Accelerated VDBE: Q-Learning with Differentiable Value-Difference Based Exploration Rate Updates

Daniel Huber

December 2025

Abstract

In this paper, I propose Accelerated VDBE, a novel modification to the VDBE algorithm [1] for Q-learning. An agent running VDBE uses the absolute value of temporal difference, which is the difference between currently observed and previously estimated Q-values, to update its exploration rate, ε, at every time step. Accelerated VDBE injects the derivative of a normalized temporal difference function with respect to temporal difference into the update equation for ε. Testing on Farama Gymnasium's cartpole environment [3] shows that Accelerated VDBE outperforms original VDBE and normal Q-learning, by learning to keep a pole upright at least 13% faster.

1 Problem Formulation

The last decade's explosion in robotics research has led to the proliferation of systems that need to behave autonomously, while constrained by limited onboard computers. In many cases, these systems cannot afford to train and perform inference using a GPU-dependent deep neural network. Additionally, robots can learn simple tasks, such as balancing a pole on a moving cart, with a simpler method than a deep neural net. This helps conserve compute power.

A common learning paradigm for simpler tasks is called Q-learning. Q-learning is a tabular reinforcement learning algorithm, meaning it works in discrete environments, those with a finite number of discrete states and actions. A Q-value is the agent's estimate of future reward given that it takes a particular action in a given state. At any time step, a Q-learning agent can either choose the action with the highest estimated future reward, or choose a random action to explore. The agent updates its previous estimated Q-values as it tries particular actions repeatedly. Q-values are updated as follows for a state $S_t \in \mathcal{S}$, action $A_t \in \mathcal{A}(s)$, learning rate α and reward R_{t+1}:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \tag{1}$$

By visiting states many times and trying different actions, the agent gradually improves its understanding of the rewards associated with different actions. Eventually, the agent learns the ideal action to take in each state, which is the one that has the highest Q-value. This knowledge forms a policy, which functions as a set of instructions for the agent to maximize reward. Policy selection in ε-greedy Q learning, which is what is used in this paper, is defined as follows, with $\pi(s)$ denoting the action chosen at state s:

$$\pi(s) = \begin{cases} \text{argmax}_{a \in \mathcal{A}(s)} Q(s,a) & \text{if } \zeta > \varepsilon \\ \text{Random action} & \text{else} \end{cases} \tag{2}$$

Where $\zeta \in [0,1]$ is randomly selected anew every step. The name ε-greedy comes from the behavior: The agent has a ε chance of selecting a random action to explore, and (1-ε) chance of selecting the best action known to it.

I have provided merely a brief overview of Q-learning. The reader is directed to section 6.5 of [2] for a full treatment.

A common issue in Q-learning is how to choose $\varepsilon \in [0,1]$, known as the exploration rate. It determines the chance of an agent choosing the best known action (i.e. exploiting existing knowledge) versus choosing a different action to explore. Agents must explore new actions in order to find the best one available, but must balance this with exploiting any already known information from previous trials so as not too spend unnecessary time acting sub-optimally. This is known as the exploration-exploitation dilemma [2]. In many Q-learning implementations, ε is simply a constant between 0 and 1, or decreases gradually over time, or is selected with some other heuristic strategy. However, these are not the most efficient ways to select ε.

In my project, I built off of an algorithm called Value-Difference Based Exploration (VDBE) developed by Tokic and Palm [1], to improve ε updates using a differentiable parameter. Thus, I achieve gains in agent performance both over standard Q-learning and the algorithm proposed in [1]. I call my modified algorithm Accelerated VDBE.

