Deep Reinforcement Learning with Multiple Sources of Reward Shaping Advice - Final Report

Michael Gimelfarb
MIE 1516

Motivation

In reinforcement learning, reward shaping is a general framework for incorporating advice from an “expert” to help the agent converge faster to the optimal policy. Unfortunately, the traditional approach assumes one source of knowledge. Furthermore, if the advice is bad then the algorithm can converge slower or even fail to converge. We wanted to see whether using Bayesian Model Combination to combine multiple experts in a trust model really accelerates convergence of deep models for reinforcement learning across multiple domains.

The Bayesian Learning Setup

Given a sequence of experts in the form of potential functions $\Phi_1, \Phi_2, \ldots \Phi_n$, we use a Dirichlet prior π_0 to model the probabilities that each expert is correct. Given data on past returns, D, we update this prior to a posterior distribution over the same space. The actual posterior is shown to be intractable, and so variational methods based on relative entropy are used to “project” the posterior distribution onto the family of $n-1$-parameter Dirichlet distributions. Details regarding how to perform this approximation will be published in the near future. Here we only provide a brief overview of the main results.

For an MDP (S, A, P, R, γ), TD(0) updates provide a simple estimate of discounted returns

$$R(t) = r_t + \gamma R(t+1)$$

where r_t is the reward observed when transitioning from state s at time t to some other state s' under action a. We assume greedy epsilon policies are used for training, although the algorithm does not depend on the policies used for training.

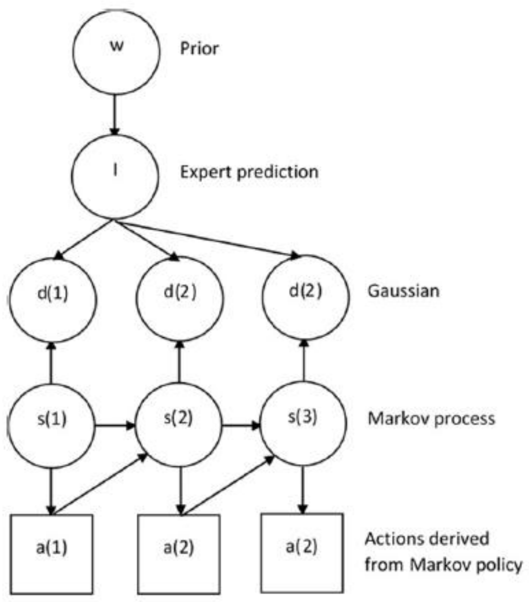
We assume that the actual return is Gaussian distributed, so given the expert Φ_i is correct, the probability of observing return $d = R(t)$ at time t is computed from the Gaussian distribution with mean Φ_i and variance σ^2, which is computed using an online variance formula https://en.wikipedia.org/wiki/Algorithms_for_calculating_variance#Online_algorithm in time O(1). The new posterior is then updated using Bayes’ rule, and the approximation is performed to project this posterior onto the family of Dirichlet distributions. Given the projection returns a Dirichlet Distribution with parameter $\vec{\alpha} > 0$, we use Bayesian Model Combination to combine multiple experts as follows:

$$\Phi(s) = \frac{\sum_i \Phi_i(s)\alpha_i}{\sum_i \alpha_i}.$$

This is the actual reward shaping function used during training, which is “added” to the ordinary rewards as follows:

$$r_t + \gamma\Phi(s_{t+1}) - \Phi(s_t)$$

where s_t is the state at time t and s_{t+1} is the state at time $t+1$. The learning architecture as a decision diagram is shown below:



Domains and Architectures Considered

In this project we consider three domains:

1. the classical inverted pendulum problem https://en.wikipedia.org/wiki/Inverted_pendulum,
2. an 11-by-11 cell Tour-De-Flags problem, and
3. an inventory control problem with perishable inventory.

For the learning architecture, I was able to implement deep feedforward neural networks for all three domains for both Q learning and SARSA.

