Using NoisyNet to Improve Exploration in Contextual Bandit Settings

Author: Sonny Ruff

Supervisor: Pascal van der Vaart Responsible Professor: Neil Yorke-Smith

Legend

- **RL:** Reinforcement Learning is an unsupervised learning system where agents learn to make decisions by interacting with an environment to maximize a numerical reward signal [1].
- **DRL:** Deep Reinforcement Learning uses deep neural networks as function approximators for RL methods [2].
- **DQN:** Deep Q-Network uses deep neural networks as approximators for the action-value functions [3].

Problem description

While DRL can achieve superhuman performance in many tasks, it often struggles in environments with sparse rewards. In these environments, agents have difficulty learning useful behaviors and eventually converge to a suboptimal solution [4]. Standard methods such as ε-greedy typically lack the structure necessary to guide consistent, long-term behavioral patterns [2].

NoisyNet - Method

The method introduced by Fortunato et al. (2018) called NoisyNet uses learned perturbations of the network weights to drive exploration. This has previously shown promise by improving efficiency on average by 50% on Atari games. In this study, it was tested on two environments:

1. **ContextualBandit-v2:** a bandit with multiple predefined functions mapping the 1-dimensional continuous input to the reward
2. **MNISTBandit-v0:** a bandit rewarding correct identification of MNIST dataset images
3. **NNBandit-v0:** a bandit with the reward being determined by a neural network.

On top of stationary versions of ContextualBandit-v2 and NNBandit-v0, two non-stationary versions were tested:

1. Reward changes every **1000** steps; midpoint of training, mostly stable solution
2. Reward changes every **200** steps; agent is still exploring, partially converged policy

Research Questions

Does a learned, context-dependent exploration method presented by NoisyNet improve the efficiency of DQNs in contextual bandit settings?

1. To which hyperparameters is a NoisyNet-DQN sensitive, and are optimal values generally task-dependent?
2. On what classes of tasks does a NoisyNet-DQN significantly outperform or underperform a regular DQN?
3. What qualitative insights can be gained from visual evaluation metrics of the agent's learned behavior?

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge University Press, 1998.

[2] M. Fortunato, K. Azizzadenesheli, Y. Tang, *et al.*, "Noisy networks for exploration," in *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

[3] "Human-level control through deep reinforcement learning," *Nature*, vol. 518, Feb. 2015.

[4] P. Ladosz, L. Weng, M. Kim, and H. Oh, "Exploration in deep reinforcement learning: A survey," Sept. 2022.

[5] S. Ruff, "testdqn - deep q-network testing repository." `https://github.com/sonnyruff/testdqn`, 2025.

Hyperparameter sensitivity

Hyperparameter	Values
`method`	`random`
`hidden_layer_size`	`[4, 8, 16, 20, 24, 40, 80]`
`noisy_layer_distr_type`	`[uniform, normal]`
`noisy_output_layer`	`[true, false]`
`noisy_reward`	`[true, false]`
`noisy_layer_init_std`	`[1e-4, 1e-3, ..., 1e6]`
`# of samples`	100

Table 1. Stationary environment sweep hyperparameters

Hyperparameter	Value
`hidden_layer_size`	40
`noisy_layer_distr_type`	`normal`
`noisy_output_layer`	`true`
`noisy_layer_init_std`	0.4
`batch_size`	20
`memory_size`	1000

Table 2. General hyperparameters

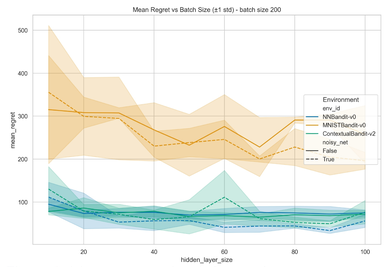

Figure 1. Mean regret ±1 std. for varying hidden_layer_size values (step size 10) for NoisyNet and Non-NoisyNet agents, tested on seeds 0 to 4 for all three stationary environments.

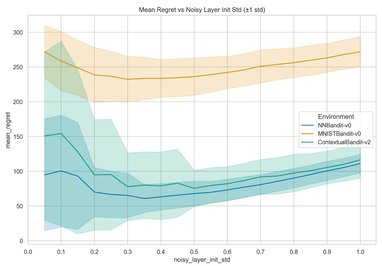

Figure 2. Mean regret ±1 std. for varying noisy_layer_init_std values (step size 0.05) for NoisyNet agents, tested on seeds 0 to 9 for all three stationary environments.