2 Previous Work: Standard VDBE

Instead of a heuristic rule for ε updates, Value-Difference Based Exploration (VDBE) [1] updates ε in accordance with how "surprised" the agent is at a given step. After providing a random starting value for ε between 0 and 1, it is modified after every step by:

$$\varepsilon_{t+1} = \delta \tanh \frac{2\alpha|T|}{\sigma} + (1-\delta)\varepsilon_t \tag{3}$$

Where temporal difference $T = R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)$. Additionally, "$\sigma$ is a positive constant called inverse sensitivity and $\delta \in [0,1)$ a parameter determining the influence of the selected action on the state-dependent exploration probability" [1]. The authors suggest a formula for δ of $\delta = \frac{1}{|\mathcal{A}(s)|}$, meaning the inverse of the size of the action space.

The overall goal of VDBE is for the agent to explore more when it is less familiar with a state, and exploit current knowledge more when it is more familiar with a state. Consider that $|T|$ measures the difference between the observed Q-value with the best possible action taken at step $t + 1$ and the previously observed Q-value for a given action at step t. So $|T|$ will be higher if the agent takes an action and finds a highly different Q-value to its previous estimate (i.e. the agent is surprised), and smaller if the observed difference is smaller. Tanh and σ serve to normalize the temporal difference, since ε must be between 0 and 1. The authors determine σ experimentally (see section 4 of [1] and section 4 of this paper).

So, if the agent is more surprised, the tanh term will be higher, and therefore ε will increase before the next step. Therefore, the agent will be more likely to choose to explore as opposed to taking a known action, so it can observe a state-action pair which it has not yet observed at length. And if it is less surprised, it will explore less since it already knows the available state-action pairs well.

3 Contributions: Accelerated VDBE

While Tokic and Palm's algorithm provides a significant performance boost over standard Q-learning, it makes updates to ε highly volatile. I theorized that this could lead the agent to increase ε to an unnecessary degree if it happens to take an action that was previously unexplored in the case that said action is very similar to the optimal action. My solution is to smooth out updates to ε by introducing differentiability.

I do this by setting δ equal to the absolute value of the derivative of the "surprise term," which I denote $U(T) = \tanh \frac{|T|}{\sigma}$, with T, σ defined as above. Note that Tokic and Palm multiply $|T|$ by the learning rate and again by 2. For simplicity I removed those factors and only use σ as my scaling factor. This is still equivalent to Tokic and Palm's work since multiplying by the learning rate times 2 is merely positive scalar multiplication, and σ is also a positive scalar.

The surprise term measures whether the agent is becoming more or less surprised at a given moment. The derivative of surprise is:

$$\frac{d}{dT}U(T) = \frac{T(1 - \tanh^2 \frac{|T|}{\sigma})}{|T|\sigma} = \frac{T(1 - U^2(T))}{|T|\sigma} = \begin{cases} \frac{1-U^2(T)}{\sigma} & \text{if } T > 0 \\ -\frac{1-U^2(T)}{\sigma} & \text{if } T < 0 \end{cases} \tag{4}$$

However, the sign of T is irrelevant for accelerated VDBE, since I only care how surprised the agent is, not whether its estimate was too low or too high. I also need my derivative to be defined at 0. Furthermore, it doesn't make sense for ε_{t+1} to be updated by a mutlipule of ε_t that is greater than 1. So for all these reasons, instead of directly setting the derivative of the surprise term equal to δ, I set

$$\delta_* = |\frac{d}{dT}U(T)| = \frac{1 - U^2(T)}{\sigma} \tag{5}$$

Where δ_* denotes the new δ in Accelerated VDBE. Plugging this back into the equation for ε updates, the result is

$$\varepsilon_{t+1} = \delta_* U + (1-\delta_*)\varepsilon_t = \frac{1-U^2(T)}{\sigma} U + \left(1 - \frac{1-U^2(T)}{\sigma}\right)\varepsilon_t \qquad (6)$$

$$\varepsilon_{t+1} = \frac{U-U^3}{\sigma} + (\frac{\sigma+U^2-1}{\sigma})\varepsilon_t \qquad (7)$$

Since it is not immediately obvious how equation (7) causes ε to behave, consider figures 1 and 2, which are simulations that compare my ε update rule with the one proposed by Tokic and Palm. In both figures, time steps are the x-axis. Figure 1 uses a randomly generated set of $|T|$ values between 0 and 100, while figure 2 I define $|T|$ as a series that increases, then decreases smoothly along a Gaussian bell curve. In both figures, the upper graphs show the progression of ε for standard VDBE and accelerated VDBE. The lower graphs show the progression of $|T|$, and δ_* for accelerated VDBE, respectively. δ is for standard VDBE, and remains constant.