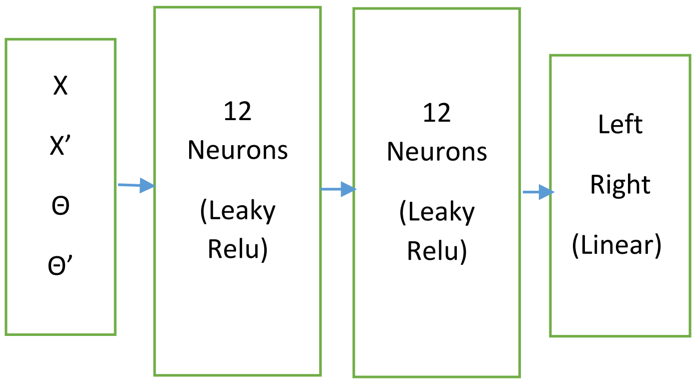
Inverted Pendulum

For this domain, we consider the following experts:

1. The zero potential function
2. A pre-trained Keras neural network with 2 hidden layers and 6 neurons per layer (in an h5 file)
3. A pre-trained Keras neural network with 3 hidden layers and 64 neurons per layer
4. A bad potential function which is the negative of the potential function in 2)
5. A random potential function which returns a random number in [0, 1]

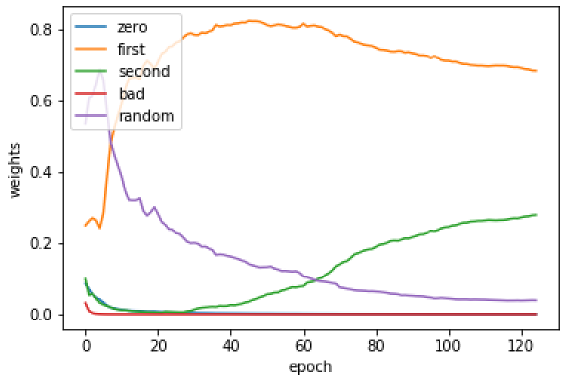
The 3rd neural network has been purposefully over-trained. This is quite realistic in practice, because experts often vary in their quality of advice (e.g. some "experts" may in fact have no knowledge of deep RL so we cannot always trust them). It would be interesting to see if the Bayesian framework can pick up on this.

The learning architecture is a feedforward as follows

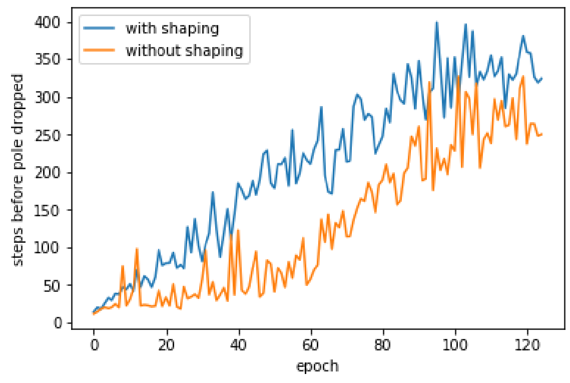


We do not perform any state encoding for this problem.

We train for 10 trials and using SARSA and report the average weights among all trials.



Clearly, the framework is able to identify the first expert (which is number 2 in this document) as the best. Expert 3 is gradually being trusted, which suggests the model is starting to over fit. Overall, we get a good performance here. Let's see what happens when we plot the average steps balanced for the combination and for no shaping (1).



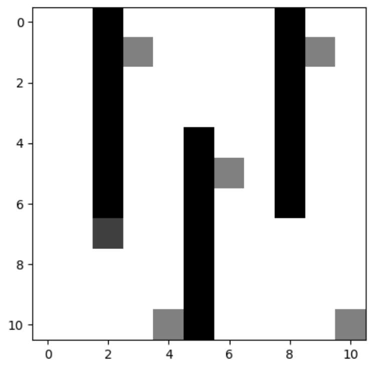
Clearly reward shaping helps for inverted pendulum and the pre-trained network has helped convergence.

In the future, more trials will be performed and all benchmarks will be compared as well.

Tour de Flags

In this problem, the agent starts at a pre-determined position on a grid, and collects a series of flags in order and then goes towards a predetermined destination cell. There may also be obstacles present.

