DEEP REINFORCEMENT LEARNING

Yuxi Li (yuxili@gmail.com)

ABSTRACT

We discuss deep reinforcement learning in an overview style. We draw a big picture, filled with details. We discuss six core elements, six important mechanisms, and twelve applications, focusing on contemporary work, and in historical contexts. We start with background of artificial intelligence, machine learning, deep learning, and reinforcement learning (RL), with resources. Next we discuss RL core elements, including value function, policy, reward, model, exploration vs. exploitation, and representation. Then we discuss important mechanisms for RL, including attention and memory, unsupervised learning, hierarchical RL, multi-agent RL, relational RL, and learning to learn. After that, we discuss RL applications, including games, robotics, natural language processing (NLP), computer vision, finance, business management, healthcare, education, energy, transportation, computer systems, and, science, engineering, and art. Finally we summarize briefly, discuss challenges and opportunities, and close with an epilogue. [1]

Keywords: deep reinforcement learning, deep RL, algorithm, architecture, application, artificial intelligence, machine learning, deep learning, reinforcement learning, value function, policy, reward, model, exploration vs. exploitation, representation, attention, memory, unsupervised learning, hierarchical RL, multi-agent RL, relational RL, learning to learn, games, robotics, computer vision, natural language processing, finance, business management, healthcare, education, energy, transportation, computer systems, science, engineering, art

[1] Work in progress. Under review for Morgan & Claypool: Synthesis Lectures in Artificial Intelligence and Machine Learning. This manuscript is based on our previous deep RL overview (Li, 2017). It benefits from discussions with and comments from many people. Acknowledgements will appear in a later version.

CONTENTS

1 INTRODUCTION

Reinforcement learning (RL) is about an agent interacting with the environment, learning an optimal policy, by trial and error, for sequential decision making problems, in a wide range of fields in natural sciences, social sciences, and engineering (Sutton and Barto, 1998; 2018; Bertsekas and Tsitsiklis, 1996; Bertsekas, 2012; Szepesvári, 2010; Powell, 2011).

The integration of reinforcement learning and neural networks has a long history (Sutton and Barto, 2018; Bertsekas and Tsitsiklis, 1996; Schmidhuber, 2015). With recent exciting achievements of deep learning (LeCun et al., 2015; Goodfellow et al., 2016), benefiting from big data, powerful computation, new algorithmic techniques, mature software packages and architectures, and strong financial support, we have been witnessing the renaissance of reinforcement learning (Krakovsky, 2016), especially, the combination of deep neural networks and reinforcement learning, i.e., deep reinforcement learning (deep RL).[2]

Deep learning, or deep neural networks, has been prevailing in reinforcement learning in the last several years, in games, robotics, natural language processing, etc. We have been witnessing breakthroughs, like deep Q-network (DQN) (Mnih et al., 2015), AlphaGo (Silver et al., 2016a; 2017), and DeepStack (Moravčík et al., 2017); each of them represents a big family of problems and large number of applications. DQN (Mnih et al., 2015) is for single player games, and single agent control in general. DQN has implications for most algorithms and applications in RL. DQN ignites this round of popularity of deep reinforcement learning. AlphaGo (Silver et al., 2016a; 2017) is for two player perfect information zero-sum games. AlphaGo makes a phenomenal achievement on a very hard problem, and sets a landmark in AI. The success of AlphaGo influences similar games directly, and Alpha Zero (Silver et al., 2017) has already achieved significant successes on chess and Shogi. The techniques underlying AlphaGo (Silver et al., 2016a) and AlphaGo Zero (Silver et al., 2017), namely, deep learning, reinforcement learning, Monte Carlo tree search (MCTS), and self-play, will have wider and further implications and applications. As recommended by AlphaGo authors in their papers (Silver et al., 2016a; 2017), the following applications are worth further investigation: general game-playing (in particular, video games), classical planning, partially observed planning, scheduling, constraint satisfaction, robotics, industrial control, online recommendation systems, protein folding, reducing energy consumption, and searching for revolutionary new materials. DeepStack (Moravčík et al., 2017) is for two player imperfect information zero-sum games, a family of problems with inherent hardness to solve. DeepStack, similar to AlphaGo, also makes an extraordinary achievement on a hard problem, and sets a milestone in AI. It will have rich implications and wide applications, e.g., in defending strategic resources and robust decision making for medical treatment recommendations (Moravčík et al., 2017).

We also see novel algorithms, architectures, and applications, like asynchronous methods (Mnih et al., 2016), trust region methods (Schulman et al., 2015; Schulman et al., 2017b; Nachum et al., 2018), deterministic policy gradient (Silver et al., 2014; Lillicrap et al., 2016), combining policy gradient with off-policy RL (Nachum et al., 2017; 2018; Haarnoja et al., 2018), interpolated policy gradient (Gu et al., 2017), unsupervised reinforcement and auxiliary learning (Jaderberg et al., 2017; Mirowski et al., 2017), hindsight experience replay (Andrychowicz et al., 2017), differentiable neural computer (Graves et al., 2016), neural architecture design (Zoph and Le, 2017), guided policy search (Levine et al., 2016), generative adversarial imitation learning (Ho and Ermon, 2016), multi-agent games (Jaderberg et al., 2018), hard exploration Atari games (Aytar et al., 2018), StarCraft II Sun et al. (2018); Pang et al. (2018), chemical syntheses planning (Segler et al., 2018), character animation (Peng et al., 2018a), dexterous robots (OpenAI, 2018), OpenAI Five for Dota 2, etc.

Creativity would push the frontiers of deep RL further w.r.t. core elements, important mechanisms, and applications, seemingly without a boundary. RL probably helps, if a problem can be regarded as or transformed into a sequential decision making problem.

Why has deep learning been helping reinforcement learning make so many and so enormous achievements? Representation learning with deep learning enables automatic feature engineering and end-to-end learning through gradient descent, so that reliance on domain knowledge is significantly

[2]We choose to abbreviate deep reinforcement learning as "deep RL", since it is a branch of reinforcement learning, in particular, using deep neural networks in reinforcement learning, and "RL" is a well established abbreviation for reinforcement learning.

reduced or even removed for some problems. Feature engineering used to be done manually and is usually time-consuming, over-specified, and incomplete. Deep distributed representations exploit the hierarchical composition of factors in data to combat the exponential challenges of the curse of dimensionality. Generality, expressiveness and flexibility of deep neural networks make some tasks easier or possible, e.g., in the breakthroughs, and novel architectures, algorithms, and applications discussed above.

Deep learning, as a specific class of machine learning, is not without limitations, e.g., as a black-box lacking interpretability, as an "alchemy" without clear and sufficient scientific principles to work with, with difficulties in tuning hyperparameters, and without human intelligence so not being able to compete with a baby in some tasks. Deep reinforcement learning exacerbates these issues, and even reproducibility is a problem (Henderson et al., 2018). However, we see a bright future, since there are lots of work to improve deep learning, machine learning, reinforcement learning, deep reinforcement learning, and AI in general.

Deep learning and reinforcement learning, being selected as one of the MIT Technology Review 10 Breakthrough Technologies in 2013 and 2017 respectively, will play their crucial roles in achieving artificial general intelligence. David Silver, the major contributor of AlphaGo (Silver et al., 2016a; 2017), proposes a conjecture: artificial intelligence = reinforcement learning + deep learning (Silver, 2016). We will further discuss this conjecture in Chapter 21.

There are several frequently asked questions about deep reinforcement learning as below. We briefly discuss them in the above, and will further elucidate them in the coming chapters.

- Why deep?
- What is the state of the art?
- What are the issues, and potential solutions?

The book outline follows. First we discuss background of artificial intelligence, machine learning, deep learning, and reinforcement learning, as well as benchmarks and resources, in Chapter 2. Next we discuss RL core elements, including value function in Chapter 3, policy in Chapter 4, reward in Chapter 5, model in Chapter 6, exploration vs exploitation in Chapter 7, and representation in Chapter 8. Then we discuss important mechanisms for RL, including attention and memory in Chapter 9, unsupervised learning in Chapter 10, hierarchical RL in Chapter 11, multi-agent RL in Chapter 12, relational RL in Chapter 13, and, learning to learn in Chapter 14. After that, we discuss various RL applications, including games in Chapter 15, robotics in Chapter 16, natural language processing (NLP) in Chapter 17, computer vision in Chapter 18, finance and business management in Section 19, and more applications in Chapter 20, including, healthcare in Section 20.1, education in Section 20.2, energy in Section 20.3, transportation in Section 20.4, computer systems in Section 20.5, and, science, engineering and arts in Section 20.6. We close in Chapter 21, with a brief summary, discussions about challenges and opportunities, and an epilogue.

Figure 1 illustrates the manuscript outline. The agent-environment interaction sits in the center, around which are core elements, next important mechanisms, then various applications.

Main readers of this manuscript are those who want to get more familiar with deep reinforcement learning, in particular, novices to (deep) reinforcement learning. For reinforcement learning experts, as well as new comers, this book are helpful as a reference. This manuscript is helpful for deep reinforcement learning courses, with selected topics and papers.

We endeavour to discuss recent, representative work, and provide as much relevant information as possible, in an intuitive, high level, conceptual approach. We attempt to make this manuscript complementary to Sutton and Barto (2018), the RL classic focusing mostly on fundamentals in RL. However, deep RL is by nature an advanced topic; and most part of this manuscript is about introducing papers, mostly without covering detailed background. (Otherwise, the manuscript length would explode.) Consequently, most parts of this manuscript would be rather technical, somewhat rough, without full details. Original papers are usually the best resources for deep understanding.

Deep reinforcement learning is growing very fast. We post blogs to complement this manuscript, and endeavour to track the development of this field , at `https://medium.com/@yuxili`.

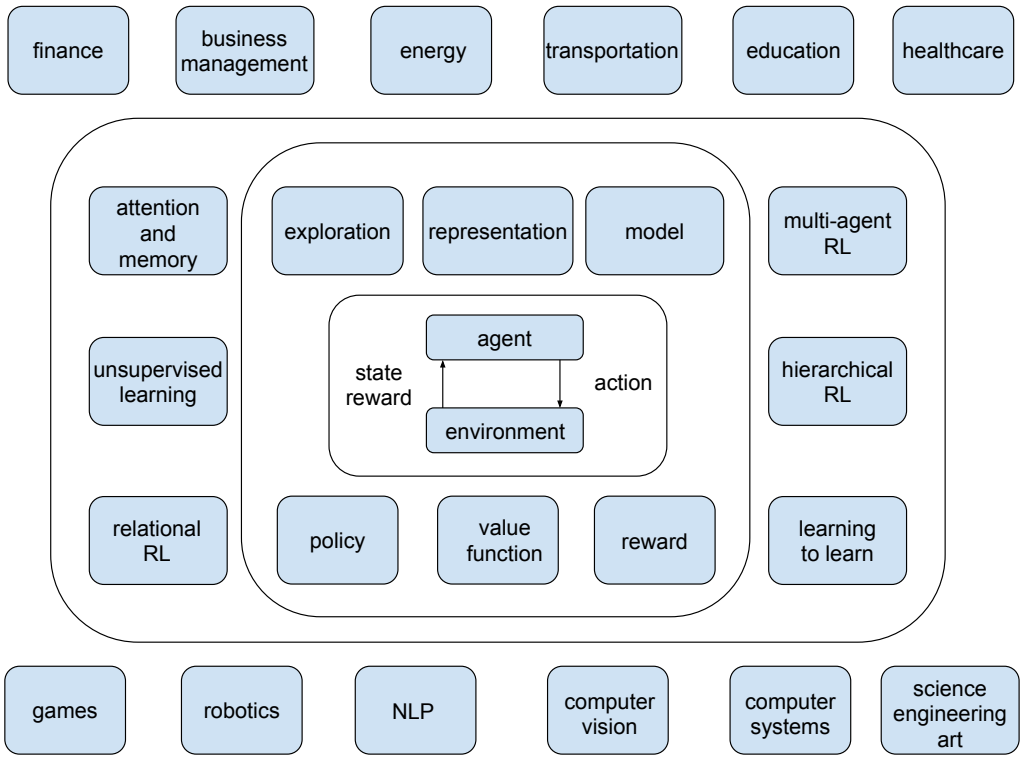

Figure 1: Outline

This manuscript covers a wide spectrum of topics in deep reinforcement learning. Although we have tried our best for excellence, there are inevitably shortcomings or even mistakes. Comments and criticisms are welcome.

2 BACKGROUND

In this chapter, we briefly introduce concepts and fundamentals in artificial intelligence (Russell and Norvig, 2009), machine learning, deep learning (Goodfellow et al., 2016), and, reinforcement learning (Sutton and Barto, 2018).

We do not give detailed background introduction for artificial intelligence, machine learning and deep learning; these are too broad to discuss in details here. Instead, we recommend the following recent Nature/Science survey papers: Jordan and Mitchell (2015) for machine learning, and LeCun et al. (2015) for deep learning. For reinforcement learning, we cover some basics as a mini tutorial, and recommend the textbook, Sutton and Barto (2018), two courses, RL course by David Silver at UCL (Silver, 2015) and Deep RL course by Sergey Levin at UC Berkeley (Levine, 2018), and a recent Nature survey paper (Littman, 2015). We present some resources for deep RL in Section 2.5.

In Figure 2, we illustrate relationship among several concepts in AI and machine learning. Deep reinforcement learning, as the name indicates, is at the intersection of deep learning and reinforcement learning. We usually categorize machine learning as supervised learning, unsupervised learning, and reinforcement learning. Deep learning can work with/as supervised learning, unsupervised learning, reinforcement learning, and other machine learning approaches. [3] Deep learning is part of machine learning, which is part of AI. Note that all these fields are evolving, e.g., deep learning and deep reinforcement learning are addressing classical AI problems, like logic, reasoning, and knowledge representation. Reinforcement learning can be important for all AI problems, as quoted from Russell and Norvig (2009), "reinforcement learning might be considered to encompass all of AI: an agent is placed in an environment and must learn to behave successfully therein", and, "reinforcement learning can be viewed as a microcosm for the entire AI problem".

2.1 ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) is a very broad area. Even an authoritative AI textbook like Russell and Norvig (2009) does not give a precise definition. Russell and Norvig (2009) discuss definitions of AI from four perspectives in two comparisons: 1) thought process and reasoning vs. behaviour, and, 2) success in terms of fidelity to human performance vs. rationality, an ideal performance measure. We follow the discussions of Russell and Norvig (2009), list four ways to define AI, quoting directly from Russell and Norvig (2009).

Definition 1, "acting humanly", follows the Turing Test approach. "The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil, 1992) "The study of how to make computers do things at which, at the moment, people are better." (Rich and Knight, 1991)

A computer passes a Turing Test, if a human interrogator can not tell if the written responses to some written question are from a computer or a human. The computer needs the following capabilities: *natural language processing* (NLP) for successful communication in English, *knowledge representation* for storage of what it knows or hears, *automated reasoning* for answering questions and drawing new conclusions from the stored information, and, *machine learning* for adaptation to new scenarios and detection and extrapolation of patterns. In a total Turing Test, video signals are involved, so that a computer needs more capabilities, *computer vision* for object perception, and, *robotics* for object manipulation and motion control. AI researchers have been devoting most efforts to the underlying principles of intelligence, mostly covered by the above six disciplines, and less on passing the Turing Test, since, duplicating an exemplar is usually not the goal, e.g., an airplane may not simply imitate a bird.

Definition 2, "thinking humanly", follows the cognitive modeling approach. "The exciting new effort to make computers think ... machines with minds, in the full and literal sense." (Haugeland,

[3] Machine learning includes many approaches: decision tree learning, association rule learning, artificial neural networks, inductive logic programming, support vector machines, clustering, Bayesian networks, reinforcement learning, representation learning, similarity and metric learning, sparse dictionary learning, genetic algorithms, and, rule-based machine learning, according to Wikipedia, `https://en.wikipedia.org/wiki/Machine_learning`. Also check textbooks like Russell and Norvig (2009), Mitchell (1997), and Zhou (2016).

