Wisse, Thijs Ramakers, and Pieter Jonker. The design of LEO: a 2D bipedal walking robot for online autonomous reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[9] Satinder P. Singh, Andrew G. Barto, Roderic Grupen, and Christopher Connolly. Robust reinforcement learning in motion planning. In *Advances in Neural Information Processing Systems 6*, pages 655–662. Morgan Kaufmann, 1994.

[10] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[11] John N. Tsitsiklis and Richard Sutton. Asynchronous stochastic approximation and q-learning. In *Machine Learning*, pages 185–202, 1994.

[12] Kenji Yamanishi, Jun-Ichi Takeuchi, Graham Williams, and Peter Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. In *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 320–324, New York, NY, USA, 2000. ACM.