Deeply AggreVaTeD: Differentiable Imitation Learning for Sequential Prediction

Wen Sun
Robotics Institute
Carnegie Mellon University
wensun@cs.cmu.edu

Arun Venkatraman
Robotics Institute,
Carnegie Mellon University
arunvenk@cs.cmu.edu

Geoffrey J. Gordon
Machine Learning Department
Carnegie Mellon University
ggordon@cs.cmu.edu

Byron Boots
Institute for Robotics and Intelligent Machines
Georgia Institute of Technology
bboots@cc.gatech.edu

J. Andrew Bagnell
Robotics Institute
Carnegie Mellon University
dbagnell@cs.cmu.edu

Abstract

Researchers have demonstrated state-of-the-art performance in sequential decision making problems (e.g., robotics control, sequential prediction) with deep neural network models. One often has access to near-optimal oracles that achieve good performance on the task during training. We demonstrate that *AggreVaTeD* — a policy gradient extension of the Imitation Learning (IL) approach of [1] — can leverage such an oracle to achieve faster and better solutions with less training data than a less-informed Reinforcement Learning (RL) technique. Using both feedforward and recurrent neural predictors, we present stochastic gradient procedures on a sequential prediction task, dependency-parsing from raw image data, as well as on various high dimensional robotics control problems. We also provide a comprehensive theoretical study of IL that demonstrates we can expect up to *exponentially* lower sample complexity for learning with *AggreVaTeD* than with RL algorithms, which backs our empirical findings. Our results and theory indicate that the proposed approach can achieve superior performance with respect to the oracle when the demonstrator is sub-optimal.

Keywords: Imitation Learning, Sequential Prediction, Deep Learning

1 Introduction

A fundamental challenge in artificial intelligence, robotics, and language processing is to reason, plan, and make a sequence of decisions to minimize accumulated cost, achieve a long-term goal, or optimize for a loss acquired only after many predictions. Reinforcement Learning (RL), especially deep RL, has dramatically advanced the state of the art in sequential decision making in high-dimensional robotics control tasks as well as in playing video and board games [2, 3]. Although conventional supervised learning of deep models has been pivotal in advancing performance in sequential prediction problems, researchers are beginning to utilize deep RL methods to achieve higher performance [4, 5]. Often in sequential prediction tasks, future predictions from the learner are dependent on the history of previous predictions; thus, a poor prediction early on can yield high accumulated loss (cost) for future predictions. Viewing the predictor as a *policy* π, deep RL algorithms are able to reason about the future accumulated cost in a sequential decision making process whether it is a traditional robotics control problem or a sequential structured prediction task.

In contrast with reinforcement learning methods, well-known *imitation learning* (IL) and sequential prediction algorithms such as SEARN [6], DAgger [7], AggreVaTe [1] reduce the sequence prediction problem to supervised learning by leveraging one special property of the sequence prediction problem: at training time we usually have a (near) optimal *cost-to-go oracle*. At any point along the sequential prediction process (i.e. state or partially completed sequential prediction), the oracle is able to select the next (near)-best action. For robotics control problems, this oracle may come from a human expert guiding the robot during the training phase [8] or from an optimal MDP solver that either may be too slow to use at test time or leverages information unavailable at test time (e.g., ground truth). Similarly, for sequential prediction problems, an oracle can be constructed by optimization (e.g., beam search) or by a clairvoyant greedy algorithm that is near-optimal on the task specific performance metric (e.g., cumulative reward, IoU, Unlabeled Attachment Score, BLEU) given the training data's ground truth.

Such an oracle, however, is only available during training time (e.g., when there is access to ground truth). Thus, the goal of IL is to learn a policy $\hat{\pi}$, with the help of the oracle (π^*, Q^*) during the training session, such that $\hat{\pi}$ achieves similar quality performance at test time when the oracle is unavailable. In contrast to IL, reinforcement learning methods often initialize with an random policy π_0 or cost-to-go (accumulated loss) Q_0 predictor which may be far from the optimal. The optimal policy (or cost-to-go) must be found through a tradeoff of dithering, directed exploration, and exploitation. The existence of oracle can be exploited to alleviate blind learning by trial and error: one can *imitate* the oracle to speed up learning process by significantly reducing exploration. *Interactive* approaches to IL such as SEARN [6], DAgger [7], and AggreVaTe [1] interleave learning and testing procedures to overcome the data mismatch issue and, as a result, work well in practical applications. Furthermore, these interactive approaches can provide strong theoretical guarantees between training time loss and test time performance through a reduction to no-regret online learning.

