CAN DEEP NETS DO PLANNING?

David Krueger

ABSTRACT

We evaluate the ability of MLP neural networks to learn to predict the Q-function of a given environment. We consider the tabular MDP setting, so environments, policies, and value-functions can all be represented as real-valued vectors, and treated as input or output of a deep neural net (DNN). Our initial results indicate that this is not an easy task for MLPs to solve, although they are sometimes able to produce the correct policy significantly more often than a greedy heuristic would.

1 INTRODUCTION

Training DNNs to produce approximate solutions to problems which are usually solved with iterative algorithms could result in heuristic solvers which could be more efficient. The complexity of action-value iteration is $\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|^2$ per iteration, the same order complexity of a *single* feedforward pass of a DNN taking the transition operator as input. The DNN computation has a constant factor which is the number of hidden units, but it can also be parallelized.

In our work, the state and action spaces of the MDPs to be evaluated must be fixed, but this could potentially be extended, although it would be non-trivial to do so. Even in this limited setting, our approach could potentially be useful. For instance it could accelerate Posterior-Sampling Reinforcement Learning (PSRL), which must repeatedly perform planning on environments sampled from a Bayesian posterior.

2 EXPERIMENTS

In each experiment, we generate a dataset with ground-truth Q-functions given by the value iteration algorithm (Sutton and Barto 2017; 4.4). Then we perform supervised learning, using a DNN to predict the Q-function with mean-squared error (MSE) cost function.

2.1 ENVIRONMENTS

We try the following two environments:

1. A grid-world with deterministic transitions (up, down, left, right).
2. An environment with random transitions generated from a Dirichlet distribution with concentration parameters $\alpha_i = \frac{1}{|\mathcal{S}|}$

For both environments, the rewards are generated independently for each state-action from a standard normal distribution. Both environments have a fixed start state (the upper-left in the grid world), and we compare $\gamma = .9$ and $\gamma = .5$.

As a sanity check, we first train 1-layer MLPs on a dataset of 2x2 gridworlds, and find that it recovers the optimal policy 40% of the time (on validation set), beating a greedy heuristic (which does so only 10% of the time).

We then aim to see how performance scales to larger environments, with 9 or 25 states. For these experiments we use square gridworlds, and random MDPs with the same number of states and actions as the gridworlds.

2.2 NEURAL NETWORKS

In both cases, we input the rewards to the DNN, and for the random MDPs, we also input the transition function (for the grid world, it is fixed). The target is the Q-function, and the cost is MSE. We also measure "accuracy", defined as the percentage of cases for which the DNN recovers the optimal policy (as the greedy policy induced by the Q-function).

We use MLPs with 1, 2, or 4 hidden layers of 100 hidden units each and ReLU nonlinearities, trained with Adam with learning rate .002 and batch size 32 for 500 epochs. We train on 10000 examples, and evaluate on 1000 examples.

3 RESULTS

See figures 1 and 2 for our results.

For both types of environment, we found that larger environments with low discounting ($\gamma = 0.9$) were difficult to learn.

Additionally, in random MDPs, our trained networks performance sometimes approached, but never surpassed greedy policies. Furthermore, 1-layer networks gave the best performance in these environments.

In contrast, for gridworld, the greedy policy is a much weaker baseline, and we were able to consistently outperform it, achieving respectable validation policy accuracies of up to 60% (for $\gamma = 0.5$ and $|\mathcal{S}| = 9$). On the smaller environments, where learning was most successful, deep nets gave the best performance.

Another interesting observation is that the gap between training and validation accuracy was generally small, whereas the gap in terms of MSE(Q) was much more significant.

Overall, our results show that MLPs have some potential to learn to perform planning in a single feed-forward pass, but don't seem particularly well suited to the task.

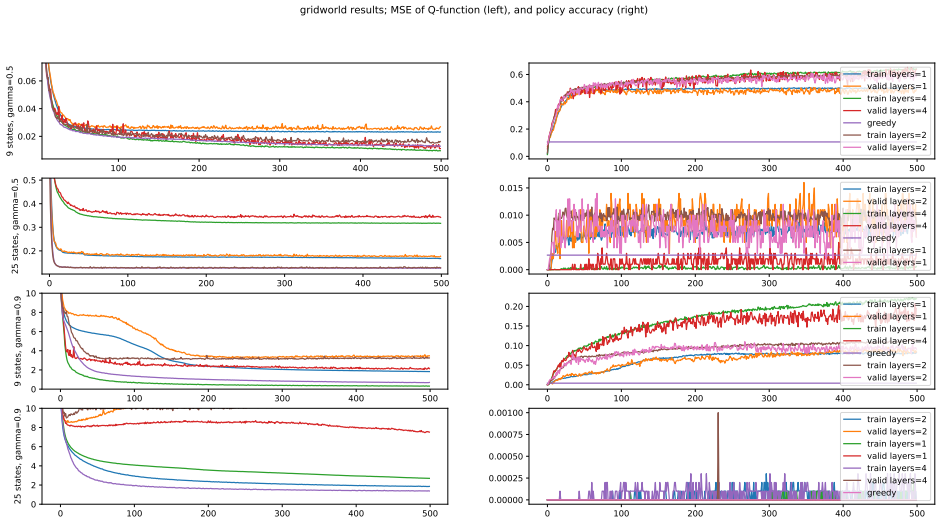

Figure 1: Results on gridworld environment

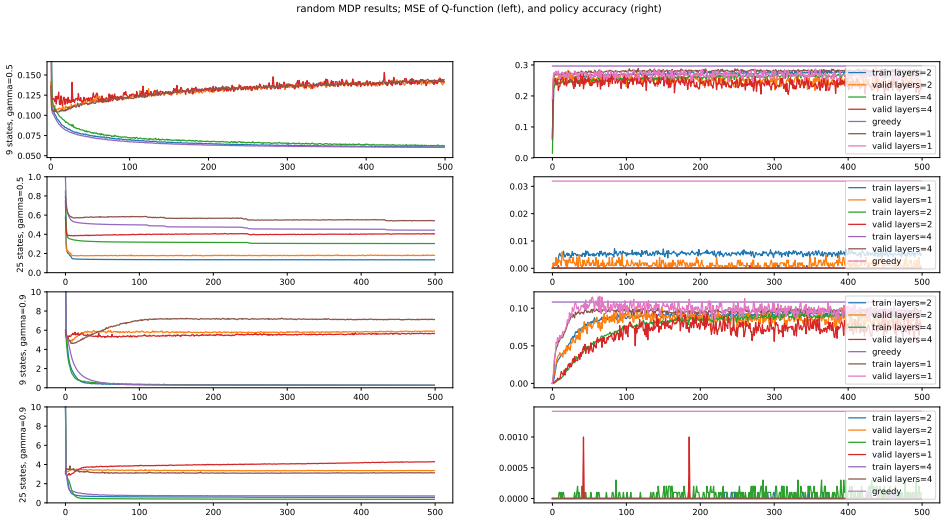

Figure 2: Results on random MDPs