The first figure shows clearly that for a volatile $|T|$, ε is much smoother with accelerated VDBE than with regular VDBE. In the second figure, it is shown that accelerated VDBE responds slower to sustained changes than the original VDBE. Accelerated VDBE actually deccelerates how quickly ε responds to changes, ironically, although as shown in section 4 this actually improves performance.

In my method, when $|T|$ is small, δ_* is high. This makes the agent react quickly to changes. So when the agent starts seeing any hint of surprise, it will ramp up ε quickly, to take advantage of this fast. However, this does not mean that ε always increases for small $|T|$. If $|T|$ is constant at 0, the $U - U^3$ term becomes 0, so ε decreases quickly. The point is that ε responds quickly to *changes* in $|T|$.

When T is large, δ_*. The agent ignores large spikes. If the agent is already very surprised, being even more surprised won't make ε spike up further. A single spike may be an anomaly and I want to make sure surprise is actually staying high consistently before the agent seriously alters ε. While it may seem counterintuitive, since one might initially assume it makes more sense to increase ε faster when the agent is more surprised, testing data shows that my method strikes a good balance between making ε too sensitive versus not sensitive enough.

4 Testing and Results

I tested the performance of Accelerated VDBE, normal VDBE, and standard Q-learning using Farama Gymnasium's cartpole environment [3]. The agent controls a cart which slides along a one-dimensional track. As per Gymnasium's website, "a pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The pendulum is placed upright on the cart and the goal