In this work, we introduce *AggreVaTeD*, a *differentiable* version of AggreVaTe (Aggregate Values to Imitate [1]) which extends interactive IL for use in sequential prediction and challenging continuous robot control tasks. We provide two gradient update procedures: a regular gradient update developed from Online Gradient Descent (OGD) [9] and a natural gradient update [10, 11] which we show is closely related to Exponential Gradient Descent (EG), another popular no-regret algorithm that enjoys an almost dimension-free property. AggreVaTeD leverages the oracle to learn rich polices that can be represented by complicated non-linear function approximators. Our experiments with deep neural networks on various robotics control simulators and on a dependency parsing sequential prediction task show that AggreVaTeD can achieve expert-level performance and even *super-expert* performance when the oracle is sub-optimal, a result rarely achieved by non-interactive IL approaches. The differentiable nature of AggreVaTeD additionally allows us to employ LSTM-based policies to handle partially observable settings (e.g., observe only partial robot state). Empirical results demonstrate that by leveraging an oracle, IL can learn much faster than RL in practice.

In addition to providing a set of practical algorithms, we develop a comprehensive theoretical study of IL on discrete MDPs. Specifically we show that we can expect up to exponentially better sample efficiency for IL than any RL algorithm. We refer readers to the full version of the paper [12] [1] for detailed analysis of IL.

2 Preliminaries

Formally, a finite-horizon Markov Decision Process (MDP) is defined as $(\mathcal{S}, \mathcal{A}, P, C, \rho_0, H)$. Here, $\mathcal{S}$ is a set of S many states and $\mathcal{A}$ is a set of A actions; given time step t, P_t is the transition dynamics such that for any $s_t \in \mathcal{S}, s_{t+1} \in \mathcal{S}, a_t \in \mathcal{A}$, $P_t(s_{t+1}|s_t, a_t)$ is the probability of transiting to state s_{t+1} from state s_t by taking action a_t at step t; C is the cost distribution such that a cost c_t at step t is sampled from $C_t(\cdot|s_t, a_t)$. Finally, we denote $\bar{c}_t$ as the expected cost, ρ_0 as the initial distribution of states, and $H \in \mathbb{N}^+$ as the finite horizon (max length) of the MDP. We define a stochastic policy π such that for any state $s \in \mathcal{S}$, $\pi(\cdot|s) \in \Delta(A)$, where $\Delta(A)$ is a A-dimension simplex, conditioned on state s. $\pi(a|s) \in [0,1]$ outputs the probability of taking action a at state s. The distribution $\rho_\pi(\tau)$ of trajectories $\tau = (s_1, a_1, \ldots, a_{H-1}, s_H)$ and the distribution $d_t^\pi(s_t)$ of the states at time step t under policy π are defined as:

$$\rho_\pi(\tau) = \rho_0(s_1) \prod_{t=2}^{H} \pi(a_{t-1}|s_{t-1}) P_{t-1}(s_t|s_{t-1}, a_{t-1}), \quad d_t^\pi(s_t) = \sum_{\{s_i, a_i\}_{i \le t-1}} \rho_0(s_1) \prod_{i=1}^{t-1} \pi(a_i|s_i) P_i(s_{i+1}|s_i, a_i). \tag{1}$$

[1] The full version of the paper can be found at `http://www.cs.cmu.edu/~wensun/DifferentiableAggrevate.pdf`

Note that the summation above can be replaced by an integral if the state or action space is continuous. The expected average cost $\mu(\pi)$ of a policy π and the state-action value $Q_t^\pi(s,a)$ for policy π are defined as:

$$\mu(\pi) = \mathop{\mathbb{E}}_{\tau\sim\rho_\pi}[\sum_{t=1}^{H}\bar{c}_t(s_t,a_t)] = \sum_{t=1}^{H}\mathop{\mathbb{E}}_{s\sim d_t^\pi(s),a\sim\pi(a|s)}[\bar{c}_t(s,a)]; \quad Q_t^\pi(s_t,a_t) = \bar{c}_t(s_t,a_t) + \mathop{\mathbb{E}}_{s\sim P_t(\cdot|s_t,a_t),a\sim\pi(\cdot|s)} Q_{t+1}^\pi(s,a), \tag{2}$$

where the expectation is taken over the randomness of the policy π and the MDP.