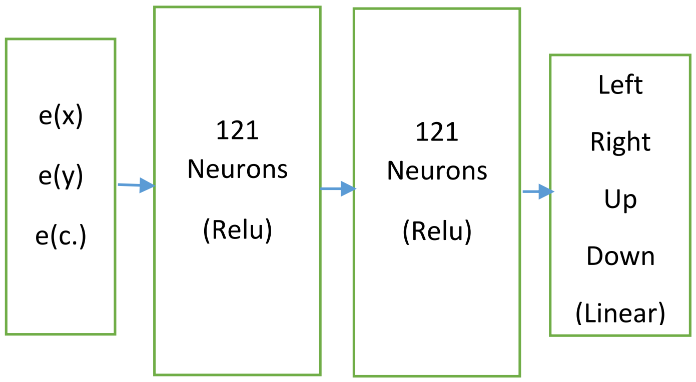
The environment we consider here specifically is an 11-by-11 environment as follows:



The agent (here as dark gray cell) starts in (y=5, x=0) and collects the flags (10, 4), (3, 1), etc (here in light gray), avoiding the obstacles (in black). The final cell (here in gray) is at (10, 10).

The state space for this model is (x, y, collected) where collected is the number of flags already collected. We use one-hot encoding which is (e(x), e(y), e(collected)) where e(x) is the x^{th} basis vector.

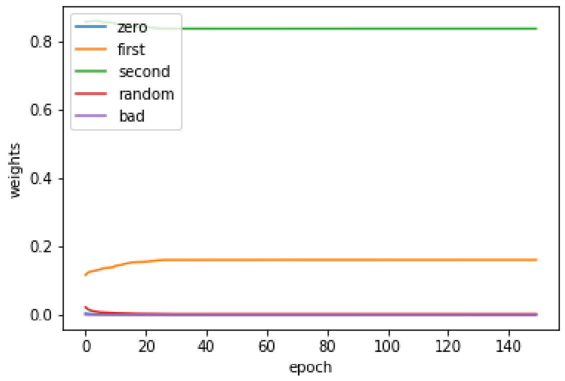
The learning architecture is a feedforward as follows



The experts we use are as follows:

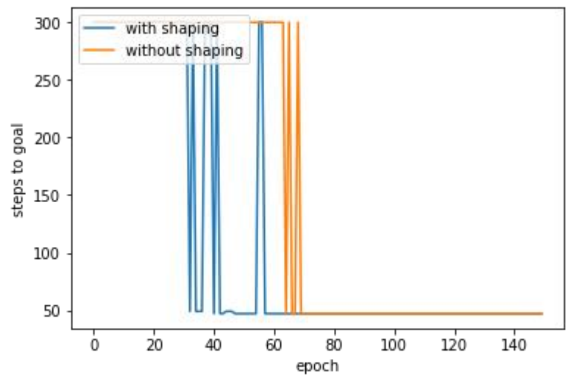
1. No shaping
2. Heuristic which splits the problem into sub-goals, each of which is to collect the next flag, and adds the Manhattan distance to the next goal from the current state
3. The potential function in Ng, Harada and Russell's original 1999 ICML paper "Policy Invariance under reward transformations"
4. The negative of the 3rd expert
5. A random expert which returns uniform numbers in [-1, 0]

Due to the amount of time necessary for training I will only perform a single trial here. Again, we use the deep architecture with SARSA



Interestingly, the model prefers expert 3 to expert 2. In the original paper by Ng et al, expert 2 should be preferred to expert 3. However, their paper does not consider barriers. We believe that ignoring barriers when defining the expert makes it less reliable than using a crudely defined expert which does not consider positional information. This makes sense given that we deduct one point per step.

We also plot the number of steps to the goal for this domain and the current setup

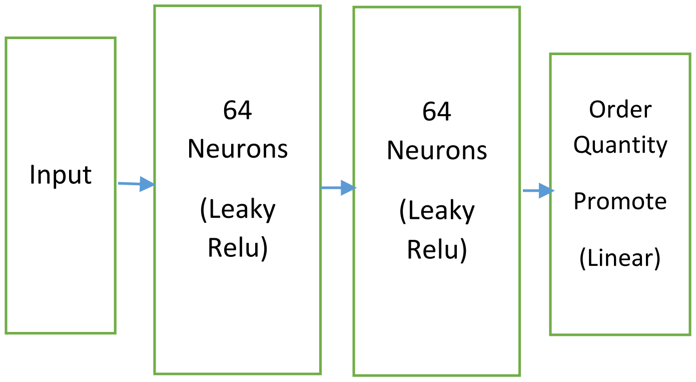


Even though I’ve only done one trial, it is clear that reward shaping has helped converge faster. More precisely, with shaping we can converge after around 30 epochs, whereas without takes over 60. I suspect when multiple trials are considered the result will be more drastic and somewhat clearer.

