224n Project: Natural Language Learning Supports Reinforcement Learning

Andrew Lampinen
Department of Psychology
Stanford University
Stanford, CA 94305
lampinen@stanford.edu

Abstract

Deep learning systems have recently achieved near human level performance on game playing. However, they require a much larger amount of data to achieve this than humans do, and often (though not always) still perform worse than humans. One factor that may account for this is the richer feedback humans receive, in particular we often receive a great deal of natural language instruction when learning tasks. In this paper, we explore providing auxiliary natural language instruction to a network in the simple game context of tic-tac-toe, and show that it accelerates learning.

1 Introduction

Neural networks are often optimized for performance on a single task. By contrast, human intelligence is fundamentally flexible, and applicable across a wide variety of tasks. It has been argued that this represents a fundamental difference in the representational structure of human intelligence and neural networks [5]. However, we have argued that this difference may arise from the richness of human experience, which can be seen as a vast set of interconnected and mutually supporting tasks [2]. We argue that human-like performance can arise from deep learning systems if they are given human-like experience.

In particular, natural language instruction is fundamental to rapid human learning in almost every context. It provides us with both direct instruction on simple explicit tasks, and with the scaffolding to build more complex skills. Lake and colleagues have argued that neural networks are too data hungry to be a good model for human behavior, or to perform sophisticated intelligence tasks without the support of other reasoning systems [5]. Can this difference be partly explained simply by the fact that deep learning systems are not generally given natural language feedback on tasks?

We explore this question in the simple game-playing context of tic-tac-toe. It is simple enough to train a neural network to play tic-tac-toe using a reinforcement learning algorithm, but suppose we give it in addition some natural language information about the task. This information may convey useful information about the task structure which is more difficult to obtain from the opaque feedback of a reinforcement learning signal. This might result in more rapid learning, and perhaps in representations that are more suitable for future learning.

2 Background/Related Work

Recent work has shown the power of combining neural networks and reinforcement learning, as with the Deep Q-Network [7]. There have been a variety of extensions and variations on this intended for training on multiple tasks [8, e.g.], with the focus of improving the flexibility of deep learning systems and allowing a single system to perform well on a variety of tasks.

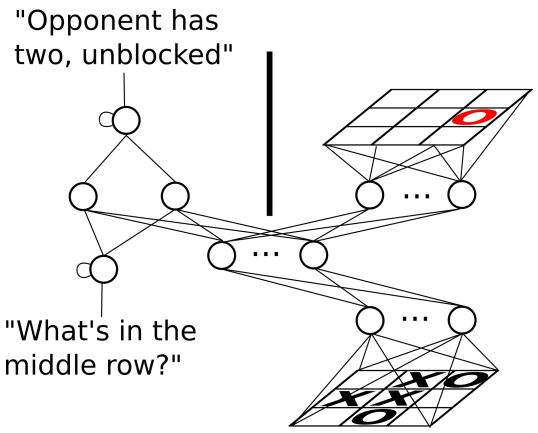

Figure 1: Sketch of our network architecture (note that this shows the action output as a single chosen action for simplicity, but in actuality the output is Q-values)

There have also been a number of studies recently focusing on the other benefit of multi-task learning: it can potentially improve performance on a single task more than training on that task alone would [6, e.g.]. In some sense, this can be seen as a more complex form of regularization. Requiring a network to solve two tasks will restrict the representations it forms to those that are beneficial for both tasks. If both tasks convey useful, distinct information about the world, this could potentially be quite beneficial.

For reinforcement learning specifically, a recent paper has suggested that adding various auxiliary rewards for tasks like correctly predicting what will happen can accelerate learning and improve final performance [3]. Kuhlman and colleagues [4] explored giving natural language advice to a reinforcment learning system, but built in the advice as rules which the system could "unlearn" by reducing their weight to the output, but did not allow for more flexible interactions of language and learning. To the best of our knowledge, this project is the only exploration of how training a reinforcement learning system on a purely auxiliary natural language task (i.e. one that does not directly affect the action outputs of the system) may affect learning.