We define π^* as the expert policy (e.g., human demonstrators, search algorithms equipped with ground-truth) and $Q_t^*(s,a)$ as the expert's cost-to-go oracle (note π^* may not be optimal, i.e., $\pi^* \not\in \arg\min_\pi \mu(\pi)$). Throughout the paper, we assume $Q_t^*(s,a)$ is known or can be estimated without bias (e.g., by rolling out π^*: starting from state s, applying action a, and then following π^* for $H-t$ steps). When π is represented by a function approximator, we use the notation π_θ to represent the policy parametrized by $\theta\in\mathbb{R}^d$: $\pi(\cdot|s;\theta)$. In this work we specifically consider optimizing policies in which the parameter dimension d may be large. We also consider the partially observable setting in our experiments, where the policy $\pi(\cdot|o_1,a_1,...,o_t;\theta)$ is defined over the whole history of partial observations and actions (o_t is generated from the hidden state s_t). We use an LSTM-based policy [13] where the LSTM's hidden states provide a compressed feature of the history.

3 Differentiable Imitation Learning

Policy based imitation learning aims to learn a policy $\hat{\pi}$ that approaches the performance of the expert π^* at testing time when π^* is not available anymore. In order to learn rich policies such as with LSTMs or deep networks [2], we derive a *policy gradient* method for imitation learning and sequential prediction. To do this, we leverage the reduction of IL and sequential prediction to online learning as shown in [1] to learn policies represented by expressive differentiable function approximators.

The fundamental idea in [1] is to use a no-regret online learner to update policies using the following loss function at each episode n:

$$\ell_n(\pi) = \frac{1}{H}\sum_{t=1}^{H}\mathop{\mathbb{E}}_{s_t\sim d_t^{\pi_n}}\Big[\mathop{\mathbb{E}}_{a\sim\pi(\cdot|s_t)}[Q_t^*(s_t,a)]\Big]. \tag{3}$$

The loss function intuitively encourages the learner to find a policy that minimize the expert's cost-to-go *under the state distribution resulting from the current learned policy* π_n. Specifically, [1] suggest an algorithm named *AggreVaTe* (Aggregate Values to Imitate) that uses Follow-the-Leader (FTL) [14] to update policies:$\pi_{n+1} = \arg\min_{\pi\in\Pi}\sum_{i=1}^{n}\ell_n(\pi)$, where Π is a pre-defined convex policy set. When $\ell_n(\pi)$ is strongly convex with respect to π and $\pi^*\in\Pi$, after N iterations AggreVaTe with FTL can find a policy $\hat{\pi}$: $\mu(\hat{\pi}) \le \mu(\pi^*) - \epsilon_N + O(\ln(N)/N)$, where $\epsilon_N = [\sum_{n=1}^{N}\ell_n(\pi^*) - \min_\pi\sum_{n=1}^{N}\ell_n(\pi)]/N$. Note that $\epsilon_N \ge 0$ and the above inequality indicates that $\hat{\pi}$ can outperform π^* when π^* is not (locally) optimal (i.e., $\epsilon_n > 0$).

A simple implementation of AggreVaTe that aggregates the values (as the name suggests) will require an exact solution to a batch optimization procedure in each episode. When π is represented by large, non-linear function approximators, the $\arg\min$ procedure generally takes more and more computation time as n increases. Online Mirror Descent (OMD) [14] is popular for online learning due to its efficiency. Therefore we consider two special cases of OMD for optimizing sequence of losses $\{\ell_n(\pi)\}_n$: Online Gradient Descent (OGD) [9] and Exponential Gradient Descent (EG) [14], which lead to a regular stochastic policy gradient descent algorithm and a natural policy gradient algorithm, respectively.

3.1 Online Gradient Descent

For discrete actions, the gradient of $\ell_n(\pi_\theta)$ (Eq. 3) with respect to the parameters θ of the policy can be computed as $\nabla_\theta\ell_n(\theta) = \frac{1}{H}\sum_{t=1}^{H}\mathbb{E}_{s_t\sim d_t^{\pi_{\theta_n}}}\sum_a\nabla_\theta\pi(a|s_t;\theta)Q_t^*(s_t,a)$. For continuous action spaces, we cannot simply replace the summation by integration since in practice it is impossible to evaluate $Q_t^*(s,a)$ for infinitely many a, so, instead, we use importance weighting to re-formulate ℓ_n (Eq. 3) as

$$\ell_n(\pi_\theta) = \frac{1}{H}\sum_{t=1}^{H}\mathop{\mathbb{E}}_{s\sim d_t^{\pi_{\theta_n}},a\sim\pi(\cdot|s;\theta_n)}\frac{\pi(a|s;\theta)}{\pi(a|s;\theta_n)}Q_t^*(s,a) = \frac{1}{H}\mathop{\mathbb{E}}_{\tau\sim\rho_{\pi_{\theta_n}}}\sum_{t=1}^{H}\frac{\pi(a_t|s_t;\theta)}{\pi(a_t|s_t;\theta_n)}Q_t^*(s_t,a_t). \tag{4}$$