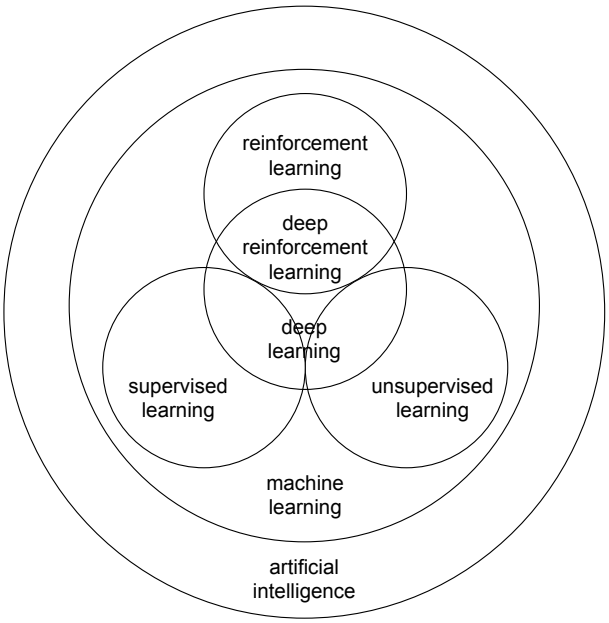

Figure 2: Relationship among deep reinforcement learning, deep learning, reinforcement learning, supervised learning, unsupervised learning, machine learning, and, artificial intelligence. Deep learning and deep reinforcement learning are addressing many classical AI problems.

1989) "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning ..." (Bellman, 1978)

Definition 3, "thinking rationally", follows the "law of thought" approach. "The study of mental faculties through the use of computational models." (Charniak and McDermott, 1985) "The study of the computation that make it possible to perceive, reason, and act." (Winston, 1992)

Definition 4, "acting rationally", follows the rational agent approach. "Computational Intelligence is the study of the design of intelligent agents." (Poole et al., 1998) "AI ... is concerned with intelligent behavior in artifacts." (Nilsson, 1998) The rational agent approach is more amenable to scientific development than those based on human behavior or human thought. Russell and Norvig (2009) thus focuse on general principles of rational agents and their building components.

We list the foundations and the questions they attempt to answer as in Russell and Norvig (2009): *philosophy*, for questions like, "Can formal rules be used to draw valid conclusions?", "How does the mind arise from a physical brain?", "Where does knowledge come from?", and, "How does knowledge lead to action?"; *mathematics*, for questions like, "What are the formal rules to draw valid conclusions?", "What can be computed?", and, "How do we reason with uncertain information?"; *economics*, for questions like, "How should we make decisions so as to maximize payoff?", "How should we do this when others may not go along?", and, "How should we do this when the payoff may be far in the future?"; *neuroscience*, for questions like, "How do brains process information?"; *psychology*, for questions like, "How do humans and animals think and act?"; *computer engineering*, for questions like, "How can we build an efficient computer?"; *control theory and cybernetics*, for questions like, "How can artifacts operate under their own control?"; and, *linguistics*, for questions like, "How does language relate to thought?"

Russell and Norvig (2009) present the history of AI as, the gestation of artificial intelligence (1943-1955), the birth of artificial intelligence (1956), early enthusiasm, great expectations (1952 - 1969),

a dose of reality (1966 - 1973), knowledge-based systems: the key to power? (1969 - 1979), AI becomes an industry (1980 - present), the return of neural networks (1986 - present), AI adopts the scientific method (1987 - present), the emergence of intelligent agents (1995 - present), and, the availability of very large data sets (2001 -present).

2.2 MACHINE LEARNING

Machine learning is about learning from data and making predictions and/or decisions. Quoting from Mitchell (1997), "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E."

Usually we categorize machine learning as supervised, unsupervised, and reinforcement learning. In supervised learning, there are labeled data; in unsupervised learning, there are no labeled data. Classification and regression are two types of supervised learning problems, with categorical and numerical outputs respectively.

Unsupervised learning attempts to extract information from data without labels, e.g., clustering and density estimation. Representation learning is a classical type of unsupervised learning. However, training deep neural networks with supervised learning is a kind of representation learning. Representation learning finds a representation to preserve as much information about the original data as possible, and, at the same time, to keep the representation simpler or more accessible than the original data, with low-dimensional, sparse, and independent representations.

Deep learning, or deep neural networks, is a particular machine learning scheme, usually for supervised or unsupervised learning, and can be integrated with reinforcement learning, for state representation and/or function approximator. Supervised and unsupervised learning are usually one-shot, myopic, considering instant rewards; while reinforcement learning is sequential, far-sighted, considering long-term accumulative rewards.

Reinforcement learning is usually about sequential decision making. In reinforcement learning, in contrast to supervised learning and unsupervised learning, there are evaluative feedbacks, but no supervised labels. Comparing with supervised learning, reinforcement learning has additional challenges like credit assignment, stability, and, exploration. Reinforcement learning is kin to optimal control (Bertsekas, 2012; Sutton et al., 1992), and operations research and management (Powell, 2011), and is also related to psychology and neuroscience (Sutton and Barto, 2018).

Machine learning is the basis for big data, data science (Blei and Smyth, 2017; Provost and Fawcett, 2013), predictive modeling (Kuhn and Johnson, 2013), data mining (Han et al., 2011), information retrieval (Manning et al., 2008), etc, and becomes a critical ingredient for computer vision, natural language processing, robotics, etc. Probability theory and statistics (Hastie et al., 2009; Murphy, 2012; Vapnik, 1998) and optimization (Boyd and Vandenberghe, 2004; Bottou et al., 2018) are important for statistical machine learning. Machine learning is a subset of AI; however, it is evolving to be critical for all fields of AI.

A machine learning algorithm is composed of a dataset, a cost/loss function, an optimization procedure, and a model (Goodfellow et al., 2016). A dataset is divided into non-overlapping training, validation, and testing subsets. A cost/loss function measures the model performance, e.g., with respect to accuracy, like mean square error in regression and classification error rate.

The concept of entropy is important for the definition of loss functions. For a random variable X with distribution p, the entropy is a measure of its uncertainty, denoted by $\mathbb{H}(X)$ or $\mathbb{H}(p)$,

$$\mathbb{H}(X) \triangleq -\sum_{k=1}^{K} p(X=k) \log_2 p(X=k). \tag{1}$$

For binary random variables, $X \in \{0, 1\}$, we have $p(X=1) = \theta$ and $p(X=0) = 1-\theta$,

$$\mathbb{H}(X) = -[\theta \log_2 \theta + (1-\theta) \log_2 (1-\theta)]. \tag{2}$$

Kullback-Leibler divergence (KL-divergence) or relative entropy, is one way to measure the dissimilarity of two probability distributions, p and q,

$$\mathbb{KL}(p\|q) \triangleq \sum_{k=1}^{K} p_k \log \frac{p_k}{q_k} \triangleq \sum_{k=1}^{K} p_k \log p_k - p_k \log q_k = -\mathbb{H}(p) + \mathbb{H}(p,q). \tag{3}$$

Here $\mathbb{H}(p,q)$ is the cross entropy. We have $\mathbb{KL}(p,q) \triangleq -\sum_{k=1}^{K} p_k \log q_k$. See Murphy (2012).

Training error measures the error on the training data, minimizing which is an optimization problem. Generalization error, or test error, measures the error on new input data, which differentiates machine learning from optimization. A machine learning algorithm tries to make the training error, and the gap between training error and testing error small. A model is under-fitting if it can not achieve a low training error; a model is over-fitting if the gap between training error and test error is large.

A model's capacity measures the range of functions it can fit. Ockham's razor[4] states that, with the same expressiveness, simple models are preferred. Training error and generalization error vs. model capacity usually form a U-shape relationship. We find the optimal capacity to achieve low training error and small gap between training error and generalization error. Bias measures the expected deviation of the estimator from the true value; while variance measures the deviation of the estimator from the expected value, or variance of the estimator. As model capacity increases, bias tends to decrease, while variance tends to increase, yielding another U-shape relationship between generalization error vs. model capacity. We try to find the optimal capacity point, of which under-fitting occurs on the left and over-fitting occurs on the right. Regularization adds a penalty term to the cost function, to reduce the generalization error, but not training error. No free lunch theorem states that there is no universally best model, or best regularizer. An implication is that deep learning may not be the best model for some problems. There are model parameters, and hyperparameters for model capacity and regularization. Cross-validation is used to tune hyperparameters, to strike a balance between bias and variance, and to select the optimal model.

Maximum likelihood estimation (MLE) is a common approach to derive good estimation of parameters. For issues like numerical underflow, the product in MLE is converted to summation to obtain negative log-likelihood (NLL). MLE is equivalent to minimizing KL divergence, the dissimilarity between the empirical distribution defined by the training data and the model distribution. Minimizing KL divergence between two distributions corresponds to minimizing the cross-entropy between the distributions. In short, maximization of likelihood becomes minimization of the negative log-likelihood (NLL), or equivalently, minimization of cross entropy.

Gradient descent is a common way to solve optimization problems. Stochastic gradient descent extends gradient descent by working with a single sample each time, and usually with minibatches.

Importance sampling is a technique to estimate properties of a particular distribution, by samples from a different distribution, to lower the variance of the estimation, or when sampling from the distribution of interest is difficult.

There are mathematical analysis frameworks for machine learning algorithms. Kolmogorov complexity (Li and Vitányi, 2008), or algorithmic complexity, studies the notion of simplicity in Ockham's razor. In probably approximately correct (PAC) learning (Valiant, 1984), a learning algorithm aims to select a generalization function, to achieve low generalization error with high probability. VC dimension (Vapnik, 1998) measures the capacity of a binary classifier.

2.3 Deep Learning

Deep learning is in contrast to "shallow" learning. For many machine learning algorithms, e.g., linear regression, logistic regression, support vector machines (SVMs), decision trees, and boosting, we have input layer and output layer, and the inputs may be transformed with manual feature engineering before training. In deep learning, between input and output layers, we have one or more hidden layers. At each layer except the input layer, we compute the input to each unit, as the weighted sum of units from the previous layer; then we usually use nonlinear transformation, or activation function, such as logistic, tanh, or more popularly recently, rectified linear unit (ReLU), to apply to the input of a unit, to obtain a new representation of the input from the previous layer. We

[4] "Occam's razor" is a popular misspelling (Russell and Norvig, 2009).

have weights on links between units from layer to layer. After computations flow forward from input to output, at the output layer and each hidden layer, we can compute error derivatives backward, and backpropagate gradients towards the input layer, so that weights can be updated to optimize some loss function.

A feedforward deep neural network or multilayer perceptron (MLP) is to map a set of input values to output values with a mathematical function formed by composing many simpler functions at each layer. A convolutional neural network (CNN) is a feedforward deep neural network, with convolutional layers, pooling layers, and, fully connected layers. CNNs are designed to process data with multiple arrays, e.g., colour image, language, audio spectrogram, and video, benefiting from the properties of such signals: local connections, shared weights, pooling, and, the use of many layers, and are inspired by simple cells and complex cells in visual neuroscience (LeCun et al., 2015). ResNets (He et al., 2016d) are designed to ease the training of very deep neural networks by adding shortcut connections to learn residual functions with reference to the layer inputs. A recurrent neural network (RNN) is often used to process sequential inputs like speech and language, element by element, with hidden units to store history of past elements. A RNN can be seen as a multilayer neural network with all layers sharing the same weights, when being unfolded in time of forward computation. It is hard for RNN to store information for very long time and the gradient may vanish. Long short term memory networks (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent unit (GRU) (Chung et al., 2014) are proposed to address such issues, with gating mechanisms to manipulate information through recurrent cells. Gradient backpropagation or its variants can be used for training all deep neural networks mentioned above.

Dropout (Srivastava et al., 2014) is a regularization strategy to train an ensemble of sub-networks by removing non-output units randomly from the original network. Batch normalization (Ioffe and Szegedy, 2015; Santurkar et al., 2018) and layer normalization (Ba et al., 2016) are designed to improve training efficiency.

Deep neural networks learn representations automatically from raw inputs to recover the compositional hierarchies in many natural signals, i.e., higher-level features are composed of lower-level ones, e.g., in images, the hierarch of objects, parts, motifs, and local combinations of edges. Distributed representation is a central idea in deep learning, which implies that many features may represent each input, and each feature may represent many inputs. The exponential advantages of deep, distributed representations combat the exponential challenges of the curse of dimensionality. The notion of end-to-end training refers to that a learning model uses raw inputs, usually without manual feature engineering, to generate outputs, e.g., AlexNet (Krizhevsky et al., 2012) with raw pixels for image classification, Graves et al. (2013) with a Fourier transformation of audio data for speech recognition; Seq2Seq (Sutskever et al., 2014) with raw sentences for machine translation, and DQN (Mnih et al., 2015) with raw pixels and score to play games.

There are efforts to design new neural network architectures, like capsules (Sabour et al., 2017; Hinton et al., 2018). There are also efforts to design deep machine learning architectures without neural networks, like DeepForest (Zhou and Feng, 2017; Feng and Zhou, 2017).

2.4 Reinforcement Learning

We provide background of reinforcement learning briefly in this section, as a mini tutorial. It is essential to have a good understanding of reinforcement learning, for a good understanding of deep reinforcement learning. Deep RL is a particular type of RL, with deep neural networks for state representation and/or function approximation for value function, policy, transition model, or reward function. Silver (2015) is a clear introductory material for reinforcement learning.

Sutton and Barto (2018) present Bibliographical and Historical Remarks at the end of each chapter. Russell and Norvig (2009) present Bibliographical and Historical Notes in Chapter 21 Reinforcement Learning. See Barto (2018) for a talk about a brief history of reinforcement learning.

We first explain some terms in RL parlance. These terms would become clearer after reading the rest of this chapter. We put them collectively here to make it convenient for readers to check them.

The prediction problem, or policy evaluation, is to compute the state or action value function for a policy. The control problem is to find the optimal policy. Planning constructs a value function or a policy with a model.

On-policy methods evaluate or improve the behaviour policy, e.g., SARSA fits the action-value function to the current policy, i.e., SARSA evaluates the policy based on samples from the same policy, then refines the policy greedily with respect to action values. In off-policy methods, an agent learns an optimal value function/policy, maybe following an unrelated behaviour policy. For instance, Q-learning attempts to find action values for the optimal policy directly, not necessarily fitting to the policy generating the data, i.e., the policy Q-learning obtains is usually different from the policy that generates the samples. The notion of on-policy and off-policy can be understood as same-policy and different-policy.

The exploration vs exploitation dilemma is about the agent needs to exploit the currently best action to maximize rewards greedily, yet it has to explore the environment to find better actions, when the policy is not optimal yet, or the system is non-stationary.

In model-free methods, the agent learns with trail-and-error from experience directly; the model, i.e., for state transition and reward, is not known. RL methods that use models are model-based methods; the model may be given, e.g. in the game of computer Go, or learned from experience.

In an online mode, training algorithms are executed on data acquired in sequence. In an offline mode, or a batch mode, models are trained on the entire data set.

With bootstrapping, an estimate of state or action value is updated from subsequent estimates.

2.4.1 Problem Setup

An RL agent interacts with an environment over time. At each time step t, the agent receives a state s_t in a state space $\mathcal{S}$, and selects an action a_t from an action space $\mathcal{A}$, following a policy $\pi(a_t|s_t)$, which is the agent's behavior, i.e., a mapping from state s_t to actions a_t. The agent receives a scalar reward r_t, and transitions to the next state s_{t+1}, according to the environment dynamics, or model, for reward function $\mathcal{R}(s, a)$, and, state transition probability $\mathcal{P}(s_{t+1}|s_t, a_t)$, respectively. In an episodic problem, this process continues until the agent reaches a terminal state and then it restarts. The return is the discounted, accumulated reward with the discount factor $\gamma \in (0, 1]$,

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}. \tag{4}$$

The agent aims to maximize the expectation of such long term return from each state. The problem is set up in discrete state and action spaces. It is not hard to extend it to continuous spaces. In partially observable environments, an agent can not observe states fully, but has observations.

When an RL problem satisfies the Markov property, i.e., the future depends only on the current state and action, but not on the past, it is formulated as a Markov decision process (MDP), defined by the 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. When the system model is available, we use dynamic programming methods: policy evaluation to calculate value/action value function for a policy, value iteration and policy iteration for finding an optimal policy. When there is no model, we resort to RL methods. RL methods also work when the model is available. An RL environment can be a multi-armed bandit, an MDP, a partially observable MDP (POMDP), a game, etc.