3 Approach

We explored a simple version of incorporating explicit instruction into reinforcement learning. We trained a network to play tic-tac-toe using Q-learning, in the style of Mnih et al. [7], while simultaneously asking the network to produce descriptions through a parallel (sequence + visual input) to sequence learning task, where the description task and the Q-learning task share the two layers which process the visual input (see fig. 1 for a sketch of the architecture).

Despite its simplicity, we think there are a variety of features that make this a worthwhile demonstration task. First, the board structure of tic-tac-toe requires a player to pay attention to certain conjunctions of features (rows, columns, and diagonals), that are not immediately apparent (especially to the network, which does not have any built in awareness of spatial features). Furthermore, despite the simplicity of the game, tic-tac-toe affords the possibility for strategies such as double attacks which are reminiscient of more advanced games. Thus we believe there is a sufficiently rich structure in this task for multiple modes of feedback to be useful.

3.1 Q-learning

One common technique for reinforcement learning is to learn a Q-function, a mapping from (state, proposed action) pairs to the value of the action. These Q-values must satisfy the following recurrence relation called the Bellman equation (for simplicity here we assume rewards depend only on the current state, not the action taken in that state, and that a greedy policy is used which always

takes the highest value action):

$$Q(s_t, a_t) = R(s_{t+1}) + \max_{a \in A} Q(s_{t+1}, a)$$

where s_t and a_t are the state and action at time t, respectively, and $R(s)$ is the reward for state s.

In a simple task where all (state, action) pairs can be tabulated, a Q function can be built explicitly as a table, and updated until convergence according to the rule given above. However, another approach is to use a function approximator (such as a neural network) to estimate Q values given a state and action as input. (The update rule for the Q-values remains the same). Using a function approximator rather than a tabular representation allows for knowledge sharing across states, and because of this it allows for Q-learning to be applied even in cases where the state $\times$ action space is far too large to explicitly tabulate. This strategy has been applied successfully to tasks such as playing Atari video games [7], and this is the reinforcement learning strategy we use in this paper.

3.2 Architecture details

Our network can be broken down into a few components:

- **Visual parser:** a two layer neural network that takes as input the board state (3×3, an array consisting of 0s for empty squares, 1s for the network's pieces, and -1s for the opponents pieces), passes it through a 100 unit hidden layer, and then to its output layer (also 100 units). This output can be thought of as the networks "internal representation" of the board state. We denote this by the function V.
- **Q-approximator:** a two layer neural network that takes as input the "internal representation" from the visual parser, and outputs Q-values for the 9 possible plays on the board. We denote this by Q.
- **Question answerer:** this system takes a question and the output of the visual parser, and produces an answer. We denote this as a whole by A. It consists of several sub-components:
  - **Encoder:** a simple RNN that encodes the question input to the system (100 hidden units) using word embeddings (20 dimensional, initialized at random, same as those used for the decoder in the output). Its final state is part of the input to the integrator. We denote this by E.
  - **Integrator:** A single layer neural network (100 units) that takes as input the final state of the encoder and the "internal representation" produced by the visual parser, and produces as output the initial state for the decoder. We denote this by I.
  - **Decoder:** a simple RNN (100 hidden units) that outputs the answer to the current question as a sequence of word vectors (dimension 20). Its initial state is the output of the integrator, its first input is "GO," and subsequently it receives its previous output as input. We denote this network by D

Symbolically, the network works as follows:

$$\text{Estimated } Q\text{-values} = Q(V(\text{board state}))$$

$$\text{Answer} = A(\text{question}, V(\text{board state})) = D(I(E(\text{question}), V(\text{board state})))$$

The Tanh function was used as a non-linearity throughout (even for the output, our possible rewards were in the range $[-1, 1]$). We used dropout as a regularization at the output of V. We trained the Q-estimator by using mean squared-error on the Bellman equation above (i.e. on the difference between the Q-estimates at one point, and the reward plus optimal Q-estimate at the next move) as a loss function. We trained the question answerer using cross-entropy error on a softmax over the logits produced by multiplying its outputs by the word embeddings as a loss function.