Inventory Control

I will not explain the details of the domain here, but it is borrowed from Chande et al, “Perishable inventory management and dynamic pricing using RFID technology” found here https://pdfs.semanticscholar.org/809a/b16944f5c96e2531450f516df09186c2f9e2.pdf. I have scaled the perishability window slightly as well as the maximum inventory. This problem is considerably more difficult to solve with value or policy iteration, or with other enumerative approaches, because the state space is quite large (I think around 1.2 billion).

The learning architecture is a feedforward as follows



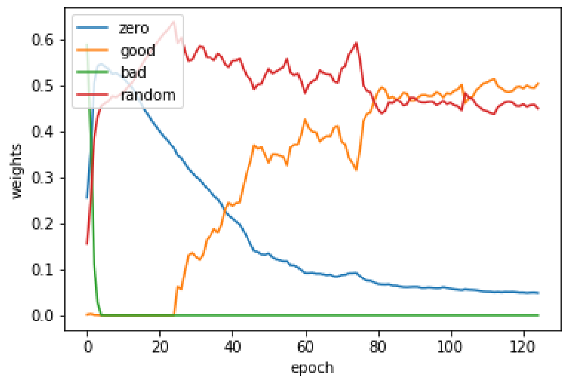
The inputs and outputs (actions) are exactly as described in the paper so I will not mention them here.

Similar to the inverted pendulum domain, we pre-train a neural network and use this as shaping advice. More precisely, here are our experts:

1. No shaping
2. A pre-trained Keras feedforward network with 2 hidden layers and 128 neurons per layer
3. The negative of the expert in 2
4. A random shaping function with uniformly generated values in [0, 10]

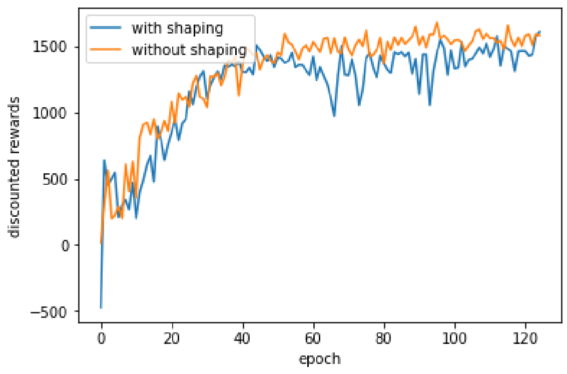
The expert 2 has been trained to optimality but care has been taken to not over-fit the model.

For 10 independent trials and SARSA, we get this for the weights:



This is quite strange. In this case, the learning algorithm is preferring to use a random expert in the early stages to the well-trained model in 2 (good). However, over time the agent learns that the well-trained model is indeed more reliable. It's not quite clear why it prefers random "noise" to no shaping, but perhaps this benefits exploration somehow. This is interesting for further investigation.

When we check the average discounted reward obtained during training, we get this



In this case, it seems that using shaping actually does not benefit at all, and in fact can slightly hurt performance. This could be because the domain is not difficult enough for the deep learning architecture to learn a near-optimal policy even without advice.

Conclusion

Based on the preliminary experiments conducted using the Bayesian reward shaping framework developed in this project, it appears that the framework is quite effective at deciding which expert to trust over time. This translates into considerably better performance on the inverted pendulum domain and faster convergence towards the optimal policy on the grid-world domain. However, more work will need to be done to get clear results for the inventory problem. Perhaps I have not pre-trained the expert well enough or have not chosen a good architecture in this case, or simply the domain is not difficult enough to benefit from reward shaping in this case. In the future I will need to increase the number of trials and compare performance with the other experts individually as baselines (in addition to zero shaping). For goal oriented domains, reward shaping improves the search policies considerably. We will need to investigate whether this is also true for inventory control and similar domains.