2.4.2 Value Function

A value function is a prediction of the expected, accumulative, discounted, future reward, measuring how good each state, or state-action pair, is. The state value,

$$v_\pi(s) = \mathbb{E}[R_t|s_t = s], \text{ where, } R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}, \tag{5}$$

is the expected return for following policy π from state s. The action value,

$$q_\pi(s, a) = \mathbb{E}[R_t|s_t = s, a_t = a], \tag{6}$$

is the expected return for selecting action a in state s and then following policy π. Value function $v_\pi(s)$ decomposes into the Bellman equation:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]. \tag{7}$$

An optimal state value,

$$v_*(s) = \max_\pi v_\pi(s) = \max_a q_{\pi^*}(s, a), \tag{8}$$

is the maximum state value achievable by any policy for state s, which decomposes into the Bellman equation:

$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')]. \tag{9}$$

Action value function $q_\pi(s, a)$ decomposes into the Bellman equation:

$$q_\pi(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma \sum_{a'} \pi(a'|s')q_\pi(s', a')]. \tag{10}$$

An optimal action value function,

$$q_*(s, a) = \max_\pi q_\pi(s, a), \tag{11}$$

is the maximum action value achievable by any policy for state s and action a, which decomposes into the Bellman equation:

$$q_*(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a'} q_*(s', a')]. \tag{12}$$

We denote an optimal policy by π^*.

Consider the shortest path problem as an example. In graph theory, the single-source shortest path problem is to find the shortest path between a pair of nodes so that the sum of weights of edges in the path is minimized. In RL, the state is the current node. At each node, following the link to each neighbour is an action. The transition model indicates that, after choosing a link to follow, the agent goes to a neighbour. The reward is then the negative of link weight/cost/distance. The discount factor can be $\gamma = 1$, since it is an episodic task. The goal is to find a path to maximize the negative of the total cost, i.e., to minimize the total distance. An optimal policy is to choose the best neighbour to traverse to achieve the shortest path; and, for each state/node, an optimal value is the shortest distance from that node to the destination. Dijkstra's algorithm is an efficient algorithm, with the information of the graph, including nodes, edges, and weights. RL can work in a model-free approach, by wandering in the graph according to some policy, without such global graph information. RL algorithms are more general than Dijkstra's algorithm, although with global graph information, Dijkstra's algorithm is very efficient.

2.4.3 Exploration vs. Exploitation

An RL agent needs to trade off between exploration of uncertain policies and exploitation of the current best policy, a fundamental dilemma in RL. Here we introduce a simple approach, ϵ-greedy, where $\epsilon \in (0, 1)$, usually a small value close to 0. In ϵ-greedy, an agent selects a greedy action $a = \arg\max_{a \in \mathcal{A}} Q(s, a)$, for the current state s, with probability $1 - \epsilon$, and, selects a random action with probability ϵ. That is, the agent exploits the current value function estimation with probability $1 - \epsilon$, and explores with probability ϵ.

We will discuss more about the exploration vs exploitation dilemma in Chapter 7, including several principles: naive methods such as ϵ-greedy, optimistic initialisation, upper confidence bounds, probability matching, and, information state search, which are developed in the settings of multi-armed bandit, but are applicable to RL problems (Silver, 2015).

2.4.4 Dynamic Programming

Dynamic programming (DP) is a general method for problems with optimal substructure and overlapping subproblems. MDPs satisfy these properties, where, Bellman equation gives recursive decomposition, and, value function stores and reuses sub-solutions (Silver, 2015). DP assumes full knowledge of the transition and reward models of the MDP. The prediction problem is to evaluate the value function for a given policy, and the control problem is to find an optimal value function and/or an optimal policy.

Iterative policy evaluation is an approach to evaluate a given policy π. It iteratively applies Bellman expectation backup,

$$v_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a|s)[\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) v_k(s')], \tag{13}$$

so that at each iteration $k+1$, for all states $s \in \mathcal{S}$, update $v_{k+1}(s)$ from value functions of its successor states $v_k(s')$. The value function will converge to v_π, the value function of the policy π.

Policy iteration (PI) alternates between policy evaluation and policy improvement, to generate a sequence of improving policies. In policy evaluation, the value function of the current policy is estimated to obtain v_π. In policy improvement, the current value function is used to generate a better policy, e.g., by selecting actions greedily with respect to the value function v_π. This policy iteration process of iterative policy evaluation and greedy policy improvement will converge to an optimal policy and value function.

We may modify the policy iteration step, stopping it before convergence. A generalized policy iteration (GPI) is composed of any policy evaluation method and any policy improvement method.

Value iteration (VI) finds an optimal policy. It iteratively applies Bellman optimality backup,

$$v_*(s) = \max_a \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) v_k(s'). \tag{14}$$

At each iteration $k+1$, it updates $v_{k+1}(s)$ from $v_k(s')$, for all states $s \in \mathcal{S}$. Such synchronous backup will converge to the value function of an optimal policy. We may have asynchronous DP, and approximate DP.

We have Bellman equation for value function in Equation (7). Bellman operator is defined as,

$$(T^\pi v)(s) \doteq \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]. \tag{15}$$

TD fix point is then,

$$v_\pi = T^\pi v_\pi. \tag{16}$$

We can define the Bellman expectation backup operator, in matrix forms,

$$T^\pi(v) = \mathcal{R} + \gamma \mathcal{P}^\pi v, \tag{17}$$

and the Bellman optimality backup operator,

$$T^*(v) = \max_{a \in \mathcal{A}} \mathcal{R}^a + \gamma \mathcal{P}^a v, \tag{18}$$

We can show that these operators are contractions with fixed points, respectively, which help prove the convergence of policy iteration, value iteration, and certain general policy iteration algorithms. See Silver (2015), Sutton and Barto (2018), and Bertsekas and Tsitsiklis (1996) for more details.

2.4.5 MONTE CARLO

Monte Carlo methods learn from complete episodes of experience, not assuming knowledge of transition nor reward models, and use sample means for estimation. Monte Carlo methods are applicable only to episodic tasks.

With Monte Carlo methods for policy evaluation, we use empirical mean return rather than expected return for the evaluation. By law of large numbers, the estimated value function converges to the value function of the policy.

On-policy Monte Carlo control follows a generalized policy iteration scheme. For policy evaluation, it uses Monte Carlo policy evaluation for the action value. For policy improvement, it uses ϵ-greedy policy improvement. It can be shown that Greedy in the limit with infinite exploration (GLIE) Monte-Carlo control converges to the optimal action-value function (Singh et al., 2000).

In off-policy learning, we evaluate a target policy, following a behaviour policy. With off-policy, we can learn with observations from humans or other agents, reuse experience from old policies,

learn an optimal policy while following an exploratory policy, and, learn multiple policies based on experience of one policy. (Silver, 2015)

We can use importance sampling for off-policy Monte Carlo methods, by multiply importance sampling correction weights along the whole episode, to evaluate the target policy with experience generated by the behaviour policy. This may increase variance dramatically though.

2.4.6 TEMPORAL DIFFERENCE LEARNING

Temporal difference (TD) learning is central in RL. TD learning usually refers to the learning methods for value function evaluation in Sutton (1988). Q-learning (Watkins and Dayan, 1992) and SARSA (Rummery and Niranjan, 1994) are temporal difference control methods.

TD learning (Sutton, 1988) learns value function $V(s)$ directly from experience with TD error, with bootstrapping, in a model-free, online, and fully incremental way. TD learning is a prediction problem. The update rule is,

$$V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)], \tag{19}$$

where α is a learning rate, and $r + \gamma V(s') - V(s)$ is called the TD error. Algorithm 1 presents the pseudo code for tabular TD learning. Precisely, it is tabular TD(0) learning, where "0" indicates it is based on one-step returns.

Bootstrapping estimates state or action value function based on subsequent estimates as in the TD update rule, and is common in RL, like in TD learning, Q-learning, and SARSA. Bootstrapping methods are usually faster to learn, and enable learning to be online and continual. Bootstrapping methods are not instances of true gradient decent, since the target depends on the values to be estimated. The concept of semi-gradient descent is then introduced (Sutton and Barto, 2018).

Input: the policy π to be evaluated
Output: value function V

initialize V arbitrarily, e.g., to 0 for all states
for *each episode* **do**
    initialize state s
    for *each step of episode, state s is not terminal* **do**
        $a \leftarrow$ action given by π for s
        take action a, observe r, s'
        $V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)]$
        $s \leftarrow s'$
    end
end

Algorithm 1: TD learning, adapted from Sutton and Barto (2018)

Output: action value function Q

initialize Q arbitrarily, e.g., to 0 for all states, set action value for terminal states as 0
for *each episode* **do**
    initialize state s
    for *each step of episode, state s is not terminal* **do**
        $a \leftarrow$ action for s derived by Q, e.g., ϵ-greedy
        take action a, observe r, s'
        $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
        $s \leftarrow s'$
    end
end

Algorithm 3: Q-learning, adapted from Sutton and Barto (2018)

Output: action value function Q

initialize Q arbitrarily, e.g., to 0 for all states, set action value for terminal states as 0
for *each episode* **do**
    initialize state s
    for *each step of episode, state s is not terminal* **do**
        $a \leftarrow$ action for s derived by Q, e.g., ϵ-greedy
        take action a, observe r, s'
        $a' \leftarrow$ action for s' derived by Q, e.g., ϵ-greedy
        $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$
        $s \leftarrow s'$, $a \leftarrow a'$
    end
end

Algorithm 2: SARSA, adapted from Sutton and Barto (2018)

SARSA, representing state, action, reward, (next) state, (next) action, is an on-policy control method to find an optimal policy, with the update rule,

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]. \tag{20}$$

Algorithm 2 presents the pseudo code for tabular SARSA, i.e., tabular SARSA(0).

Q-learning is an off-policy control method to find the optimal policy. Q-learning learns the action value function, with the update rule,

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]. \tag{21}$$

Q-learning refines the policy greedily w.r.t. action values by the max operator. Algorithm 3 presents the pseudo code for Q-learning, precisely, tabular Q(0) learning.

TD-learning, Q-learning and SARSA converge under certain conditions. From an optimal action value function, we can derive an optimal policy.

2.4.7 Multi-step Bootstrapping

The above algorithms are referred to as TD(0), Q(0) and SARSA(0), learning with one-step return. We have variants with multi-step return for them in the forward view. In n-step update, $V(s_t)$ is updated toward the n-step return, defined as,

$$r_t + \gamma r_{t+1} + \cdots + \gamma^{n-1} r_{t+n-1} + \gamma^n V(s_{t+n}). \tag{22}$$

The eligibility trace from the backward view provides an online, incremental implementation, resulting in TD(λ), Q(λ) and SARSA(λ) algorithms, where $\lambda \in [0,1]$. TD(1) is the same as the Monte Carlo approach. Eligibility trace is a short-term memory, usually lasting within an episode, assists the learning process, by affecting the weight vector. The weight vector is a long-term memory, lasting the whole duration of the system, determines the estimated value. Eligibility trace helps with the issues of long-delayed rewards and non-Markov tasks (Sutton and Barto, 2018).

TD(λ) unifies one-step TD prediction, TD(0), with Monte Carlo methods, TD(1), using eligibility traces and the decay parameter λ, for prediction algorithms. De Asis et al. (2018) make unification for multi-step TD control algorithms.

Comparison of DP, MC, and TD

In the following, we compare dynamic programming (DP), Monte Carlo (MC), and temporal difference (TD) learning, based on Silver (2015) and Sutton and Barto (2018).

DP requires the model; TD and MC are model-free. DP and TD bootstrap; MC does not. TD and MC work with sample backups, DP and exhaustive search work with full backups. DP works with one step backups; TD works with one or multi-step backups; MC and exhaustive search work with deep backups, until the episode terminates.

TD can learn online, from incomplete sequences, in continuous environments. MC learns from complete sequences, in episodic environments. TD has low variance, some bias, usually more efficient than MC, more sensitive to initialization. TD(0) converges to value function of a policy, may diverge with function approximation. MC has high variance, zero bias, simple to understand and use, insensitive to initilization. MC has good convergence properties, even with function approximation.

TD exploits Markov property, so it is more efficient in Markov environments. MC does not assume Markov property, so it is usually more effective in non-Markov environments.

Table 1 compare DP with TD (Silver, 2015; Sutton and Barto, 2018).

Full Backup (DP)	Sample Backup
iterative policy evaluation $v(s) \leftarrow \mathbb{E}[r + \gamma v(s')\|s]$	TD learning $v(s) \overset{\alpha}{\leftarrow} r + \gamma v(s')$
Q policy iteration $Q(s,a) \leftarrow \mathbb{E}[r + \gamma Q(s',a')\|s,a]$	SARSA $Q(s,a) \overset{\alpha}{\leftarrow} r + \gamma Q(s',a')$
Q value iteration $Q(s,a) \leftarrow \mathbb{E}[r + \gamma \max_{a' \in \mathcal{A}} Q(s',a')\|s,a]$	Q-learning $Q(s,a) \overset{\alpha}{\leftarrow} r + \gamma \max_{a' \in \mathcal{A}} Q(s',a')$

Table 1: Comparison between DP and TD Learning, where $x \overset{\alpha}{\leftarrow} y \doteq x \leftarrow \alpha(y - x)$.

2.4.8 MODEL-BASED RL

Sutton (1990) proposes Dyna-Q to integrate learning, acting, and planning, by not only learning from real experience, but also planning with simulated trajectories from a learned model. Learning uses real experience from the environment; and planning uses experience simulated by a model. Algorithm 4 presents the pseudo code for tabular Dyna-Q. We will discuss more about model-based RL in Chapter 6.

// $Model(s,a)$ denotes predicted reward and next state for state action pair (s,a)
initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$
for *true* **do**
  $s \leftarrow$ current, nonterminal, state
  $a \leftarrow$ action for s derived by Q, e.g., ϵ-greedy
  //acting
  take action a; observe reward r, and next state s'
  // direct reinforcement learning
  $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
  // model learning
  $Model(s,a) \leftarrow r, s'$
  // planning
  for N *iterations* **do**
    $s \leftarrow$ random state previously observed
    $a \leftarrow$ random action previously taken
    $r, s' \leftarrow Model(s,a)$
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
  end
end

Algorithm 4: Dyna-Q, adapted from Sutton and Barto (2018)

2.4.9 FUNCTION APPROXIMATION

We discuss tabular cases above, where a value function or a policy is stored in a tabular form. Function approximation is a way for generalization when the state and/or action spaces are large or continuous. Function approximation aims to generalize from examples of a function to construct an approximate of the entire function; it is usually a concept in supervised learning, studied in the

fields of machine learning, patten recognition, and statistical curve fitting; function approximation in reinforcement learning usually treats each backup as a training example, and encounters new issues like nonstationarity, bootstrapping, and delayed targets (Sutton and Barto, 2018). Linear function approximation is a popular choice, partially due to its desirable theoretical properties, esp. before the work of deep Q-network (Mnih et al., 2015). However, the integration of reinforcement learning and neural networks dates back a long time ago (Sutton and Barto, 2018; Bertsekas and Tsitsiklis, 1996; Schmidhuber, 2015).

Algorithm 5 presents the pseudo code for TD(0) with function approximation. $\hat{v}(s, \boldsymbol{w})$ is the approximate value function, $\boldsymbol{w}$ is the value function weight vector, $\nabla \hat{v}(s, \boldsymbol{w})$ is the gradient of the approximate value function w.r.t. the weight vector, which is updated following the update rule,

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha[r + \gamma \hat{v}(s', \boldsymbol{w}) - \hat{v}(s, \boldsymbol{w})] \nabla \hat{v}(s, \boldsymbol{w}). \tag{23}$$

Input: the policy π to be evaluated
Input: a differentiable value function $\hat{v}(s, \boldsymbol{w})$, $\hat{v}(terminal, \cdot) = 0$
Output: value function $\hat{v}(s, \boldsymbol{w})$

initialize value function weight $\boldsymbol{w}$ arbitrarily, e.g., $\boldsymbol{w} = 0$
for *each episode* **do**
    initialize state s
    for *each step of episode, state s is not terminal* **do**
        $a \leftarrow \pi(\cdot|s)$
        take action a, observe r, s'
        $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha[r + \gamma \hat{v}(s', \boldsymbol{w}) - \hat{v}(s, \boldsymbol{w})] \nabla \hat{v}(s, \boldsymbol{w})$
        $s \leftarrow s'$
    end
end

Algorithm 5: TD(0) with function approximation, adapted from Sutton and Barto (2018)

When combining off-policy, function approximation, and bootstrapping, instability and divergence may occur (Tsitsiklis and Van Roy, 1997), which is called the deadly triad issue (Sutton and Barto, 2018). All these three elements are necessary: function approximation for scalability and generalization, bootstrapping for computational and data efficiency, and off-policy learning for freeing behaviour policy from target policy. What is the root cause for the instability? Learning or sampling is not, since dynamic programming suffers from divergence with function approximation; exploration, greedification, or control is not, since prediction alone can diverge; local minima or complex non-linear function approximation is not, since linear function approximation can produce instability (Sutton, 2016). It is unclear what is the root cause for instability – each single factor mentioned above is not – there are still many open problems in off-policy learning (Sutton and Barto, 2018).