3.3 Data generation

Because game playing involves interactions between two opponents, data generation is more complicated than on some other tasks. Fortunately, tic-tac-toe is simple, and both the plays and the descriptions could be generated on the fly.

Games: We trained the network by playing it against a hand-coded optimal opponent (opening plays were hand coded, and thereafter the opponent used an efficient set of rules about checking for threats from opponent, opportunities, etc. later in the game), using the Q-learning procedure stated above for the Q-net.

Descriptions: For the purposes of this project, we implemented the simplest possible form of linguistic interaction: answering purely descriptive questions. For example: "What's in the third row?" "My piece, empty square, opponents piece." In total, there were eight possible questions, corresponding to the three rows, three columns, and two diagonals relevant to the task. We asked the network to answer one question (randomly selected from this set of eight questions) before each of its moves.

3.4 Details

All sentences (input and output) were padded to length 10, input sentences were reversed. Weights and biases were randomly initialized from a normal distribution with mean 0 and standard deviation 0.1. We masked the Q-value output to include only legal plays before taking the max (i.e. only considering playing on squares where there was not already a piece on the board) – this improved learning early on. We used ϵ-greedy learning with $\epsilon = 0.1$. We trained all networks by stochastic gradient descent with minibatch size 1 (i.e. a single example at a time). We varied the learning rate for the Q-learning task, but all results in this paper are presented with a fixed learning rate ($\eta = 0.001$) for the description task.

4 Experiments

We evaluated the network on whether or not it successfully achieved perfect performance (no losses) against the optimal opponent within 400 games, and if it did succeed, how many training epochs it took to do so (each epoch consisted of playing 20 games to completion). We compared the performance of the network trained with descriptions to two comparison networks:

1. **Basic:** The Q-network without the description task.

2. **Control:** The Q-network with a different description task – counting the number of pieces on the board. In this task there was only a single question ("How many are there?") and responses were of the form ("There are five"). This active control is to test for a possible effect of just having any other task. It's possible that any benefits from the description task would be due to overcoming the vanishing gradients and having learning signals reach the first layer more quickly, having a different auxiliary task which still relies on parsing the board state will fix this.

(On a given run, all networks were initialized with identical random seeds, and weights were initialized in an order which ensured that shared weights between two networks were the same.) Our hypothesis was that the description network would outperform both of the control networks as measured by both metrics (number of successes and time to success), although they might require different hyperparameter settings. We would like to highlight that given enough time (and somewhat reasonable hyperparameters), all the networks would learn this simple task. However, exploring which learn more rapidly can give us insight into which strategies we should incorporate in more complex tasks or in tasks where data is more difficult to collect.

5 Results

η	basic	control	description
0.05	0%	0%	0%
0.01	50%	40%	**65%**
0.005	5%	10%	**79%**
0.001	0%	**15%**	**15%**

(a) Percent successful runs (runs where optimal performance was reached)

η	basic	control	description
0.05	N/A	N/A	N/A
0.01	14.1	13.8	**11.4**
0.005	19.0	12.5	**9.5***
0.001	N/A	16.0	**14.3**

(b) Mean epochs to success on runs where optimal performance was reached

Table 1: Results across varying learning rates (bold indicates best result in that row, overall best result is starred)