With this reformulation, the gradient with respect to θ is: $\nabla_\theta\ell_n(\theta) = \frac{1}{H}\mathbb{E}_{\tau\sim\rho_{\pi_{\theta_n}}}\sum_{t=1}^{H}\frac{\nabla_\theta\pi(a_t|s_t;\theta)}{\pi(a_t|s_t;\theta_n)}Q_t^*(s_t,a_t)$. The above gradient computation enables an efficient update procedure with OGD: $\theta_{n+1} = \theta_n - \eta_n\nabla_\theta\ell_n(\theta)|_{\theta=\theta_n}$, where η_n is the learning rate.

3.2 Policy Updates with Natural Gradient Descent

We derive a natural gradient update procedure for imitation learning inspired by the success of natural gradient descent in RL [10, 11, 2]. We can show that Exponential Gradient Descent (EG) can be leveraged to speed up imitation learning in discrete MDPs. Then we extend EG to continuous MDPs, where we show that, with three steps of approximation, EG leads to a natural gradient update procedure. We skip the derivation of EG in discrete MDP due to space limit and refer readers to [12] for details.

We consider the following proximal operator:

$$\theta = \arg\min_{\theta} \frac{1}{H}\sum_{t=1}^{H} \mathop{\mathbb{E}}_{s\sim d_t^{\pi_{\theta_n}}} \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s;\theta)} [Q_t^*(s,a)] + \mathop{\mathbb{E}}_{s\sim \bar{d}^{\pi_{\theta_n}}} KL(\pi_\theta || \pi_{\theta_n})/\eta_n. \tag{5}$$

In order to solve for θ from Eq. 5, we apply several approximations. We first approximate $\ell_n(\theta)$ (the first part of the RHS of the above equation) by its first-order Taylor expansion: $\ell_n(\theta) \approx \ell_n(\theta_n) + \nabla_{\theta_n}\ell_n(\theta_n)\cdot(\theta-\theta_n)$. When θ and θ_n are close, this is a valid local approximation. Second, we replace $KL(\pi_\theta||\pi_{\theta_n})$ by $KL(\pi_{\theta_n}||\pi_\theta)$, which is a local approximation since $KL(q||p)$ and $KL(p||q)$ are equal up to the second order [15, 2]. Third, we approximate $KL(\pi_{\theta_n}||\pi_\theta)$ by a second-order Taylor expansion around θ_n, such that we can approximate the penalization using the Fisher information matrix: $\mathbb{E}_{s\sim\bar{d}^{\pi_{\theta_n}}} KL(\pi_{\theta_n}||\pi_\theta) \approx (1/2)(\theta-\theta_n)^T I(\theta_n)(\theta-\theta_n)$, where the Fisher information matrix [11]: $I(\theta_n) = \frac{1}{H^2}\mathbb{E}_{\tau\sim\rho_{\pi_{\theta_n}}} \nabla_{\theta_n}\log(\rho_{\pi_{\theta_n}}(\tau))\nabla_{\theta_n}\log(\rho_{\pi_{\theta_n}}(\tau))^T$, where $\nabla_\theta \log(\rho_{\pi_\tau}(\tau))$ is the gradient of the log likelihood of the trajectory τ which can be computed as $\sum_{t=1}^{H}\nabla_\theta\log(\pi_\theta(a_t|s_t))$.

Inserting these three approximations into Eq. 5, and solving for θ, we reach the following update rule $\theta_{n+1} = \theta_n - \eta_n I(\theta_n)^{-1}\nabla_\theta \ell_n(\theta)|_{\theta=\theta_n}$,

4 Sample-Based Practical Algorithms

In the previous section, we derived a regular gradient update procedure and a natural gradient update procedure for IL. Note that all of the computations of gradients and Fisher information matrices assumed it was possible to exactly compute expectations including $\mathbb{E}_{s\sim d^\pi}$ and $\mathbb{E}_{a\sim\pi(a|s)}$. In this section, we provide practical algorithms where we approximate the gradients and Fisher information matrices using finite samples collected during policy execution.