Table 2 presents various algorithms that tackle various issues (Sutton, 2016). ADP algorithms refer to approximate dynamic programming algorithms like policy evaluation, policy iteration, and value iteration, with function approximation. Least square temporal difference (LSTD) (Bradtke and Barto, 1996) computes TD fix-point directly in batch mode. LSTD is data efficient, yet with squared time complexity. LSPE (Nedić and Bertsekas, 2003) extends LSTD. Fitted-Q algorithms (Ernst et al., 2005; Riedmiller, 2005) learn action values in batch mode. Residual gradient algorithms (Baird, 1995) minimize Bellman error. Gradient-TD (Sutton et al., 2009a;b; Mahmood et al., 2014) methods are true gradient algorithms, perform SGD in the projected Bellman error (PBE), converge robustly under off-policy training and non-linear function approximation. Expected SARSA (van Seijen et al., 2009) has the same convergence guarantee as SARSA, with lower variance. Emphatic-TD (Sutton et al., 2016) emphasizes some updates and de-emphasizes others by reweighting, improving computational efficiency, yet being a semi-gradient method. See Sutton and Barto (2018) for more details. Du et al. (2017) propose variance reduction techniques for policy evaluation to achieve fast convergence. Liu et al. (2018a) study proximal gradient TD learning. White and White (2016) perform empirical comparisons of linear TD methods, and make suggestions about their practical use. Jin et al. (2018) study the sample efficiency of Q-learning. Lu et al. (2018) study non-delusional Q-learning and value-iteration.

		algorithm					
		TD(λ) SARSA(λ)	ADP	LSTD(λ) LSPE(λ)	Fitted-Q	Residual Gradient	GTD(λ) GQ(λ)
issue	linear computation	✓	✓			✓	✓
	nonlinear convergent				✓	✓	✓
	off-policy convergent			✓		✓	✓
	model-free, online	✓		✓		✓	✓
	converges to PBE = 0	✓	✓	✓	✓		✓

Table 2: RL Issues vs. Algorithms, adapted from Sutton (2016)

2.4.10 POLICY OPTIMIZATION

In contrast to value-based methods like TD learning and Q-learning, policy-based methods optimize the policy $\pi(a|s;\boldsymbol{\theta})$ (with function approximation) directly, and update the parameters $\boldsymbol{\theta}$ by gradient ascent. Comparing with value-based methods, policy-based methods usually have better convergence properties, are effective in high-dimensional or continuous action spaces, and can learn stochastic policies. However, policy-based methods usually converge to local optimum, are inefficient to evaluate, and encounter high variance (Silver, 2015). Stochastic policies are important since some problems have only stochastic optimal policies, e.g., in the rock-paper-scissors game, an optimal policy for each player is to take each action (rock, paper, or scissors) with probability 1/3.

For a differentiable policy $\pi(a|s;\boldsymbol{\theta})$, we can compute the policy gradient analytically, whenever it is non-zero,

$$\nabla_{\boldsymbol{\theta}}\pi(a|s;\boldsymbol{\theta}) = \pi(a|s;\boldsymbol{\theta})\frac{\nabla_{\boldsymbol{\theta}}\pi(a|s;\boldsymbol{\theta})}{\pi(a|s;\boldsymbol{\theta})} = \pi(a|s;\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}}\log\pi(a|s;\boldsymbol{\theta}) \tag{24}$$

We call $\nabla_{\boldsymbol{\theta}}\log\pi(a|s;\boldsymbol{\theta})$ score function, or likelihood ratio. Policy gradient theorem (Sutton et al., 2000) states that for a differentiable policy $\pi(a|s;\boldsymbol{\theta})$, the policy gradient is,

$$\mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}}\log\pi(a|s;\boldsymbol{\theta})Q^{\pi_{\boldsymbol{\theta}}}(s,a)]. \tag{25}$$

We omit $\pi_{\boldsymbol{\theta}}$ in value functions below for simplicity.

REINFORCE (Williams, 1992) updates $\boldsymbol{\theta}$ in the direction of

$$\nabla_{\boldsymbol{\theta}}\log\pi(a_t|s_t;\boldsymbol{\theta})R_t, \tag{26}$$

by using return R_t as an unbiased sample of $Q(s_t,a_t)$. Usually a baseline $b_t(s_t)$ is subtracted from the return to reduce the variance of gradient estimate, yet keeping its unbiasedness, to yield the gradient direction,

$$\nabla_{\boldsymbol{\theta}}\log\pi(a_t|s_t;\boldsymbol{\theta})(Q(a_t,s_t) - b_t(s_t)). \tag{27}$$

Using $V(s_t)$ as the baseline $b_t(s_t)$, we have the advantage function,

$$A(a_t,s_t) = Q(a_t,s_t) - V(s_t). \tag{28}$$

Algorithm 6 presents the pseudo code for REINFORCE in the episodic case.

In actor-critic algorithms, the critic updates action-value function parameters, and the actor updates policy parameters, in the direction suggested by the critic. Algorithm 7 presents the pseudo code for one-step actor-critic algorithm in the episodic case.

As summarized in Silver (2015), policy gradient may take various forms: $\nabla_{\boldsymbol{\theta}}\log\pi(a|s;\boldsymbol{\theta})R_t$ for REINFORCE, $\nabla_{\boldsymbol{\theta}}\log\pi(a|s;\boldsymbol{\theta})Q(s,a;\boldsymbol{w})$ for Q actor-critic, $\nabla_{\boldsymbol{\theta}}\log\pi(a|s;\boldsymbol{\theta})A(s,a;\boldsymbol{w})$ for advantage actor-critic, $\nabla_{\boldsymbol{\theta}}\log\pi(a|s;\boldsymbol{\theta})\delta$ for TD actor-critic, $\nabla_{\boldsymbol{\theta}}\log\pi(a|s;\boldsymbol{\theta})\delta e$ for TD(λ) actor-critic, and $G_{\boldsymbol{\theta}}\boldsymbol{w}$ for natural gradient decent, where $G_{\boldsymbol{\theta}} = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}}\log\pi(a|s;\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}}\log\pi(a|s;\boldsymbol{\theta})^T]$ is the Fisher information matrix; and the critic may use Monte Carlo or TD learning for policy evaluation to estimate the value functions Q, A or V.

Input: policy $\pi(a|s, \boldsymbol{\theta})$, $\hat{v}(s, \boldsymbol{w})$
Parameters: step sizes, $\alpha > 0$, $\beta > 0$
Output: policy $\pi(a|s, \boldsymbol{\theta})$

initialize policy parameter $\boldsymbol{\theta}$ and state-value weights $\boldsymbol{w}$
for *true* **do**
    generate an episode $s_0, a_0, r_1, \cdots, s_{T-1}, a_{T-1}, r_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
    for *each step* t *of episode* $0, \cdots, T-1$ **do**
        $G_t \leftarrow$ return from step t
        $\delta \leftarrow G_t - \hat{v}(s_t, \boldsymbol{w})$
        $\boldsymbol{w} \leftarrow \boldsymbol{w} + \beta\delta\nabla_{\boldsymbol{w}}\hat{v}(s_t, \boldsymbol{w})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\gamma^t\boldsymbol{\delta}\nabla_{\boldsymbol{\theta}} log\pi(a_t|s_t, \boldsymbol{\theta})$
    end
end

Algorithm 6: REINFORCE with baseline (episodic), adapted from Sutton and Barto (2018)

Input: policy $\pi(a|s, \boldsymbol{\theta})$, $\hat{v}(s, \boldsymbol{w})$
Parameters: step sizes, $\alpha > 0$, $\beta > 0$
Output: policy $\pi(a|s, \boldsymbol{\theta})$

initialize policy parameter $\boldsymbol{\theta}$ and state-value weights $\boldsymbol{w}$
for *true* **do**
    initialize s, the first state of the episode
    $I \leftarrow 1$
    for s *is not terminal* **do**
        $a \sim \pi(\cdot|s, \boldsymbol{\theta})$
        take action a, observe s', r
        $\delta \leftarrow r + \gamma\hat{v}(s', \boldsymbol{w}) - \hat{v}(s, \boldsymbol{w})$ (if s' is terminal, $\hat{v}(s', \boldsymbol{w}) \doteq 0$)
        $\boldsymbol{w} \leftarrow \boldsymbol{w} + \beta\delta\nabla_{\boldsymbol{w}}\hat{v}(s_t, \boldsymbol{w})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha I\delta\nabla_{\boldsymbol{\theta}} log\pi(a_t|s_t, \boldsymbol{\theta})$
        $I \leftarrow \gamma I$
        $s \leftarrow s'$
    end
end

Algorithm 7: Actor-Critic (episodic), adapted from Sutton and Barto (2018)

2.4.11 DEEP RL

We obtain deep reinforcement learning (deep RL) methods when we use deep neural networks to represent the state or observation, and/or to approximate any of the following components of reinforcement learning: value function, $\hat{v}(s; \boldsymbol{\theta})$ or $\hat{q}(s, a; \boldsymbol{\theta})$, policy $\pi(a|s; \boldsymbol{\theta})$, and model (state transition function and reward function). Here, the parameters $\boldsymbol{\theta}$ are the weights in deep neural networks. When we use "shallow" models, like linear function, decision trees, tile coding and so on as the function approximator, we obtain "shallow" RL, and the parameters $\boldsymbol{\theta}$ are the weight parameters in these models. Note, a shallow model, e.g., decision trees, may be non-linear. The distinct difference between deep RL and "shallow" RL is what function approximator is used. This is similar to the difference between deep learning and "shallow" machine learning. We usually utilize stochastic gradient descent to update weight parameters in deep RL. When off-policy, function approximation, in particular, non-linear function approximation, and bootstrapping are combined together, instability and divergence may occur (Tsitsiklis and Van Roy, 1997). However, recent work like deep Q-network (Mnih et al., 2015) and AlphaGo (Silver et al., 2016a) stabilize the learning and achieve outstanding results. There are efforts for convergence proof of control with non-linear function approximation, e.g., Dai et al. (2018b); Nachum et al. (2018).

2.4.12 BRIEF SUMMARY

An RL problem is formulated as an MDP when the observation about the environment satisfies the Markov property. An MDP is defined by the 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. A central concept in RL is value function. Bellman equations are cornerstone for developing RL algorithms. Temporal difference learning algorithms are fundamental for evaluating/predicting value functions. Control algorithms find optimal policies. Policy-based methods become popular recently. Reinforcement learning algorithms may be based on value function and/or policy, model-free or model-based, on-policy or off-policy, with function approximation or not, with sample backups (TD and Monte Carlo) or full backups (dynamic programming and exhaustive search), and about the depth of backups, either one-step return (TD(0) and dynamic programming) or multi-step return (TD(λ), Monte Carlo, and exhaustive search). When combining off-policy, function approximation, and bootstrapping, we face instability and divergence (Tsitsiklis and Van Roy, 1997), the deadly triad issue (Sutton and Barto, 2018). Theoretical guarantee has been established for linear function approximation, e.g., Gradient-TD (Sutton et al., 2009a;b; Mahmood et al., 2014), Emphatic-TD (Sutton et al., 2016) and Du et al. (2017). When RL integrates with deep neural networks, either for representation or function approximation, we have deep RL. Deep RL algorithms like Deep Q-Network (Mnih et al., 2015) and AlphaGo (Silver et al., 2016a; 2017) stabilize the learning and achieve stunning results.

2.5 RESOURCES

We present some resources for deep RL in the following. We maintain a blog titled Resources for Deep Reinforcement Learning at `https://medium.com/@yuxili/`.

Sutton and Barto's RL book (Sutton and Barto, 2018) covers fundamentals and reflects new progress, e.g., in deep Q-network, AlphaGo, policy gradient methods, as well as in psychology and neuroscience. David Silver's RL course (Silver, 2015) and Sergey Levine's Deep RL course (Levine, 2018) are highly recommended.

Goodfellow et al. (2016) is a recent deep learning book. Bishop (2011), Hastie et al. (2009), and Murphy (2012) are popular machine learning textbooks. James et al. (2013) is an introduction book for machine learning. Domingos (2012), Zinkevich (2017), and Ng (2018) are about practical machine learning advices. See Ng (2016b) and Schulman (2017) for practical advices for deep learning and deep RL respectively.

There are excellent summer schools and bootcamps, e.g., Deep Learning and Reinforcement Learning Summer School: 2018 at `https://dlrlsummerschool.ca`, 2017 at `https://mila.umontreal.ca/en/cours/deep-learning-summer-school-2017/`; Deep Learning Summer School: 2016 at `https://sites.google.com/site/deeplearningsummerschool2016/`, and, 2015 at `https://sites.google.com/site/deeplearningsummerschool/`; and, Deep RL Bootcamp: at `https://sites.google.com/view/deep-rl-bootcamp/`.

Common benchmarks for general RL algorithms are Atari games in the Arcade Learning Environment (ALE) for discrete control, and simulated robots using the MuJoCo physics engine in OpenAI Gym for continuous control.

The Arcade Learning Environment (ALE) (Bellemare et al., 2013; Machado et al., 2017) is a framework composed of Atari 2600 games to develop and evaluate AI agents. OpenAI Gym, at `https://gym.openai.com`, is a toolkit for the development of RL algorithms, consisting of environments, e.g., Atari games and simulated robots, and a site for the comparison and reproduction of results. MuJoCo, Multi-Joint dynamics with Contact, at `http://www.mujoco.org`, is a physics engine. DeepMind Lab (Beattie et al., 2016) is a first-person 3D game platform, at `https://github.com/deepmind/lab`. DeepMind Control Suite (Tassa et al., 2018) provides RL environments with the MuJoCo physics engine, at `https://github.com/deepmind/dm_control`. Dopamine (Bellemare et al., 2018) is a Tensorflow-based RL framework from Google AI. ELF, at `https://github.com/pytorch/ELF`, is a platform for RL research (Tian et al., 2017). ELF OpenGo is a reimplementation of AlphaGo Zero/Alpha Zero using the ELF framework.

PART I: CORE ELEMENTS

A reinforcement learning (RL) agent observes states, executes actions, and receives rewards, with major components of value function, policy and model. A RL problem may be formulated as a prediction, control, or planning problem, and solution methods may be model-free or model-based, and value-based or policy-based. Exploration vs. exploitation is a fundamental tradeoff in RL. Representation is relevant to all elements in RL problems.

Figure 3 illustrates value- and policy-based methods. TD methods, e.g., TD-learning, Q-learning, SARSA, and, Deep Q-network (DQN) (Mnih et al., 2015), are purely value-based. Direct policy search methods include policy gradient, e.g., REINFORCE (Williams, 1992), trust region methods, e.g., Trust Region Policy Optimization (TRPO) (Schulman et al., 2015), and, Proximal Policy Optimization (PPO) (Schulman et al., 2017b), and, evolution methods, e.g., Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen, 2016). Actor-critic methods combine value function and policy. Maximum entropy methods further bridge the gap between value- and policy-based methods, e.g., soft Q-learning (Haarnoja et al., 2017), Path Consistency Learning (PCL) (Nachum et al., 2017), and, trust-PCL (Nachum et al., 2018). Policy iteration, value iteration, and, generalized policy iteration, e.g., AlphaGo (Silver et al., 2016a; 2017) and DeepStack (Moravčík et al., 2017), are based on (approximate) dynamic programming.

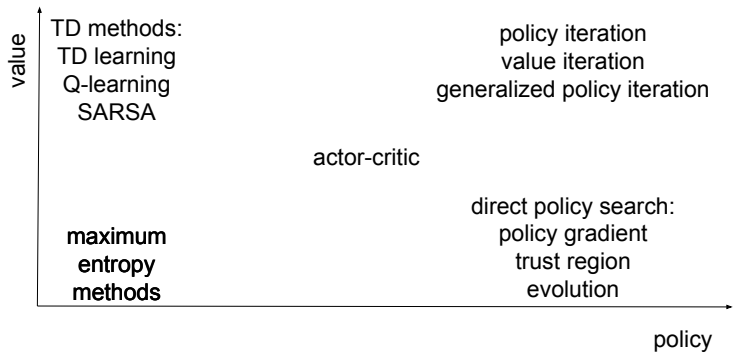

Figure 3: Value- and Policy-based RL Methods

In this part, we discuss RL core elements: value function in Chapter 3, policy in Chapter 4, reward in Chapter 5, model-based RL in Chapter 6, exploration vs exploitation in Chapter 7, and representation in Chapter 8.