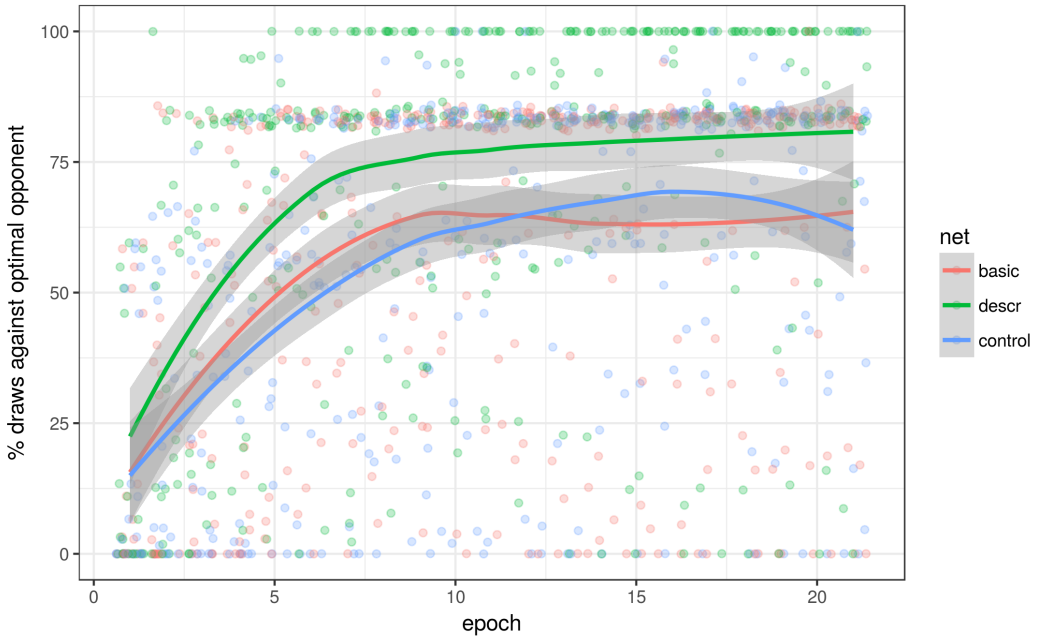

Figure 2: Percent draws vs. training epoch for each network with its optimal hyperparameter settings (average curves and individual data points from each run)

The description network outperformed the other networks, as measured by both number of successful runs (runs where the network achieved optimal performance – always drawing against the optimal opponent – see table 1a) and mean time to success in the successful runs (see table 1b). Comparing the best result obtained with the description network to the best result obtained with the basic network, we see a speed-up of 33%, from an average of 14.1 epochs (282 games) to success for the basic network to an average of 9.5 epoch (190 games) for the description network. See fig. 2 for a plot of the results with the optimal hyperparameter settings for each network.

The control network performed comparably to the basic network, although it appeared to have a slight avantage over it at very low learning rates, likely due to the extra weight updates allowing the signals to propagate earlier in the learning process. At the learning rates where the basic network was successful, the control task did not appear to be beneficial.

η	control	description
0.05	N/A	N/A
0.01	1.1	19.0
0.005	1.2	25.2
0.001	1.5	23.3

Table 2: Final loss on natural language task (average per game loss on successful runs)

The description network did not completely solve the description task in the time allotted to it. The loss at the end of training averaged around 19-25, corresponding to the network incorrectly identifying a few squares on a few of the questions, see discussion.

6 Discussion

The natural language feedback appears to be beneficial, and comparing to the control task, it seems that its benefits are not attributable solely to allowing the gradients to propagate back more rapidly. Why is the description task beneficial? One possibility is that it conveys information about the structure of the "world" that is not easy to obtain from a reinforcement learning signal. Having the additional information from the natural language task could inform the system that rows, columns, and diagonals are useful configurations to pay attention to. Without this additional signal, it may be harder for the network to figure out which sets of squares it should and should not pay attention to.

We find it interesting to consider the extent to which various forms of feedback like this may explain the difference between human and machine learning. We obtained a speedup of 1/3 the learning time on a simple task by incorporating a very simple supplementary task. How much more speedup could we achieve in a more complex system, on a more complicated task? How much more benefit could we obtain by having multiple forms of feedback, such as language and prediction as suggested by Jaderberg and colleagues [3]? We speculate that building human-like intelligence will require rich training on many complementary tasks.