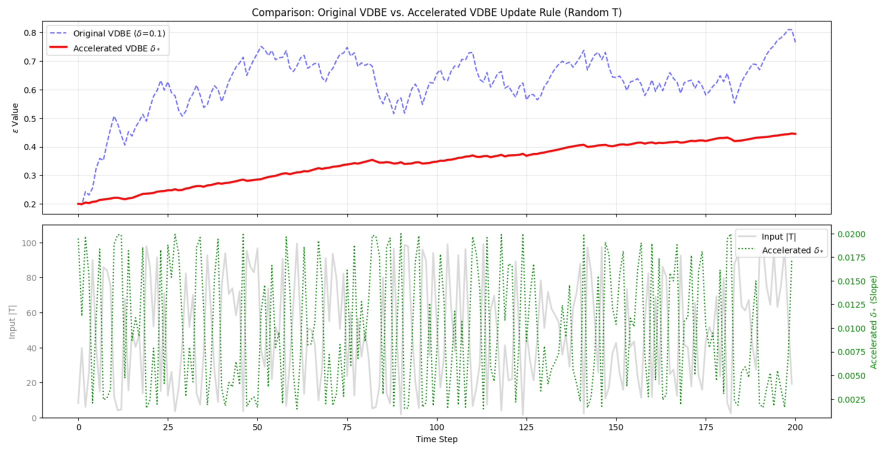


Figure 1: $|T|$ varies randomly

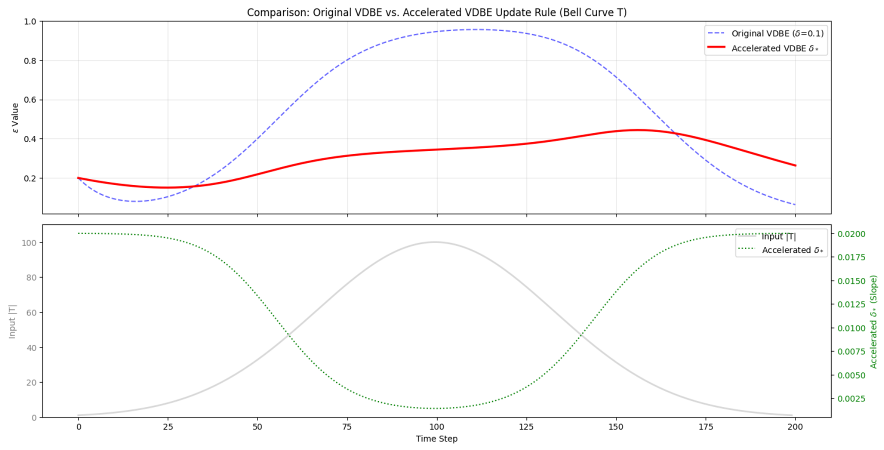


Figure 2: $|T|$ follows a smooth curve

is to balance the pole by applying forces in the left and right direction on the cart" [3]. Figure 3 is a screenshot of the environment.

The agent can choose at any time-step to move the cart either left or right. The episode ends if the pole falls below $\pm 12°$ of vertical. A reward of "+1 is given for every step taken, including the termination step" [3]. The pole may start each episode at any random angle within about $\pm 1.4°$ of vertical. I defined a successfully trained agent as one that was able to learn to keep the pole upright for an average of 225 time steps for 100 episodes in a row. This corresponds to the trained agent knowing how to keep the pole upright for 250 time steps in an average episode, or about 5 seconds of simulation time on my computer. I capped each training cycle at 5000 episodes, and considered

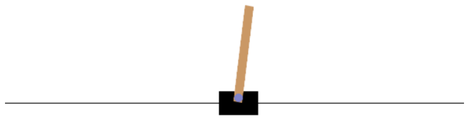
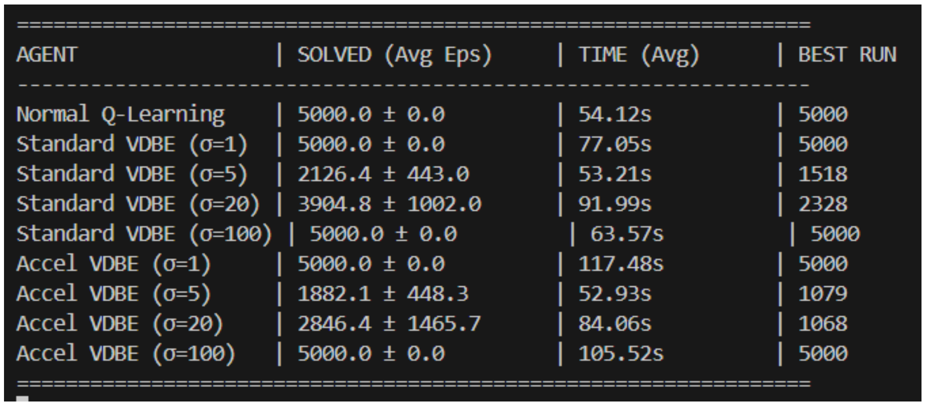


Figure 3: Cartpole Environment



```
=================================================================
AGENT                   | SOLVED (Avg Eps)      | TIME (Avg)       | BEST RUN
-----------------------------------------------------------------
Normal Q-Learning       | 5000.0 ± 0.0          | 54.12s           | 5000
Standard VDBE (σ=1)     | 5000.0 ± 0.0          | 77.05s           | 5000
Standard VDBE (σ=5)     | 2126.4 ± 443.0        | 53.21s           | 1518
Standard VDBE (σ=20)    | 3904.8 ± 1002.0       | 91.99s           | 2328
Standard VDBE (σ=100)   | 5000.0 ± 0.0          | 63.57s           | 5000
Accel VDBE (σ=1)        | 5000.0 ± 0.0          | 117.48s          | 5000
Accel VDBE (σ=5)        | 1882.1 ± 448.3        | 52.93s           | 1079
Accel VDBE (σ=20)       | 2846.4 ± 1465.7       | 84.06s           | 1068
Accel VDBE (σ=100)      | 5000.0 ± 0.0          | 105.52s          | 5000
=================================================================
```