3 VALUE FUNCTION

Value function is a fundamental concept in reinforcement learning. A value function is a prediction of the expected, accumulative, discounted, future reward, measuring the goodness of each state, or each state-action pair. Temporal difference (TD) learning (Sutton, 1988) and its extension, Q-learning (Watkins and Dayan, 1992), are classical algorithms for learning state and action value functions respectively. Once we have an optimal value function, we may derive an optimal policy.

In the following, we first introduce Deep Q-Network (DQN) (Mnih et al., 2015), a recent breakthrough, and its extensions. DQN ignited this wave of deep reinforcement learning, combating the stability and convergence issues with experience replay and target networks, which make Q-learning closer to supervised learning. Next we introduce value distribution, rather than value expectation as in classical TD and Q learning. Then we discuss general value function, usually with the goal as a parameter of a value function, besides the state or the state action pair. General value functions hold great promise for further development of RL and AI.

Recently, there are more and more work using policy-based methods to obtain an optimal policy directly. There are also work combing policy gradient with off-policy Q-learning, e.g., Gu et al. (2017b); O'Donoghue et al. (2017); Gu et al. (2017); Haarnoja et al. (2017; 2018); Nachum et al. (2017; 2018); Dai et al. (2018b). Dai et al. (2018b) propose to solve the Bellman equation using primal-dual optimization; instead TD learning and Q-learning are based on fixed point iteration. We discuss policy optimization in next Chapter.

3.1 DEEP Q-LEARNING

Mnih et al. (2015) introduce Deep Q-Network (DQN) and ignite the field of deep RL. There are early work to integrate neural networks with RL, e.g. Tesauro (1994) and Riedmiller (2005). Before DQN, it is well known that RL is unstable or even divergent when action value function is approximated with a nonlinear function like neural networks. That is, the deadly triad issue, when combining off-policy, function approximation, and, bootstrapping. DQN makes several contributions: 1) stabilizing the training of action value function approximation with deep neural networks, in particular, CNNs, using experience replay (Lin, 1992) and target network; 2) designing an end-to-end RL approach, with only the pixels and the game score as inputs, so that only minimal domain knowledge is required; 3) training a flexible network with the same algorithm, network architecture and hyperparameters to perform well on many different tasks, i.e., 49 Atari games (Bellemare et al., 2013), outperforming previous algorithms, and performing comparably to a human professional tester. Note, different games are trained separately, so the network weights are different.

DQN uses a CNN to approximate the optimal action value function,

$$Q^*(s,a) = \max_\pi \mathbb{E}[\sum_0^\infty \gamma^i r_{t+1} | s_t = s, a_t = a, \pi]. \tag{29}$$

In Section 2.4.9, we discuss deadly triad, i.e., instability and divergence may occur, when combining off-policy, function approximation, and bootstrapping. Several factors cause the instability: 1) correlations in sequential observations; 2) small updates to action value function Q may change the policy dramatically, and consequently change the data distribution; 3) correlations between action values Q and $r + \gamma \max_{a'} Q(s', a')$, which is usually used as the target value in batch Q-learning.

DQN uses experience replay and target networks to address the instability issues. In experience replay, observation sequences (s_t, a_t, r_t, s_{t+1}) are stored in the replay buffer, and sampled randomly, to remove correlations in the data, and to smooth data distribution changes. In DQN, experiences are sampled uniformly, and as we discuss later, prioritized experience replay (Schaul et al., 2016) samples experiences according to their importance. A target network keeps its separate network parameters, and update them only periodically, to reduce the correlations between action values Q and the target $r + \gamma \max_{a'} Q(s', a')$. The following is the loss function Q-learning uses to update network parameters at iteration i,

$$(r + \gamma \max_{a'} Q(s', a'; \boldsymbol{\theta}_i^-) - Q(s, a, \boldsymbol{\theta}_i))^2 \tag{30}$$

where $\boldsymbol{\theta}_i$ are parameters of the Q-network at iteration i, $\boldsymbol{\theta}_i^-$ are parameters of the target network at iteration i. The target network parameters $\boldsymbol{\theta}_i^-$ are updated periodically, and held fixed in between. We present DQN pseudo code in Algorithm 8.

Input: the pixels and the game score
Output: Q action value function (from which we obtain a policy and select actions)

---

initialize replay memory D
initialize action-value function Q with random weight $\boldsymbol{\theta}$
initialize target action-value function $\hat{Q}$ with weights $\boldsymbol{\theta}^- = \boldsymbol{\theta}$
for *episode = 1 to M* **do**
  initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  for *t = 1 to T* **do**
    following ϵ-greedy policy, select $a_t = \begin{cases} \text{a random action} & \text{with probability } \epsilon \\ \arg\max_a Q(\phi(s_t), a; \boldsymbol{\theta}) & \text{otherwise} \end{cases}$
    execute action a_i in emulator and observe reward r_t and image x_{t+1}
    set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    `// experience replay`
    sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \boldsymbol{\theta}^-) & \text{otherwise} \end{cases}$
    perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \boldsymbol{\theta}))^2$ w.r.t. the network parameter $\boldsymbol{\theta}$
    `// periodic update of target network`
    in every C steps, reset $\hat{Q} = Q$, i.e., set $\boldsymbol{\theta}^- = \boldsymbol{\theta}$
  end
end

Algorithm 8: Deep Q-Nework (DQN), adapted from Mnih et al. (2015)

DQN has a preprocessing step to reduce the input dimensionality. DQN also uses error clipping, clipping the update $r + \gamma \max_{a'} Q(s', a'; \boldsymbol{\theta}_i^-) - Q(s, a, \boldsymbol{\theta}_i)$ in $[-1, 1]$, to help improve stability. (This is not reflected in the pseudocode.) Mnih et al. (2015) also present visualization results using t-SNE (van der Maaten and Hinton, 2008).

DQN makes a breakthrough by showing that Q-learning with non-linear function approximation, in particular, deep convolutional neural networks, can achieve outstanding results over many Atari games. Following DQN, many work improve DQN in various aspects, e.g., in the following, we will discuss over-estimation in Q-learning, prioritized experience replay, and a dueling network to estimate state value function and associated advantage function, and then combine them to estimate action value function. We also discuss an integration method called Rainbow to combine several techniques together.

In later chapters, we will discuss more extensions, like asynchronous advantage actor-critic (A3C) (Mnih et al., 2016) in Section 4.2, better exploration strategy to improve DQN (Osband et al., 2016) in Chapter 7, and hierarchical DQN in Chapter 11, etc. Experience replay uses more memory and computation for each interaction, and it requires off-policy RL algorithms. This motivates the asynchronous methods as we will discuss in Section 4.2.

See more work as the following. Anschel et al. (2017) propose to reduce variability and instability by an average of previous Q-values estimates. Farebrother et al. (2018) study generalization and regularization in DQN. Guo et al. (2014) combine DQN with offline Monte-Carlo tree search (MCTS) planning. Hausknecht and Stone (2015) propose deep recurrent Q-network (DRQN) to add recurrency to DQN for the partial observability issue in Atari games. He et al. (2017) propose to accelerate DQN by optimality tightening, a constrained optimization approach, to propagate reward faster, and to improve accuracy over DQN. Kansky et al. (2017) propose schema networks and empirically study variants of Breakout in Atari games. Liang et al. (2016) propose to replicate

DQN with shallow features w.r.t. spatial invariance, non-Markovian features, and object detection, together with basic tile-coding features. Zahavy et al. (2018) study action elimination.

See a blog at `https://deepmind.com/research/dqn/`. See Chapter 16 in Sutton and Barto (2018) for a detailed and intuitive description of Deep Q-Network. See Chapter 11 in Sutton and Barto (2018) for more details about the deadly triad.

DOUBLE DQN

van Hasselt et al. (2016) propose Double DQN (D-DQN) to tackle the over-estimate problem in Q-learning (van Hasselt, 2010). In standard Q-learning, as well as in DQN, the parameters are updated as follows:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha(y_t^Q - Q(s_t, a_t; \boldsymbol{\theta}_t))\nabla_{\boldsymbol{\theta}_t} Q(s_t, a_t; \boldsymbol{\theta}_t), \tag{31}$$

where

$$y_t^Q = r_{t+1} + \gamma \max_a Q(s_{t+1}, a; \boldsymbol{\theta}_t), \tag{32}$$

so that the max operator uses the same values to both select and evaluate an action. As a consequence, it is more likely to select over-estimated values, and results in over-optimistic value estimates. van Hasselt et al. (2016) propose to evaluate the greedy policy according to the online network, but to use the target network to estimate its value. This can be achieved with a minor change to the DQN algorithm, replacing y_t^Q with

$$y_t^{D-DQN} = r_{t+1} + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}_t^-), \tag{33}$$

where $\boldsymbol{\theta}_t$ is the parameter for online network and $\boldsymbol{\theta}_t^-$ is the parameter for target network. For reference, y_t^Q can be written as

$$y_t^Q = r_{t+1} + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}_t). \tag{34}$$

PRIORITIZED EXPERIENCE REPLAY

In DQN, experience transitions are uniformly sampled from the replay memory, regardless of the significance of experience. Schaul et al. (2016) propose to prioritize experience replay, so that important experience transitions can be replayed more frequently, to learn more efficiently. The importance of experience transitions are measured by TD errors. The authors design a stochastic prioritization based on the TD errors, using importance sampling to avoid the bias in the update distribution. The authors use prioritized experience replay in DQN and D-DQN, and improve their performance on Atari games. In planning, prioritized sweeping (Moore and Atkeson, 1993) is used to set priorities for state action pairs according to TD errors to achieve more efficient updates.

DUELING ARCHITECTURE

Wang et al. (2016b) propose the dueling network architecture to estimate state value function $V(s)$ and the associated advantage function $A(s, a)$, and then combine them to estimate action value function $Q(s, a)$, to converge faster than Q-learning. In DQN, a CNN layer is followed by a fully connected (FC) layer. In dueling architecture, a CNN layer is followed by two streams of FC layers, to estimate value function and advantage function separately; then the two streams are combined to estimate action value function. Usually we use the following to combine $V(s)$ and $A(s, a)$ to obtain $Q(s, a)$,

$$Q(s, a; \boldsymbol{\theta}, \alpha, \beta) = V(s; \boldsymbol{\theta}, \beta) + \big(A(s, a; \boldsymbol{\theta}, \alpha) - \max_{a'} A(s, a'; \boldsymbol{\theta}, \alpha)\big) \tag{35}$$

where α and β are parameters of the two streams of FC layers. Wang et al. (2016b) propose to replace max operator with average as below for better stability,

$$Q(s, a; \boldsymbol{\theta}, \alpha, \beta) = V(s; \boldsymbol{\theta}, \beta) + \big(A(s, a; \boldsymbol{\theta}, \alpha) - \frac{a}{|\mathcal{A}|} A(s, a'; \boldsymbol{\theta}, \alpha)\big) \tag{36}$$

Dueling architecture implemented with D-DQN and prioritized experience replay improves previous work, DQN and D-DQN with prioritized experience replay, on Atari games.

RAINBOW

Hessel et al. (2018) propose Rainbow to combine DQN, double Q-learning, prioritized replay, dueling networks, multi-step learning, value distribution (discussed in Section 3.2 below), and noisy nets (Fortunato et al. (2018), an exploration technique by adding parametric noises to network weights), and achieve better data efficiency and performance on Atari games. The ablation study show that removing either prioritization or multi-step learning worsens performance for most games; however, the contribution of each component vary significantly for different games.

RETRACE

Munos et al. (2016) propose Retrace(λ) for a safe and efficient return-based off-policy control RL algorithm, for low variance, safe use of samples from any behaviour policy, and efficiency with using samples from close behaviour policies. The authors analyze the property of convergence to the optimal action value Q^*, without the assumption of Greedy in the Limit with Infinite Exploration (GLIE) (Singh et al., 2000), and the convergence of Watkins' $Q(\lambda)$. The authors experiment with Atari games. Gruslys et al. (2017) extend Retrace (Munos et al., 2016) for actor-critic.

3.2 DISTRIBUTIONAL VALUE FUNCTION

Usually, value-based RL methods use expected values, like TD learning and Q-learning. However, expected values usually do not characterize a distribution, and more information about value distribution may be beneficial. Consider a commuter example. For 4 out of 5 days, she takes 15 minutes to commute to work, and for the rest 1 out of 5 days, she takes 30 minutes. On average, she takes 18 minutes to work. However, the commute takes either 15 or 30 minutes, but never 18 minutes.

Bellemare et al. (2017) propose a value distribution approach to RL problems. A value distribution is the distribution of the random return received by a RL agent. In contrast to, and analogous with, the Bellman equation for action value function,

$$Q(s,a) = \mathbb{E}R(s,a) + \gamma \mathbb{E}Q(S',A') \tag{37}$$

Bellemare et al. (2017) establish the distributional Bellman equation,

$$Z(s,a) = R(s,a) + \gamma Z(S',A') \tag{38}$$

Here, Z is the random return, and its expectation is the value Q. Three random variables characterize the distribution of Z: the reward R, the next state action pair (S', A'), and the random return $Z(S', A')$.

Bellemare et al. (2017) prove the contraction of the policy evaluation Bellman operator for the value distribution, so that, for a fixed policy, the Bellman operator contracts in a maximal form of the Wasserstein metric. The Bellman operator for the value distribution, $\mathcal{T}^\pi : \mathcal{Z} \to \mathcal{Z}$, was defined as,

$$\mathcal{T}^\pi Z(s,a) = R(s,a) + \gamma P^\pi Z(s,a) \tag{39}$$

where $P^\pi Z(s,a) = Z(S',A'), S' \sim P(\cdot|s,a), A' \sim \pi(\cdot|S')$. The authors also show instability in the control setting, i.e., in the Bellman optimal equation for the value distribution. The authors argue that value distribution approximations have advantages over expectation approximations, for preserving multimodality in value distributions, and mitigating the effects of a nonstationary policy on learning, and demonstrate the importance of value distribution for RL both theoretically and empirically with Atari games.

Bellemare et al. (2017) propose a categorical algorithm for approximate distributional reinforcement learning. The value distribution is modelled using a discrete distribution, with a set of atoms as the support, $\{z_i = V_{MIN} + i\triangle z : 0 \leq i < N\}$, $\triangle z = \frac{V_{MAX} - V_{MIN}}{N-1}$, where $N \in \mathbb{N}, V_{MIN} \in \mathbb{R}, V_{MAX} \in \mathbb{R}$ are parameters. The atom probabilities are then given by

$$Z_\theta(s,a) = z_i, \text{ with probability, } p_i(s,a) = \frac{e^{\theta_i(s,a)}}{\sum_j e^{\theta_j(s,a)}}. \tag{40}$$

To deal with the issue of disjoint supports caused by Bellman update $\mathcal{T}Z_\theta$, and learning with sample transitions, Bellemare et al. (2017) project the sample update $\hat{\mathcal{T}}Z_\theta$ onto the support of Z_θ, by

computing the Bellman update $\hat{\mathcal{T}} z_j = r + \gamma z_j$ for each atom z_j, then distributing its probability to the immediate neighbours of the update $\hat{\mathcal{T}} z_j$. Use $\Phi \hat{\mathcal{T}} Z_\theta(s, a)$ to denote the operations of a sample Bellman operator following the projection of distributing probabilities. The sample loss function is the cross-entropy term of the KL divergence, and can be optimized with gradient descent,

$$\mathcal{L}_{s,a}(\theta) = D_{KL}(\Phi \hat{\mathcal{T}} Z_\theta(s, a) || Z_\theta(s, a)). \tag{41}$$

Algorithm 9 presents pseudo-code for the categorial algorithm, which computes the sampling Bellman operator followed by a projection of distributing probabilities for one transition sample.