6.1 Future directions

It is interesting to note that these benefits occurred despite the fact that the description loss does not generally approach zero within the period of training on this task for the description network, instead it pretty consistently averages around 19-25 (per game), which corresponds to the network usually getting one location wrong in several of the questions throughout the game, for example saying a square is empty when it actually has an opponents piece. It is perhaps not surprising that this would be the case – there is considerable ambiguity in mapping descriptions onto board configurations. For humans this is solved by explicit instructions about which positions to pay attention to, and incorporating explicit instructions like these would be an interesting direction for future research.

We think these results alone are very promising, however. Tic-tac-toe is a very simple game, and our descriptions were very simple, so it was not clear we would see any improvement whatsoever. It would be exciting to extend this to a richer game (such as chess or go), where the descriptions could include more interesting configurational information (e.g. tactical or strategic information). This is another interesting direction for future research.

One of the advantages to using natural language is that it can provided by relatively na ive end users. For example, one could imagine that a more complex version of this system could be useful if and when reinforcement learning systems are incorporated in autonomous vehicles – if a near accident occurs, the human riders could provide feedback on what the causes were that could help the system learn more rapidly to reduce danger in the future.

7 Conclusions

We have demonstrated that providing an auxiliary natural language learning task to a reinforcement learning system can accelerate learning on a game-playing task. We began this paper by highlighting the importance of language for human learning, and we think this represents an exciting step towards

bringing this feature of human intelligence to deep learning. Our results suggest that incorporating natural language instruction may be a fruitful technique for making complex tasks more rapidly and easily learnable.

References

[1] DIENES, Z., ALTMANN, G. T. M., AND GAO, S.-J. Mapping across Domains Without Feedback: A Neural Network Model of Transfer of Implicit Knowledge. *Cognitive Science 23*, 1 (1999), 53–82.

[2] HANSEN, S. S., LAMPINEN, A., SURI, G., AND MCCLELLAND, J. L. Building on Prior Knowledge Without Building it in. *Accepted to Behavioral and Brain Sciences*.

[3] JADERBERG, M., MNIH, V., CZARNECKIM, M. W., SCHAUL, T., LEIBO, J. Z., SILVER, D., AND KAVUKCUOGLU, K. Reinforcement Learning with Unsupervised Auxiliary Tasks. *arXiv* (2016), 1–11.

[4] KUHLMANN, G., STONE, P., MOONEY, R., AND SHAVLIK, J. Guiding a Reinforcement Learner with Natural Language Advice : Initial Results in RoboCup Soccer. 30–35.

[5] LAKE, B. M., ULLMAN, T. D., TENENBAUM, J. B., AND GERSHMAN, S. J. Building Machines that learn and think like people. *arXiv:1604.00289v1[cs.AI]* (2016), 1–55.

[6] LUONG, M.-T., LE, Q. V., SUTSKEVER, I., VINYALS, O., AND KAISER, L. Multi-task Sequence to Sequence Learning. *Iclr* (2016), 1–9.

[7] MNIH, V., KAVUKCUOGLU, K., SILVER, D., RUSU, A. A., VENESS, J., BELLEMARE, M. G., GRAVES, A., RIEDMILLER, M., FIDJELAND, A. K., OSTROVSKI, G., PETERSEN, S., BEATTIE, C., SADIK, A., ANTONOGLOU, I., KING, H., KUMARAN, D., WIERSTRA, D., LEGG, S., AND HASSABIS, D. Human-level control through deep reinforcement learning. *Nature 518*, 7540 (2015), 529–533.

[8] RUSU, A. A., GOMEZ COLMENAREJO, S., GULCEHRE, C., DESJARDINS, G., KIRKPATRICK, J., PASCANU, R., MNIH, V., KAVUKCUOGLU, K., AND HADSELL, R. Policy Distillation. *arXiv* (2015), 1–12.