4.1 Gradient and Fisher Information Matrix Estimation and Variance Reduction

We consider an episodic framework where given a policy π_n at episode n, we roll out π_n K times to collect K trajectories $\{\tau_i^n\}$, for $i\in[K]$, $\tau_i^n = \{s_1^{i,n}, a_1^{i,n}, ...\}$. For gradient $\nabla_\theta \ell_n(\theta)|_{\theta=\theta_n}$ we can compute an unbiased estimate using $\{\tau_i^n\}_{i\in[K]}$:

$$\tilde{\nabla}_{\theta_n} = \frac{1}{HK}\sum_{i=1}^{K}\sum_{t=1}^{H}\sum_{a}\nabla_{\theta_n}\pi_{\theta_n}(a|s_t^{i,n})Q_t^*(s_t^{i,n},a), \quad \tilde{\nabla}_{\theta_n} = \frac{1}{HK}\sum_{i=1}^{K}\sum_{t=1}^{H}\frac{\nabla_{\theta_n}\pi_{\theta_n}(a_t^{i,n}|s_t^{i,n})}{\pi_{\theta_n}(a_t^{i,n}|s_t^{i,n})}Q_t^*(s_t^{i,n},a_t^{i,n}). \tag{6}$$

for discrete and continuous setting respectively. When we can compute $V_t^*(s)$ (e.g., $\min_a Q_t^*(s,a)$), we can replace $Q_t^*(s_t^{i,n},a)$ in Eq. 6 by the state-action advantage function $A_t^*(s_t^{i,n},a) = Q_t^*(s_t^{i,n},a) - V_t^*(s_t^{i,n})$. Note that using advantage function in practice leads to smaller variance.

The Fisher information matrix $I(\theta_n)$ is approximated as:

$$\tilde{I}(\theta_n) = \frac{1}{H^2K}\sum_{i=1}^{K}\nabla_{\theta_n}\log(\rho_{\pi_{\theta_n}}(\tau_i))\nabla_{\theta_n}\log(\rho_{\pi_{\theta_n}}(\tau_i))^T = S_nS_n^T, \tag{7}$$

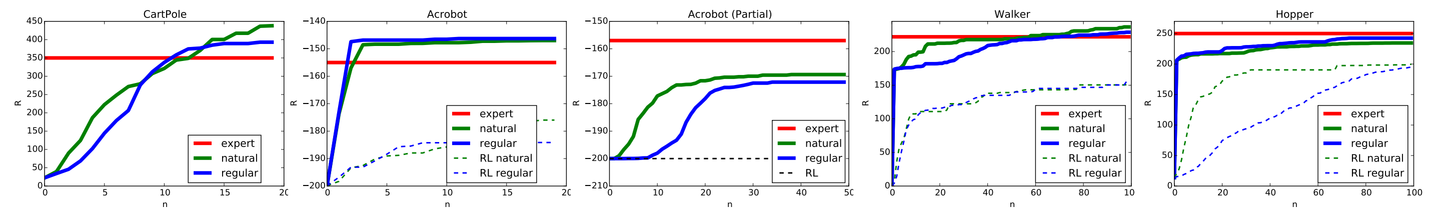
where, for notation simplicity, we denote S_n as a $d\times K$ matrix where the i's th column is $\nabla_{\theta_n}\log(\rho_{\pi_{\theta_n}}(\tau_i))/(H\sqrt{K})$. Namely the Fisher information matrix is represented by a sum of K rank-one matrices. For large policies represented by neural networks, $K \ll d$, and hence $\tilde{I}(\theta_n)$ a low rank matrix. One can find the descent direction δ_{θ_n} by solving the linear system $S_nS_n^T\delta_{\theta_n} = \tilde{\nabla}_{\theta_n}$ for δ_{θ_n} using Conjugate Gradient (CG) with a fixed number of iterations, which is equivalent to solving the above linear systems using the Partial Least Square. Our representation of the Fisher matrix is in the form of $S_nS_n^T$ and in CG we never need to explicitly compute or store $S_nS_n^T$ which requires d^2 space and time. Instead, we only compute and store S_n ($O(Kd)$) and the total computational time is still $O(K^2d)$. The learning-rate for natural gradient descent can be chosen as $\eta_n = \sqrt{\delta_{KL}/(\tilde{\nabla}_{\theta_n}^T\delta_{\theta_n})}$, such that $KL(\rho_{\pi_{\theta_{n+1}}(\tau)}||\rho_{\pi_{\theta_n}(\tau)}) \approx \delta_{KL} \in \mathbb{R}^+$.