Input: a transition $s_t, a_t, r_t, s_{t+1}, \gamma_t$
Output: cross-entropy loss

// $\{z_i\}$ are atom support, $p_i(s, a)$ are atom probabilities, $i \in \{0, \dots, N-1\}$
$Q(s_{t+1}, a) = \sum_i z_i p_i(s_{t+1}, a)$
$a^* \leftarrow \arg\max_a Q(s_{t+1}, a)$
$m_i = 0, i \in \{0, \dots, N-1\}$
for $j \in \{0, \dots, N-1\}$ **do**
  //compute the projection of $\hat{\mathcal{T}} z_j$ onto the support of $\{z_i\}$
  // $[\cdot]_a^b$ bounds the argument in the range of $[a, b]$
  $\hat{\mathcal{T}} z_j \leftarrow [r + t + \gamma_t z_j]_{V_{MIN}}^{V_{MAX}}$
  // $b_j \in [0, N-1]$
  $b_j \leftarrow (\hat{\mathcal{T}} - V_{MIN}) / \triangle z$
  $l \leftarrow \lfloor b_j \rfloor, u \leftarrow \lceil b_j \rceil$
  //distribute probability of $\hat{\mathcal{T}} z_j$
  $m_l \leftarrow m_l + p_j(x_{t+1}, a^*)(u - b_j)$
  $m_u \leftarrow m_u + p_j(x_{t+1}, a^*)(b_j - l)$
end
output $-\sum_i m_i \log p_i(x_t, a_t)$

Algorithm 9: Categorical Algorithm, adapted from Bellemare et al. (2017)

Morimura et al. (2010a;b) propose risk sensitive algorithms. Rowland et al. (2018) analyze the categorical distributional reinforcement learning proposed in Bellemare et al. (2017). Doan et al. (2018) propose GAN Q-learning for distributional RL. Dabney et al. (2018) propose to utilize quantile regression for state-action return distribution.

See a talk by Marc Bellemare at `https://vimeo.com/235922311`. See a blog at `https://deepmind.com/blog/going-beyond-average-reinforcement-learning/`, with a video about Atari games experiments.

3.3 General Value Function

Horde

Sutton et al. (2011) discuss that value functions provide semantics for predictive knowledge and goal-oriented (control) knowledge, and propose to represent knowledge with general value function, where policy, termination function, reward function, and terminal reward function are parameters. The authors then propose Horde, a scalable real-time architecture for learning in parallel general value functions for independent sub-agents from unsupervised sensorimotor interaction, i.e., nonreward signals and observations. Horde can learn to predict the values of many sensors, and policies to maximize those sensor values, with general value functions, and answer predictive or goal-oriented questions. Horde is off-policy, i.e., it learns in real-time while following some other behaviour policy, and learns with gradient-based temporal difference learning methods, with constant time and memory complexity per time step.

UNIVERSAL VALUE FUNCTION APPROXIMATORS

Schaul et al. (2015) propose Universal Value Function Approximators (UVFAs) $V(s, g; \theta)$ to generalize over both states s and goals g, to extend normal state value function approximators. The aim of UVFAs is to exploit structure across both states and goals, by taking advantages of similarities encoded in the goal representation, and the structure in the induced value function. Schaul et al. (2015) show that such UVFAs can be trained using direct bootstrapping with a variant of Q-learning, and the derived greedy policy can generalize to previously unseen action-goal pairs. UVFAs can be regarded as an infinite Horde of demons, without scalability issue as in Horde.

HINDSIGHT EXPERIENCE REPLAY

Andrychowicz et al. (2017) propose Hindsight Experience Replay (HER) to combat with the sparse reward issue, inspired by UVFAs (Schaul et al., 2015). The idea is, after experiencing some episodes, storing every transition in the replay buffer, with not only the original goal for this episode, but also some other goals. HER can replay each trajectory with any goal with an off-policy RL algorithm, since the original goal pursued does not influence the environment dynamics, albeit it influences the actions. Andrychowicz et al. (2017) combine HER with DQN and DDPG (as will be discussed in Section 4.1) on several robotics arm tasks, push, slide and pick-and-place, and perform well.

An OpenAI blog describes HER, with videos about HER experiments, introduces a baseline HER implementation and simulated robot environment, and proposes potential research topics with HER, `https://blog.openai.com/ingredients-for-robotics-research/`.

4 POLICY

A policy maps a state to an action, or, a distribution over actions, and policy optimization is to find an optimal mapping. Value-based methods optimize value functions first, then derive optimal policies. Policy-based methods directly optimize an objective function, usually cumulative rewards.

REINFORCE (Williams, 1992) is a classical policy gradient method, with a Monte Carlo approach to estimate policy gradients. Policy gradient methods can stay stable when combined with function approximation, under some conditions. However, sample inefficiency is a major issue; and Monte Carlo sampling with rollouts usually results in high variance in policy gradient estimates.

Incorporating a baseline or critic can help reduce variance. In actor-critic algorithms (Barto et al., 1983; Konda and Tsitsiklis, 2003), the critic updates action value function parameters, the actor updates policy parameters, in the direction suggested by the critic, and the value function can help reduce the variance of policy parameter estimates, by replacing rollout estimates, with the possibility of encountering a higher bias. We discuss policy gradient, actor critic, and REINFORCE in Section 2.4.10.

On-policy methods, like TD learning, are usually sample inefficient by using data only once, with estimations based on trajectories from the current policy. Off-policy methods, like Q-learning, can learn from any trajectories from any policies, e.g., expert demonstrations, from the same environment. Recently, experience replay regains popularity after DQN, as we discuss in Chapter 3. This usually makes off-policy methods more sample efficient than on-policy methods. Importance sampling is a variance reduction technique in Monte Carlo methods. Precup et al. (2001) study off-policy TD learning with importance sampling. Degris et al. (2012) propose off-policy actor-critic with importance sampling. Liu et al. (2018b) propose an off-policy estimation method with importance sampling to avoid the exploding variance issue.

Kakade (2002) introduce natural policy gradient to improve stability and convergence speed of policy based methods. This leads to trust region methods, like Trust Region Policy Optimization (TRPO) (Schulman et al., 2015), and Proximal Policy Optimization (PPO) (Schulman et al., 2017b), two on-policy methods. Trust-PCL (Nachum et al., 2018) is an off-policy trust region method.

It is desirable to improve data efficiency of policy gradient, while keeping its stability and unbiasedness. Asynchronous advantage actor-critic (A3C) (Mnih et al., 2016) integrates policy gradient with on-line critic. There are recent work for policy gradient with off-policy critic, like deep deterministic policy gradient (DDPG) (Lillicrap et al., 2016), and policy gradient with Q-learning (PGQL) (O'Donoghue et al., 2017). Interpolated policy gradient (Gu et al., 2017), and Q-Prop (Gu et al., 2017b) are proposed to combine stability of trust region methods with off-policy sample efficiency. However, the deadly triad issue results from the combination off-policy, function approximation, and bootstrapping, so that instability and divergence may occur (Sutton and Barto, 2018).

There are efforts to establish theoretical guarantees, tackling the deadly triad, e.g., Retrace (Munos et al., 2016), path consistency learning (PCL) (Nachum et al., 2017), Trust-PCL (Nachum et al., 2018), and SBEED (Dai et al., 2018b), etc.

Maximum entropy methods integrate policy gradient with off-policy learning and attempt to bridge the gap between value- and policy-based methods, e.g., Ziebart et al. (2008), Ziebart et al. (2010), G-learning (Fox et al., 2016), soft Q-learning (Haarnoja et al., 2017), and several mentioned above, including PGQL, PCL, and Trust-PCL.

In conventional RL, we are content with deterministic policies (Sutton and Barto, 2018). However, stochastic policies are desirable sometimes, for reasons like, partial observable environments, handling uncertainty (Ziebart et al., 2008; 2010), convergence and computation efficiency (Gu et al., 2017b), exploration and compositionality (Haarnoja et al., 2017), and optimal solutions for some games like rock-paper-scissors, etc. Policy gradient methods usually obtain stochastic policies. So are maximum entropy regularized methods.

Here we focus on model-free policy optimization algorithms. We will discuss model-based ones, like stochastic value gradients (SVG) (Heess et al., 2015), and normalized advantage functions (NAF) (Gu et al., 2016), in Chapter 6. we discuss guided policy search (GPS) (Levine et al., 2016) in Chapter 16.

Evolution strategies achieve excellent results, e.g., Petroski Such et al. (2017), Salimans et al. (2017), Lehman et al. (2017). See Hansen (2016) for a tutorial. Khadka and Tumer (2018) propose evolutionary reinforcement learning.

Policy search methods span a wide spectrum from direct policy search to value-based RL, includes: evolutionary strategies, covariance matrix adaptation evolution strategy (CMA-ES) (Hansen and Ostermeier, 2001; Hansen, 2016), episodic relative entropy policy search (REPS) (Jan Peters, 2010), policy gradients, probabilistic inference for learning control (PILCO) (Deisenroth and Rasmussen, 2011), model-based REPS (Abdolmaleki et al., 2015), policy search by trajectory optimization (Levine and Koltun, 2014), actor critic, natural actor critic (Kakade, 2002), episodic natural actor critic (eNAC), advantage weighted regression (Peters and Schaal, 2007), conservative policy iteration (Kakade and Langford, 2002), least square policy iteration (LSPI) (Lagoudakis and Parr, 2003), Q-learning, and fitted Q-learning (Riedmiller, 2005). See Peters and Neumann (2015) for more details. AlphaGo (Silver et al., 2016a; 2017), as well as Sun et al. (2018), follows the scheme of generalized policy iteration. We will discuss AlphaGo in Section 15.1.

See Abbeel (2017b) for a tutorial on policy optimization. Levine (2018) discusses connections between RL and control, in particular, maximum entropy RL, and probabilistic inference. See NIPS 2018 Workshop on Infer to Control: Probabilistic Reinforcement Learning and Structured Control, at `https://sites.google.com/view/infer2control-nips2018`.

In the following, we discuss policy gradient in Section 4.1, actor-critic in Section 4.2, trust region methods in Section 4.3, policy gradient with off-policy learning in Section 4.4, and, benchmark results in Section 4.5.

4.1 Policy Gradient

Policy gradients are popular methods in RL, optimizing policies directly. Policies may be deterministic or stochastic. Silver et al. (2014) propose Deterministic Policy Gradient (DPG) and Lillicrap et al. (2016) extend it to deep DPG (DDPG) for efficient estimation of policy gradients. Houthooft et al. (2018) propose evolved policy gradients with meta-learning.

As discussed in Heess et al. (2015), most policy gradient methods, like REINFORCE, use likelihood ratio method as discussed in Section 2.4.10, by sampling returns from interactions with the environment in a model-free manner; another approach, value gradient method, is to estimate the gradient via backpropagation, and DPG and DDPG follow this approach.

Silver et al. (2014) introduce the Deterministic Policy Gradient (DPG) algorithm for RL problems with continuous action spaces. The deterministic policy gradient is the expected gradient of the action-value function, which integrates over the state space; whereas in the stochastic case, the policy gradient integrates over both state and action spaces. Consequently, the deterministic policy gradient can be estimated more efficiently than the stochastic policy gradient. The authors introduce an off-policy actor-critic algorithm to learn a deterministic target policy from an exploratory behaviour policy, and to ensure unbiased policy gradient with the compatible function approximation for deterministic policy gradients. Empirical results show its superior to stochastic policy gradients, in particular in high dimensional tasks, on several problems: a high-dimensional bandit; standard benchmark RL tasks of mountain car, pendulum, and 2D puddle world with low dimensional action spaces; and controlling an octopus arm with a high-dimensional action space. The experiments are conducted with tile-coding and linear function approximators.

Lillicrap et al. (2016) propose an actor-critic, model-free, Deep Deterministic Policy Gradient (DDPG) algorithm in continuous action spaces, by extending DQN (Mnih et al., 2015) and DPG (Silver et al., 2014). With actor-critic as in DPG, DDPG avoids the optimization of action value function at every time step to obtain a greedy policy as in Q-learning, which will make it infeasible in complex action spaces with large, unconstrained function approximators like deep neural networks. To make the learning stable and robust, similar to DQN, DDPQ deploys experience replay and an idea similar to target network, a "soft" target, which, rather than copying the weights directly as in DQN, updates the soft target network weights $\boldsymbol{\theta}'$ slowly to track the learned networks weights $\boldsymbol{\theta}$: $\boldsymbol{\theta}' \leftarrow \tau\boldsymbol{\theta} + (1-\tau)\boldsymbol{\theta}'$, with $\tau \ll 1$. The authors adapt batch normalization to handle the issue that the different components of the observation with different physical units. As an off-policy algorithm, DDPG learns an actor policy with experiences from an exploration policy by adding noises sampled

from a noise process to the actor policy. More than 20 simulated physics tasks of varying difficulty in the MuJoCo environment are solved with the same learning algorithm, network architecture and hyper-parameters, and obtain policies with performance competitive with those found by a planning algorithm with full access to the underlying physical model and its derivatives. DDPG can solve problems with 20 times fewer steps of experience than DQN, although it still needs a large number of training episodes to find solutions, as in most model-free RL methods. It is end-to-end, with raw pixels as input.

Hausknecht and Stone (2016) extend DDPG by considering parameterization of action spaces, and experiment with the domain of simulated RoboCup soccer.

4.2 ACTOR-CRITIC

An actor critic algorithm learns both a policy and a state value function, and the value function is used for bootstrapping, i.e., updating a state from subsequent estimates, to reduce variance and accelerate learning (Sutton and Barto, 2018).

In the following, we focus on asynchronous advantage actor-critic (A3C) (Mnih et al., 2016). Mnih et al. (2016) also discuss asynchronous one-step SARSA, one-step Q-learning and n-step Q-learning. A3C achieves the best performance among these asynchronous methods, and it can work for both discrete and continuous cases.

---

global shared parameter vectors $\boldsymbol{\theta}$ and $\boldsymbol{\theta}_v$, thread-specific parameter vectors $\boldsymbol{\theta}'$ and $\boldsymbol{\theta}'_v$
global shared counter $T = 0$, T_{max}
initialize step counter $t \leftarrow 1$
for $T \leq T_{max}$ **do**
  reset gradients, $d\boldsymbol{\theta} \leftarrow 0$ and $d\boldsymbol{\theta}_v \leftarrow 0$
  synchronize thread-specific parameters $\boldsymbol{\theta}' = \boldsymbol{\theta}$ and $\boldsymbol{\theta}'_v = \boldsymbol{\theta}_v$
  set $t_{start} = t$, get state s_t
  for s_t *not terminal and* $t - t_{start} \leq t_{max}$ **do**
    take a_t according to policy $\pi(a_t|s_t;\boldsymbol{\theta}')$
    receive reward r_t and new state s_{t+1}
    $t \leftarrow t+1, T \leftarrow T+1$
  end
  $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t, \boldsymbol{\theta}'_v) & \text{otherwise} \end{cases}$
  for $i \in \{t-1, ..., t_{start}\}$ **do**
    $R \leftarrow r_i + \gamma R$
    accumulate gradients wrt $\boldsymbol{\theta}'$: $d\boldsymbol{\theta} \leftarrow d\boldsymbol{\theta} + \nabla_{\boldsymbol{\theta}'} \log \pi(a_i|s_i;\boldsymbol{\theta}')(R - V(s_i;\boldsymbol{\theta}'_v))$
    accumulate gradients wrt $\boldsymbol{\theta}'_v$: $d\boldsymbol{\theta}_v \leftarrow d\boldsymbol{\theta}_v + \nabla_{\boldsymbol{\theta}'_v}(R - V(s_i;\boldsymbol{\theta}'_v))^2$
  end
  update asynchronously $\boldsymbol{\theta}$ using $d\boldsymbol{\theta}$, and $\boldsymbol{\theta}_v$ using $d\boldsymbol{\theta}_v$
end

Algorithm 10: A3C, each actor-learner thread, based on Mnih et al. (2016)

We present pseudo code for A3C for each actor-learner thread in Algorithm 10. A3C maintains a policy $\pi(a_t|s_t;\boldsymbol{\theta})$ and an estimate of the value function $V(s_t;\boldsymbol{\theta}_v)$, being updated with n-step returns in the forward view, after every t_{max} actions or reaching a terminal state, similar to using minibatches. In n-step update, $V(s)$ is updated toward the n-step return, defined as,

$$r_t + \gamma r_{t+1} + \cdots + \gamma^{n-1} r_{t+n-1} + \gamma^n V(s_{t+n}). \tag{42}$$

Lines 15-19 show, each n-step update results in a one-step update for the last state, a two-step update for the second last state, and so on for a total of up to t_{max} updates, for both policy and value function parameters. The gradient update can be seen as

$$\nabla_{\boldsymbol{\theta}'} \log \pi(a_t|s_t;\boldsymbol{\theta}') A(s_t, a_t; \boldsymbol{\theta}, \boldsymbol{\theta}_v), \tag{43}$$

where

$$A(s_t, a_t; \boldsymbol{\theta}, \boldsymbol{\theta}_v) = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{t+k};\boldsymbol{\theta}_v) - V(s_t;\boldsymbol{\theta}_v) \tag{44}$$

is an estimate of the advantage function, with k upbound by t_{max}.