Figure 4: Testing results in tabular form. For each agent, the average number of episodes required to train completely, ± the standard deviation, the time to train completely, and the best training cycle are listed. Any agent listed as having solved the task in 5000 episodes did not train successfully.

an agent as having failed to train successfully if it did not meet the successful training criteria within 5000 episodes.

I tested each agent 10 times. For Accelerated and normal VDBE, I tested using using four values of σ: 1, 5, 20, and 100. For normal VDBE, Tokic and Palm [1] suggest setting $\delta = \frac{1}{|\mathcal{A}(s)|}$, which would be $\frac{1}{2}$ in my case. However, I found $\frac{1}{10}$ to have the best performance in my tests, so I used this value instead. Basic Q-learning uses a learning rate of 0.2, which was the best learning rate I found when testing. It should also be noted that my environment and basic Q-learning agent scripts are based on the ones provided by the Farama Gym website [3], which I modified to suit my needs and to implement both versions of VDBE. The results of my tests are shown in figures 4, 5, and 6.

As shown in figure 4, the best performing agent was the Accelerated VDBE agent with $\sigma = 5$, which learned to balance the pole in an average of 1882.1 episodes. That is a 13.0% improvement over the standard VDBE method with

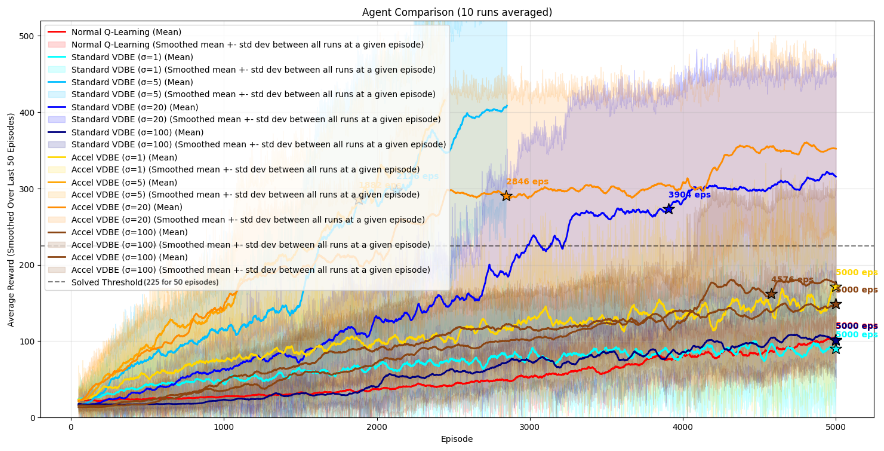


Figure 5: Results of all tests. The average reward over 10 training cycles is marked as a solid line for each of the agents. The average number of episodes it took each agent to train completely is marked by a star. The semi-translucent colorings above and below each line mark the standard deviation of cumulative reward at a given time step over the 10 runs of an agent. Note again that all agents showing a star at 5000 episodes did not train successfully.