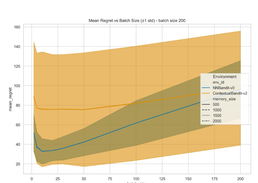

Figure 3. Hyperparameter sensitivity of batch_size in stationary environments

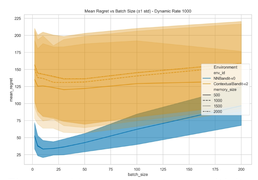

Figure 4. Hyperparameter sensitivity of batch_size in non-stationary environments with a dynamic_rate of 1000

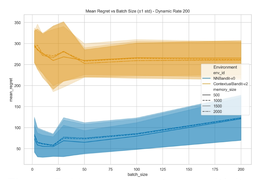

Figure 5. Hyperparameter sensitivity of batch_size in non-stationary environments with a dynamic_rate of 200

Conclusion

RQ1: Effects across all environments:

- `noisy_layer_distr_type`: no effect
- `noisy_output_layer`: no effect
- `noisy_layer_distr_type`: 0.3 to 0.4 optimal
- `batch_size:` in stationary and infrequently changing environments

RQ2: Across all environments, NoisyNet-DQN showed inferior results to regular DQN, both in stationary and non-stationary settings.

RQ3: In most environments, NoisyNet-DQN and regular DQN show similar behavior. In the stationary NNBandit-v0 environment, NoisyNet-DQN showed faster convergence to an optimal policy, although the steeper slope of the NoisyNet-DQN indicates a worse policy. Further visual evaluation on high-dimensional spaces requires more sophisticated methods.

Results

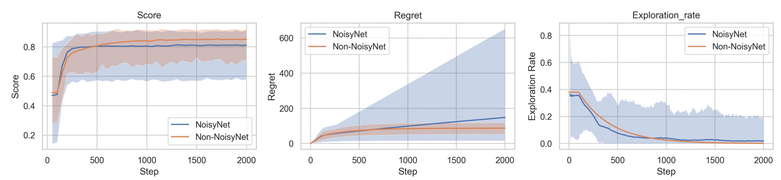

Figure 6. Stationary ContextualBandit-v2 metrics

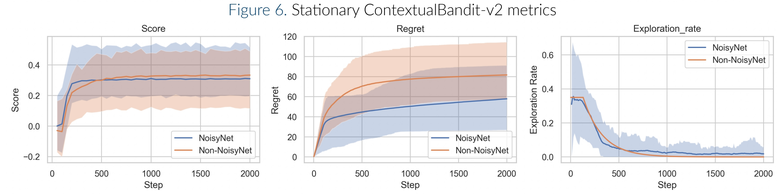

Figure 7. Stationary NNBandit-v0 metrics

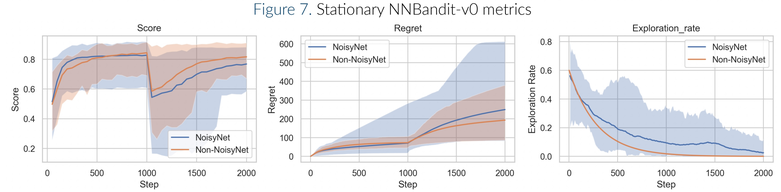

Figure 8. Stationary ContextualBandit-v2 metrics; Dynamic rate set to 1000 steps

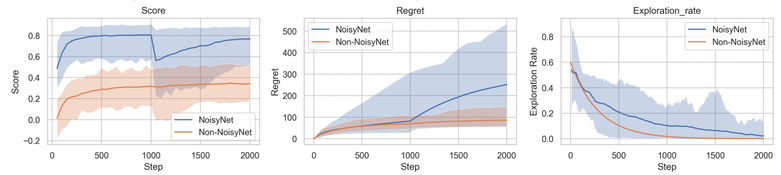

Figure 9. Stationary NNBandit-v0 metrics; Dynamic rate set to 1000 steps

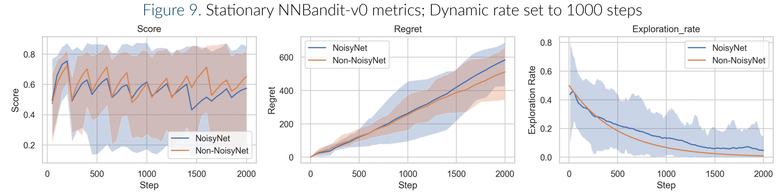

Figure 10. Non-Stationary ContextualBandit-v2 metrics; Dynamic rate set to 200 steps

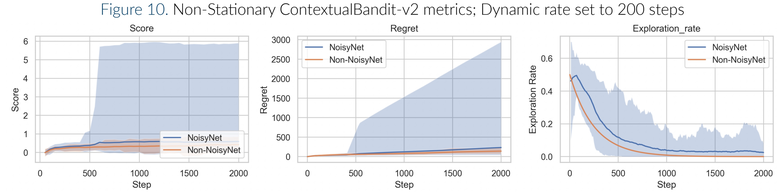

Figure 11. Stationary NNBandit-v0 metrics; Dynamic rate set to 200 steps