In A3C, parallel actors employ different exploration policies to stabilize training, so that experience replay is not utilized, although experience replay could improve data efficiency. Experience replay uses more memory and computation for each interaction, and it requires off-policy RL algorithms. Asynchronous methods can use on-policy RL methods. Moreover, different from most deep learning algorithms, asynchronous methods can run on a single multi-core CPU.

For Atari games, A3C runs much faster yet performs better than or comparably with DQN, Gorila (Nair et al., 2015), D-DQN, Dueling D-DQN, and Prioritized D-DQN. A3C also succeeds on continuous motor control problems: TORCS car racing games and MujoCo physics manipulation and locomotion, and Labyrinth, a navigating task in random 3D mazes using visual inputs.

Wang et al. (2017c) propose ACER, a stable and sample efficient actor-critic deep RL model using experience replay, with truncated importance sampling, stochastic dueling network (Wang et al. (2016b) as discussed in Section 3.1), and trust region policy optimization (Schulman et al. (2015) as will be discussed in Section 4.3). Babaeizadeh et al. (2017) propose a hybrid CPU/GPU implementation of A3C. Gruslys et al. (2017) propose Reactor to extend Retrace (Munos et al., 2016) for the actor-critic scheme. Horgan et al. (2018) propose Apex, a distributed version of actor-critic, with prioritized experience replay, and improve the performance on Atari games substantially. One important factor is that Apex can learn on a large amount of data. Espeholt et al. (2018) propose IMPALA, a distributed actor-critic agent, and show good performance in multi-task settings. Dai et al. (2018a) propose dual actor-critic, in which the critic is not learned by standard algorithms like TD but is optimized to help compute gradient of the actor.

4.3 Trust Region Methods

Trust region methods are an approach to stabilize policy optimization by constraining gradient updates. In the following, we discuss Trust Region Policy Optimization (TRPO) (Schulman et al., 2015), and Proximal Policy Optimization (PPO) (Schulman et al., 2017b). Nachum et al. (2018) propose Trust-PCL, and extension of TRPO for off-policy learning, which we will discuss in Section 4.4. Heess et al. (2017) propose distributed proximal policy optimization. Wu et al. (2017) propose scalable TRPO with Kronecker-factored approximation to the curvature. Liu et al. (2018a) study proximal gradient TD learning. See a video about TRPO at, `https://sites.google.com/site/trpopaper/`. See a blog about PPO with videos at, `https://blog.openai.com/openai-baselines-ppo/`.

Trust Region Policy Optimization (TRPO)

Schulman et al. (2015) introduce an iterative procedure to monotonically improve policies theoretically, guaranteed by optimizing a surrogate objective function, and then make several approximations to develop a practical algorithm, Trust Region Policy Optimization (TRPO). In brief summary, TRPO iterates the following steps:

1. collect state action pairs and Monte Carlo estimates of Q values
2. average samples, construct the estimated objective and constraint in the previous optimization problem, where, θ_{old} denotes previous policy parameters, q denotes the sampling distribution, and δ is the trust region parameter

$$\max_{\theta} \hat{\mathbb{E}}\Big[\frac{\pi_\theta(a|s)}{q(a|s)} Q_{\theta_{old}}(s,a)\Big] \text{ subject to } \hat{\mathbb{E}}[KL(\pi_{\theta_{old}}(\cdot|s)\|\pi_\theta(\cdot|s))] \leq \delta \tag{45}$$

3. solve the above constrained optimization problem approximately, update the policy parameter θ

Schulman et al. (2015) unify policy iteration and policy gradient with analysis, and show that policy iteration, policy gradient, and natural policy gradient (Kakade, 2002) are special cases of TRPO. In the experiments, TRPO methods perform well on simulated robotic tasks of swimming, hopping, and walking, as well as playing Atari games in an end-to-end manner directly from raw images.

PROXIMAL POLICY OPTIMIZATION (PPO)

Schulman et al. (2017b) propose Proximal Policy Optimization (PPO), to alternate between data sampling and optimization, and to benefit the stability and reliability from TRPO, with the goal of simpler implementation, better generalization, and better empirical sample complexity. In PPO, parameters for policy and value function can be shared in a neural network, and advantage function can be estimated to reduce variance of policy parameters estimation. PPO utilizes a truncated version of Generalized Advantage Estimator (GAE) (Schulman et al., 2016), reducing to multi-step TD update when $\lambda = 1$, $\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}$, where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$. PPO achieves good performance on several continuous tasks in MuJoCo, on continuous 3D humanoid running and steering, and on discrete Atari games. As mentioned in an OpenAI blog about PPO, `https://blog.openai.com/openai-baselines-ppo/`, "PPO has become the default reinforcement learning algorithm at OpenAI because of its ease of use and good performance".

4.4 POLICY GRADIENT WITH OFF-POLICY LEARNING

It is desirable to combine stability and unbiasedness of policy gradient, and sample efficiency of off-policy learning. Levine (2018) connects RL and control with probabilistic inference, and discusses that maximum entropy RL is equivalent to exact and variational probabilistic inference in deterministic and stochastic dynamics respectively. We discuss several recent works following the approach of maximum entropy RL, including Haarnoja et al. (2017), Nachum et al. (2017), Nachum et al. (2018), Haarnoja et al. (2018), Gu et al. (2017b), etc. Maximum entropy RL can help exploration, compositionality, and partial observability (Levine, 2018).

SOFT Q-LEARNING

Haarnoja et al. (2017) design a soft Q-learning algorithm, by applying a method for learning energy-based policies to optimize maximum entropy policies. In soft Q-learning, an optimal policy is expressed with a Boltzmann distribution, and a variational method is employed to learn a sampling network to approximate samples from this distribution. Soft Q-learning can improve exploration, and help stochastic energy-based policies achieve compositionality for better transferability.

Haarnoja et al. (2018) propose soft actor-critic, based on the maximum energy RL framework in (Haarnoja et al., 2017), so that the actor aims to maximize both expected reward and entropy. Schulman et al. (2017a) show equivalence between entropy-regularized Q-learning and policy gradient. Kavosh and Littman (2017) propose a new Q-value operator.

PATH CONSISTENCY LEARNING (PCL)

Nachum et al. (2017) introduce the notion of softmax temporal consistency, to generalize the hardmax Bellman consistency as in off-policy Q-learning, and in contrast to the average consistency as in on-policy SARSA and actor-critic. The authors establish the correspondence and a mutual compatibility property between softmax consistent action values and the optimal policy maximizing entropy regularized expected discounted reward. The authors propose Path Consistency Learning (PCL), attempting to bridge the gap between value and policy based RL, by exploiting multi-step path-wise consistency on traces from both on and off policies. The authors experiment with several algorithmic tasks. Soft Q-learning (Haarnoja et al., 2017) is a one-step special case of PCL.

Nachum et al. (2018) propose Trust-PCL, an off-policy trust region method, to address sample inefficiency issue with the on-policy nature of trust region methods like TRPO and PPO. The authors observe that an objective function maximizing rewards, regularized by relative entropy, led to an optimal policy and state value function which satisfy several multi-step pathwise consistencies along any path. Therefore, Trust-PCL achieves stability and off-policy sample efficiency by employing relative entropy regularization. The authors design a method to determine the coefficient for the relative entropy regularization term, to simplify the task of hyperparameter tuning. The authors experiment on standard continuous control tasks.

Dai et al. (2018b) reformulate the Bellman equation into a primal-dual optimization problem, and propose smoothed Bellman error embedding (SBEED) to solve it. The authors provide "the first convergence guarantee for general non-linear function approximation, and analyze the algorithm's

sample complexity", and experiment with several control tasks. SBEED can be viewed as a debiased version of PCL and generalizes PCL.

RECENT WORK

Gu et al. (2017b) propose Q-Prop to take advantage of the stability of policy gradient and the sample efficiency of off-policy learning. Q-Prop utilizes a Taylor expansion of the off-policy critic as a control variate, which gives an analytical gradient term through critic, and a Monte Carlo policy gradient term.

O'Donoghue et al. (2017) propose PGQL to combine policy gradient with off-policy Q-learning, to benefit from experience replay. The authors also show that action value fitting techniques and actor-critic methods are equivalent, and interprete regularized policy gradient techniques as advantage function learning algorithms.

Gu et al. (2017) show that the interpolation of off-policy updates with value function estimation and on-policy policy gradient updates can satisfy performance guarantee. The authors employ control variate methods for analysis, and design a family of policy gradient algorithms, with several recent ones as special cases, including Q-Prop, PGQL, and ACER (Wang et al., 2017c). The author study the correspondence between the empirical performance and the degree of mixture of off-policy gradient estimates with on-policy samples, on several continuous tasks.

4.5 BENCHMARK RESULTS

Duan et al. (2016) present a benchmark study for continuous control tasks, including classic tasks like cart-pole, tasks with very large state and action spaces such as 3D humanoid locomotion and tasks with partial observations, and tasks with hierarchical structure. The authors implement and compare various algorithms, including batch algorithms: REINFORCE, Truncated Natural Policy Gradient (TNPG), Reward-Weighted Regression (RWR), Relative Entropy Policy Search (REPS), Trust Region Policy Optimization (TRPO), Cross Entropy Method (CEM), Covariance Matrix Adaption Evolution Strategy (CMA-ES); and online algorithms: Deep Deterministic Policy Gradient (DDPG); and recurrent variants of batch algorithms. See the open source at `https://github.com/rllab/rllab`.

Henderson et al. (2018) investigate reproducibility, experimental techniques, and reporting procedures for deep RL. The authors show that hyperparameters, including network architecture and reward scale, random seeds and trials, environments (like Hopper or HalfCheetah etc. in OpenAI Baseline), and codebases influenced experimental results. This causes difficulties for reproducing deep RL results.

Tassa et al. (2018) present the DeepMind Control Suite, a set of continuous tasks, implemented in Python, based on the MuJoCo physics engine. Tassa et al. (2018) include benchmarks for A3C, DDPG, and distributed distributional deterministic policy gradients (D4PG) (Barth-Maron et al., 2018). The open source is at, `https://github.com/deepmind/dm_control`, and a video showing the tasks is at, `https://youtu.be/rAai4QzcYbs`.

5 REWARD

Rewards provide evaluative feedbacks for a RL agent to make decisions. Reward function is a mathematical formulation for rewards.

Rewards may be sparse so that it is challenging for learning algorithms, e.g., in computer Go, a reward occurs at the end of a game. Hindsight experience replay (Andrychowicz et al., 2017) is a way to handle sparse rewards, as we discuss in Chapter 3. Unsupervised auxiliary learning (Jaderberg et al., 2017) is an unsupervised ways to harness environmental signals, as we discuss in Chapter 10. Reward shaping is to modify reward function to facilitate learning while maintaining optimal policy. Ng et al. (2000) show that potential-based reward shaping can maintain optimality of the policy. Reward shaping is usually a manual endeavour. Jaderberg et al. (2018) employ a learning approach in an end-to-end training pipeline.

Reward functions may not be available for some RL problems. In imitation learning, an agent learns to perform a task from expert demonstrations, with samples of trajectories from the expert, without reinforcement signal. Two main approaches for imitation learning are behavioral cloning and inverse reinforcement learning. Behavioral cloning, or learning from demonstration, maps state-action pairs from expert trajectories to a policy, maybe as supervised learning, without learning the reward function (Ho et al., 2016; Ho and Ermon, 2016).

Levine (2018) discusses about imitation learning and RL. Pure imitation learning is supervised learning, stable and well-studied; however, it encounters the issue of distributional shift, and it can not perform better than the demonstrations. Pure RL is unbiased, and can improve until optimal, however, with challenging issues of exploration and optimization. Initialization with imitation learning then fine-tuning with RL can take advantage of both approaches; however, it can forget initialization from demonstration due to distributional shift. Pure RL with demonstrations as off-policy data is still RL, keeping advantages of RL; however, demonstrations may not always help. A hybrid objective including both RL and imitation objectives, can take advantage of both, do not forget demonstrations; however, it is not pure RL, may be biased, and may require considerable tuning.

Inverse reinforcement learning (IRL) is the problem of determining a reward function given observations of optimal behaviour (Ng and Russell, 2000). Abbeel and Ng (2004) approach apprenticeship learning via IRL. Finn et al. (2016b) study inverse optimal cost. Probabilistic approaches are developed for IRL with maximum entropy (Ziebart et al., 2008) and maximum causal entropy (Ziebart et al., 2010) to deal with uncertainty in noisy and imperfect demonstrations. Ross et al. (2010) reduce imitation learning and structured prediction (Daumé et al., 2009) to no-regret online learning, and propose DAGGER, which requires interaction with the expert. Syed and Schapire (2007), Syed et al. (2008), and Syed and Schapire (2010) study apprenticeship learning with linear programming, game theory, and reduction to classification.

A reward function may not represent the intention of the designer. A negative side effect of a misspecified reward refers to potential poor behaviours resulting from missing important aspects (Amodei et al., 2016). Hadfield-Menell et al. (2017) give an old example about the wish of King Midas, that everything he touched, turned into gold. Unfortunately, his intention did not include food, family members, and many more. Russell and Norvig (2009) give an example that a vacuum cleaner collects more dust to receive more rewards by ejecting collected dust.

Singh et al. (2009) and Singh et al. (2010) discuss fundamental issues like, homeostatic vs. non-homeostatic (heterostatic) theories, primary rewards vs. conditioned or secondary rewards, internal vs. external environments, intrinsic vs. extrinsic motivation, and, intrinsic vs. extrinsic reward. The authors then formulate an optimal reward computational framework. Oudeyer and Kaplan (2007) present a typology of computational approaches to these concepts.

See Yue and Le (2018) for a tutorial on imitation learning, Rhinehart et al. (2018) for a tutorial on IRL for computer vision, and, Argall et al. (2009) for a survey of robot learning from demonstration. See NIPS 2018 Workshop on Imitation Learning and its Challenges in Robotics, at `https://sites.google.com/view/nips18-ilr`.

In the following, we first discuss RL methods with and without reward learning respectively, when there is no reward function given. We then discuss an approach to handle complex reward functions.

See also Amin et al. (2017), Huang et al. (2018), Leike et al. (2018), Merel et al. (2017), Stadie et al. (2017), Su et al. (2016), Wang et al. (2017), and, Zheng et al. (2018b).

We discuss robotics with imitation learning in Section 16.2, including Duan et al. (2017); Finn et al. (2017c); Yu et al. (2018); Finn et al. (2016b). Hu et al. (2018d) leverage IRL techniques to learn knowledge constraints in deep generative models.

Reward Learning

Hadfield-Menell et al. (2016) observe flaws with IRL: a robot may take a human's reward function as its own, e.g., a robot may learn to drink coffee, rather than learn to cook coffee; and, by assuming optimal demonstrations which achieve a task efficiently, a robot may miss chances to learn useful behaviours, e.g., a robot learns to cook coffee by passive observing would miss chances to learn many useful skills during the process of cooking coffee by active teaching and learning.

The authors then propose a cooperative inverse reinforcement learning (CIRL) game for the value alignment problem. CIRL is a two-player game of partial information, with a human, knowing the reward function, a robot, not knowing it, and the robot's payoff is human's reward. An optimal solution to CIRL maximizes the human reward, and may involve active teaching by the human and active learning by the robot. The authors reduce finding an optimal policy pair for human and robot to the solution of a single agent POMDP problem, prove that optimality in isolation, like apprenticeship learning and inverse reinforcement learning, is suboptimal in CIRL, and present an approximate algorithm to solve CIRL.

Hadfield-Menell et al. (2017) introduce inverse reward design (IRD), to infer the true reward function, based on a designed reward function, an intended decision problem, e.g., an MDP, and a set of possible reward functions. The authors propose approximate algorithms to solve the IRD problem, and experiment with the risk-averse behaviour derived from planning with the resulting reward function. Experiments show that IRD reduces chances of undesirable behaviours like misspecified reward functions and reward hacking.

Christiano et al. (2017) propose to learn a reward function based on human preferences by comparing pairs of trajectory segments. The proposed method maintains a policy and an estimated reward function, approximated by deep neural networks. The networks are updated asynchronously with three processes iteratively: 1) produce trajectories with the current policy, and the policy is optimized with a traditional RL method, 2) select pairs of segments from trajectories, obtain human preferences, and, 3) optimize the reward function with supervised learning based on human preferences. Experiments show the proposed method can solve complex RL problems like Atari games and simulated robot locomotion.