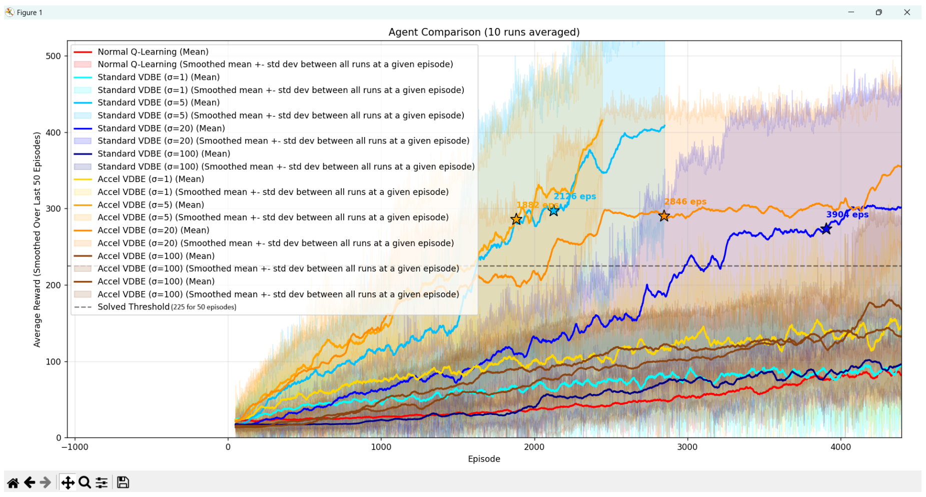


Figure 6: The same as figure 5, but shifted rightwards so the best agents are easier to see.

$\sigma = 5$, which was the second best performing method at 2126.4 average training episodes. Accelerated VDBE with $\sigma = 20$ performed 37.2% better than standard VDBE with $\sigma = 20$. In both cases, the best run was much quicker with Accelerated VDBE. Real life use cases of Accelerated VDBE may not always have time or compute resources to test many different σ values, so this robustness over different σ values is useful. Note that any agent listed in figure 4 as having solved the task in 5000 episodes did not train successfully. So, the only agents which trained successfully are both versions of VDBE with $\sigma = 5$ and $\sigma = 20$. The information in figure 4 is presented in graphical form in figures 5 and 6.

As far as I am aware, Accelerated VDBE marks an improvement over the best existing single-step methods I could find for tabular Q-learning.

5 Other Attempts to Improve Q-Learning

It should be noted that my original project was actually an attempt to improve Q-learning by implementing a differentiable learning rate, that was somehow proportional to the rate of change of reward. My many attempts at this led to the conclusion that it doesn't work well for tabular reinforcement learning methods method like Q-learning. One reason for this is that if the learning rate changes significantly over the course of training, it will skew the Q-values a lot, and render the agent's previous exploration useless since older Q-values were updated at different scales. Another reason is the difficulty in taking the derivative of reward with respect to actions or states in an environment where both are discrete. I did implement a version of Q-learning that set the learning rate proportional to the derivative of reward with respect to time, but because of the first reason this performed poorer than standard Q-learning. It is due to these considerations that I shifted my project to focus on exploration rate.

6 Conclusions and Future Work

I have presented Accelerated VDBE, a novel modification of the VDBE algorithm that updates the exploration rate of a Q-learning agent in proportion to the derivative with respect to temporal difference of the agent's level of "surprise". Accelerated VDBE performs significantly better that regular VDBE and normal Q-learning, since it allows for dynamic exploration rate updates that respond quickly to new observations, but are not overly sensitive to anomalies.

In the future, I hope to turn this project into a publication. To that end, I propose two further avenues of research to explore. One is to implement an exploration-action selection mechanism that is more intelligent than a purely random choice. The second is to determine whether it might be possible to combine Accelerated VDBE with n-step methods. These methods do not update Q-values every step, but use more sophisticated algorithms to update them less frequently, thus improving learning efficiency.

7 References

1. Tokic, Michel, and Günther Palm. Value-Difference Based Exploration: Adaptive Control between Epsilon-Greedy and Softmax. KI'11: Proceedings of the 34th Annual German conference on Advances in artificial intelligence, 4 Oct. 2011. https://www.tokic.com/www/tokicm/publikationen/papers/KI2011.pdf

2. Sutton, R. S., & Barto, A. G. (2021). Reinforcement Learning an Introduction. MTM.

3. Farama Foundation. (2025). Gymnasium documentation. Cart Pole - Gymnasium Documentation. https://gymnasium.farama.org/environments/classic_control/cart_pole