4.2 Differentiable Imitation Learning: AggreVaTeD

The imitation learning framework *AggreVaTeD* is extremely simple: every iteration, we roll in the current policy π_n to generate K trajectories; with Q^*, we then compute the descent direction δ_{θ_n} which is $\tilde{\nabla}_{\theta_n}$ (regular gradient) or $\tilde{I}(\theta_n)^{-1}\tilde{\nabla}_{\theta_n}$ (natural gradient); finally we update the policy $\theta_{n+1} = \theta_n - \eta_n\delta_{\theta_n}$. The computational complexity of each episode scales as $O(d)$

5 Experiments

We evaluate our algorithms on robotics simulations from OpenAI Gym [17] and on Handwritten Algebra Dependency Parsing [16]. We report reward instead of cost, since OpenAI Gym by default uses reward and dependency parsing aims to maximize UAS score. For RL we use REINFORCE [18] and Truncated Natural Policy Gradient (TNPG) [13]. We refer readers to [12] for detailed experiment and neural network (e.g., feedforward and LSTMs) setup.



(a) Cartpole (b) Acrobot (c) Acrobot (POMDP) (d) Walker (e) Hopper

Figure 1: {Performance (cumulative reward R on y-axis) versus number of episodes (n on x-axis) of AggreVaTeD (blue and green), experts (red), and RL algorithms (dotted) on different robotics simulators.}

Arc-Eager	AggreVaTeD (LSTMs)	AggreVaTeD (NN)	SL-RL (LSTMs)	SL-RL(NN)	RL (LSTMs)	RL (NN)	DAgger	SL (LSTMs)	SL (NN)	Random
Regular	**0.924**$\pm$0.10	0.851$\pm$0.10	0.826$\pm$ 0.09	0.386$\pm$0.1	0.256$\pm$0.07	0.227$\pm$0.06	0.832$\pm$0.02	0.813$\pm$0.1	0.325$\pm$0.2	$\sim$0.150
Natural	0.915$\pm$0.10	0.800$\pm$0.10	0.824$\pm$0.10	0.345$\pm$0.1	0.237$\pm$0.07	0.241$\pm$0.07				

Table 1: {Performance (UAS) of different approaches on handwritten algebra dependency parsing. *SL* stands for supervised learning using expert's samples: maximizing the likelihood of expert's actions under the sequences generated by expert itself. *SL-RL* means RL with initialization using SL. *Random* stands for the initial performances of random policies (LSTMs and NN). The performance of DAgger with Kernel SVM is from [16].}

Fig. 1 shows AggreVaTeD is able to learn expert-level polices for complex continuous robotics control tasks and the learning speed is much faster than the corresponding RL algorithms. Also we can see that AggreVaTeD can outperform the experts. Table 1 shows the performance of different approaches with different neural network polices in dependency parsing for handwritten algebra. For dependency parsing, a near-optimal expert can be constructed with the access to the ground-truth parse trees in the training data. AggreVaTeD with a LSTM-based policy achieves 97% of the expert's performance (UAS$\approx$95).

References

[1] Stephane Ross and J Andrew Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*, 2014.

[2] John Schulman, Sergey Levine, Pieter Abbeel, Michael I Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, pages 1889–1897, 2015.

[3] David Silver et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

[4] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *ICLR 2016*, 2015.

[5] Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*, 2016.

[6] Hal Daumé III, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine learning*, 2009.

[7] Stéphane Ross, Geoffrey J Gordon, and J.Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

[8] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML*, page 1. ACM, 2004.

[9] Martin Zinkevich. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. In *ICML*, 2003.

[10] Sham Kakade. A natural policy gradient. *NIPS*, 2002.

[11] J Andrew Bagnell and Jeff Schneider. Covariant policy search. IJCAI, 2003.

[12] Wen Sun, Arun Venkatraman, Geoffrey Gordon, Byron Boots, and J. Andrew Bagnell. Deeply aggrevated: Differentiable imitation learning for sequential prediction. 2017.

[13] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *ICML*, 2016.

[14] Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 2012.

[15] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *ICML*, 2002.

[16] James A Duyck and Geoffrey J Gordon. Predicting structure in handwritten algebra data from low level features. *Data Analysis Project Report, MLD, CMU*, 2015.

[17] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[18] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 1992.