Learning from Demonstration

Here we discuss several recent papers without reward learning.

We first discuss deep Q-learning from demonstrations. Hester et al. (2018) propose deep Q-learning from demonstrations (DQfD) to attempt to accelerate learning by leveraging demonstration data, using a combination of temporal difference (TD), supervised, and regularized losses. In DQfQ, reward signal is not available for demonstration data; however, it is available in Q-learning. The supervised large margin classification loss enables the policy derived from the learned value function to imitate the demonstrator; the TD loss enables the validity of value function according to the Bellman equation and its further use for learning with RL; the regularization loss function on network weights and biases prevents overfitting on small demonstration dataset. In the pre-training phase, DQfD trains only on demonstration data, to obtain a policy imitating the demonstrator and a value function for continual RL learning. After that, DQfD self-generates samples, and mixes them with demonstration data according to certain proportion to obtain training data. Experiments on Atari games show DQfD in general has better initial performance, more average rewards, and learns faster than DQN.

In AlphaGo (Silver et al., 2016a), as we discuss in Section 15.1, the supervised learning policy network is learned from expert moves as learning from demonstration; the results initialize the RL policy network. See also Kim et al. (2014); Pérez-D'Arpino and Shah (2017); Večerík et al. (2017).

Now we discuss generative adversarial imitation learning. With IRL, an agent learns a reward function first, then from which derives an optimal policy. Many IRL algorithms have high time complexity, with a RL problem in the inner loop. Ho and Ermon (2016) propose the generative adversarial imitation learning (GAIL) algorithm to learn policies directly from data, bypassing the intermediate IRL step. Generative adversarial training is deployed to fit the discriminator, about the distribution of states and actions that defines expert behavior, and the generator, representing the policy. Generative adversarial networks (GANs) are a recent unsupervised learning framework, which we discuss in Section 10.

GAIL finds a policy $\pi_{\boldsymbol{\theta}}$ so that a discriminator $\mathcal{D}_R$ can not distinguish states following the expert policy π_E and states following the imitator policy $\pi_{\boldsymbol{\theta}}$, hence forcing $\mathcal{D}_R$ to take 0.5 in all cases and $\pi_{\boldsymbol{\theta}}$ not distinguishable from π_E in the equillibrium. Such a game is formulated as:

$$\max_{\pi_{\boldsymbol{\theta}}} \min_{\mathcal{D}_R} -E_{\pi_{\boldsymbol{\theta}}}[\log \mathcal{D}_R(s)] - E_{\pi_E}[\log(1 - \mathcal{D}_R(s))]$$

The authors represent both $\pi_{\boldsymbol{\theta}}$ and $\mathcal{D}_R$ as deep neural networks, and find an optimal solution by repeatedly performing gradient updates on each of them. $\mathcal{D}_R$ can be trained with supervised learning with a data set formed from traces from a current $\pi_{\boldsymbol{\theta}}$ and expert traces. For a fixed $\mathcal{D}_R$, an optimal $\pi_{\boldsymbol{\theta}}$ is sought. Hence it is a policy optimization problem, with $-\log \mathcal{D}_R(s)$ as the reward. The authors train $\pi_{\boldsymbol{\theta}}$ by trust region policy optimization (Schulman et al., 2015), and experiment on various physics-based control tasks with good performance.

Li et al. (2017) extend GAIL to InfoGAIL for not only imitating, but also learning interpretable representations of complex behaviours, by augmenting the objective with a mutual information term between latent variables and trajectories, i.e., the observed state-action pairs. InfoGAIL learns a policy in which more abstract, high level latent variables would control low level actions. The authors experiment on autonomous highway driving using TORCS driving simulator (Wymann et al., 2014). See the open source at `https://github.com/ermongroup/InfoGAIL`. Song et al. (2018) extend GAIL to the multi-agent setting.

Reward Manipulating

van Seijen et al. (2017) propose hybrid reward architecture (HRA) to tackle the issue that optimal value function may not be embedded in low dimensional representation, by decomposing reward function into components, and learning value functions for them separately. Each component may be embedded in a subset of all features, so its value function may be learned and represented in a low dimensional space relatively easily. HRA agents learn with sample trajectories using off-policy learning in parallel, similar to Horde (Sutton et al., 2011). Experiments on Atari game Ms. Pac-Man show above-human performance. See open source at `https://github.com/Maluuba/hra`.

6 MODEL

A model is an agent's representation of the environment, including the state transition model and the reward model. Usually we assume the reward model is known. We discuss how to handle unknown reward models in Chapter 5.

Model-free RL approaches handle unknown dynamical systems, which usually requires large number of samples. This may work well for problems with good simulators to sample data, e.g., Atari games and the game of Go. However, this may be costly or prohibitive for real physical systems.

Model-based RL approaches learn value function and/or policy in a data-efficient way, however, they may suffer from issues of model identification, so that the estimated models may not be accurate, and the performance is limited by the estimated model. Planning constructs a value function or a policy usually with a model.

Combining model-free RL with on-line planning can improve value function estimation. Sutton (1990) proposes Dyna to integrate learning and planning, by learning from both real experiences and simulated trajectories from a learned model.

Monte Carlo tree search (MCTS) (Browne et al., 2012; Gelly and Silver, 2007; Gelly et al., 2012) and upper confidence bounds (UCB) (Auer, 2002) applied to trees (UCT) (Kocsis and Szepesvári, 2006) are important techniques for planning. A typical MCTS builds a partial tree starting from the current state, in the following stages: 1) select a promising node to explore further, 2) expand a leaf node and collect statistics, 3) evaluate a leaf node, e.g., by a rollout policy, or some evaluation function, 4) backup evaluations to update the action values. An action is then selected. A prominent example is AlphaGo (Silver et al., 2016a; 2017) as we will discuss in Section 15.1. Sun et al. (2018) study dual policy iteration.

Several papers design new deep neural networks architectures for RL problems, e.g., value iteration networks (VIN), predictron, value prediction network (VPN), TreeQN and ATreeC, imagination-augmented agents (IA2), temporal difference models (TDMs), MCTSnets, which we will discuss below, as well as dueling network as we discuss in Chapter 3. VIN, predictron, and, MCTSnets follow the techniques of learning to learn, which we discuss in Chapter 14.

R-MAX (Brafman and Tennenholtz, 2002) and E^3 (Kearns and Singh, 2002) achieve guaranteed efficiency for tabular cases. Li et al. (2011) present the "know what it knows" framework. Deisenroth and Rasmussen (2011) present probabilistic inference for learning control (PILCO), and McAllister and Rasmussen (2017) extend PILCO to POMDPs. See papers about model predictive control (MPC) (Amos et al., 2018; Finn and Levine, 2015; Lenz et al., 2015). See more papers, e.g., Berkenkamp et al. (2017), Buckman et al. (2018), Chebotar et al. (2017), Chua et al. (2018), de Avila Belbute-Peres et al. (2018), Farahmand (2018), Ha and Schmidhuber (2018), Haber et al. (2018), Henaff et al. (2017), Watter et al. (2015). We will discuss guided policy search (GPS) (Levine et al., 2016), and, Chebotar et al. (2017) in Chapter 16

Sutton (2018) discusses that "planning with a learned model" means "Intelligence is just knowing a lot about the world, being able to use that knowledge flexibly to achieve goals", and, mentions that "planning $\approx$ reasoning $\approx$ thought", and "world model $\approx$ knowledge $\approx$ propositions $\approx$ facts". He quotes from Yann LeCun, "obstacles to AI: learning models of the world, learning to reason and plan", and "predictive learning $\approx$ unsupervised learning $\approx$ model-based RL". He also quotes from Yoshua Bengio's most important next step in AI, "learning how the world ticks", and "predictive, causal, explanatory models with latent variables ...". He lists the following as some answers to the problem of planning with a learned model: function approximation, off-policy learning, Dyna, linear Dyna, non-linear Dyna, GVFs, Horde, options, option models, prioritized sweeping, intrinsic motivation, curiosity, recognizers, predictive state representations, TD networks (Sutton and Tanner, 2004), TD networks with options (Sutton et al., 2005), and, propagation with valuableness. LeCun (2018) also talks about world model, highlighting the role of self-supervised learning.

Geffner (2018) discusses model-free learners and model-based solvers, and planners as particular solvers. Learners are able to infer behaviour and functions from experience and data, solvers are able to address well-defined but intractable models like classical planning and POMDPs, and, planners are particular solvers for models with goal-directed behaviours. Geffner (2018) makes connections between model-free learners vs. model-based solvers, and the two systems in current theories of

human mind (Kahneman, 2011): System 1 with a fast, opaque, and inflexible intuitive mind, vs. System 2 with a slow, transparent, and flexible analytical mind.

See Finn (2017) for a tutorial, and Silver (2015) and Levine (2018) for lectures, about model-based (deep) RL. See NIPS 2018 Workshop on Modeling the Physical World: Perception, Learning, and Control at `http://phys2018.csail.mit.edu`. We discuss Lake et al. (2016) in Section 8.3, and, scene understanding and physics model learning in Section 18.3.

MODEL-BASED RL

We discuss several recent papers about model-based RL. We may roughly have methods with RL flavour, e.g., Dyna, VPN, IA2, using TD methods; methods with optimal control flavour, e.g., GPS, NAF, using local models like linear quadratic regulator (LQR) and MPC; and, methods with physics flavour, e.g., those with physics models as discussed in Lake et al. (2016).

In policy gradient methods, the gradient can be estimated either by likelihood ratio method as in REINFORCE, or by value gradient methods with backpropagation as in DPG/DDPG. Value gradients methods are used for deterministic policies. Heess et al. (2015) propose stochastic value gradients (SVG) for stochastic policies, to combine advantages of model-free and model-based methods, and to avoid their shortcomings. SVG treats noise variables in Bellman equation as exogenous inputs, allowing direct differentiation w.r.t. states. This results in a spectrum of policy gradient algorithms, ranging from model-free ones with value functions, to model-based ones without value functions. SVG learns model, value function, and policy jointly, with neural networks trained using experience from interactions with the environment. SVG mitigates compounded model errors by computing value gradients with real trajectories from the environment rather than those from an inaccurate, estimated model. SVG uses models for computing policy gradient, but not for prediction. Heess et al. (2015) experiment with physics-based continuous control tasks in simulation.

Oh et al. (2017) propose value prediction network (VPN) to integrate model-free and model-based RL into a neural network. VPN learns a dynamic model to train abstract states to predict future values, rewards, and discount, rather than future observations as in typical model-based methods. The author propose to train VPN by TD search (Silver et al., 2012) and multi-step Q learning. In VPNs, values are predicted with Q-learning, rewards are predicted with supervised learning, and lookahead planning are performed for choosing actions and computing target Q-values. VPN is evaluate on a 2D navigation collect task and Atari games.

Pong et al. (2018) propose temporal difference models (TDMs) to combine benefits of model-free and model-based RL. TDMs are general value functions, as discussed in Section 3.3. TDMs can be trained with model-free off-policy learning, and be used for model-based control. TDM learning interpolates between model-free and model-based learning, seeking to achieve sample efficiency in model-based learning, and at the same time, avoiding model bias. Pong et al. (2018) evaluate TDMs on both simulated and real-world robot tasks.

Farquhar et al. (2018) propose TreeQN, using a differentiable, recursive tree structure neural network architecture to map the encoded state to the predicted action value Q function, for discrete actions. TreeQN uses such recursive model to refine the estimate of Q function, with the learned transition model, reward function, and value function, by tree transitioning, and value prediction & backup steps in the recursive tree structure neural network. TreeQN takes advantage of the prior knowledge that Q values are induced by MDPs. In contrast, DQN uses fully connected layers, not implementing such inductive bias. Farquhar et al. (2018) also propose ATreeC, an actor-critic variant. The authors evaluate TreeQN and ATreeC in a box-pushing environment and on Atari games.

Weber et al. (2017) propose imagination-augmented agents (IA2), a neural network architecture, to combine model-free and model-based RL. IA2 learns to augment model-free decisions by interpreting environment models.

Gu et al. (2016) propose normalized advantage functions (NAF) to enable experience replay with Q-learning for continuous task, and propose to refit local linear models iteratively. NAF extends Dyna to continuous tasks. The authors evaluate NAF on several simulated MuJoCo robotic tasks.

Hester and Stone (2017) propose variance-and-novelty-intrinsic-rewards algorithm (TEXPLORE-VANAIR), a model-based RL algorithm with intrinsic motivations. The authors study two intrinsic

motivations, one for model uncertainty, and another for acquiring novel experiences new to the model. The authors conduct empirical study on a simulated domain and a real world robot and show good results.

Leonetti et al. (2016) propose domain approximation for reinforcement learning (DARLING), taking advantage of both RL and planning, so that the agent can adapt to the environment, and the agent's behaviour is constrained to reasonable choices. The authors perform evaluation on a service robot for tasks in an office building.

PLANNING

We discuss several recent papers about planning.

Guez et al. (2018) propose to learn to search with MCTSnets, a neural network architecture that incorporates simulation-based search, using a vector embedding for expansion, evaluation, and backup. The authors propose to jointly optimize in MCTSnets a simulation policy for where to traverse in the simulation, an evaluation network for what to evaluate in the reached states, and a backup network for how to backup evaluations, end-to-end with a gradient-based approach. The authors experiment MCTSnets with a classical planning problem Sokoban.

Tamar et al. (2016) introduce value iteration networks (VIN), a fully differentiable CNN planning module to approximate the value iteration algorithm, to learn to plan, e.g, policies in RL. In contrast to conventional planning, VIN is model-free, where reward and transition probability are part of the neural network to be learned. VIN can be trained end-to-end with backpropagation. VIN can generalize in a diverse set of tasks: simple gridworlds, Mars Rover Navigation, continuous control and WebNav Challenge for Wikipedia links navigation (Nogueira and Cho, 2016). VIN plans via value iteration over the full state space, and with local transition dynamics for states, e.g., 2D domains, which limits applicability of VIN. Lee et al. (2018) propose gated path planning networks to extend VIN.

Silver et al. (2016b) propose the predictron to integrate learning and planning into one end-to-end training procedure with raw input in Markov reward process (MRP), which can be regarded as Markov decision process without actions. Predictron rolls multiple planning steps of an abstract model represented by an MRP for value prediction; in each step, predictron applies the model to an internal state, and generates a next state, reward, discount, and value estimate. The predictron focuses on evaluation tasks in uncontrolled environments; however, it can be used as Q-network, e.g., in DQN, for control tasks.

Silver et al. (2012) propose temporal difference search (TD search) to combine TD learning with simulation based search. TD search updates value function from simulated experience, and generalizes among states using value function approximation and bootstrapping. Xiao et al. (2018) propose memory-augmented MCTS. Srinivas et al. (2018) propose universal planning networks.

7 EXPLORATION VS. EXPLOITATION

A fundamental tradeoff in RL is between exploration of uncertain policies and exploitation of the current best policy. Online decision making faces a central issue: either exploiting the information collected so far to make the best decision, or exploring for more information. In sequential decision making, we may have to sacrifice short-term losses to achieve long-term gains. It is essential to collect enough information to make the best overall decisions.

There are several principles in trading off between exploration and exploitation, namely, naive methods, optimism in the face of uncertainty, including, optimistic initialization, upper confidence bounds, and, probability matching, and, information state search (Silver, 2015). These are developed in the settings of multi-armed bandit, but are applicable to RL problems.

The multi-armed bandit problem is classical for studying exploration and exploitation. It is defined by a tuple $< \mathcal{A}, \mathcal{R} >$, where $\mathcal{A}$ is a given set of arms, or actions, and $\mathcal{R}(r|a) = \mathbb{P}(r|a)$ is a probability distribution over rewards, unknown to the agent. At each step t, the agent selects an action $a_t \in \mathcal{A}$, receives a reward $r_t \sim \mathcal{R}(\cdot|a_t)$ from the environment, and the goal is to maximize the cumulative reward $\sum_{\tau=1}^{t} r_\tau$.

The action-value function is the expected reward for action a, $Q(a) = \mathbb{E}[r|a]$. The optimal value is $V^* = Q(a^*) = \max_{a \in \mathcal{A}} Q(a)$. The regret is one step loss,

$$l_t = \mathbb{E}[V^* - Q(a_t)]. \tag{46}$$

The total regret until time step t is then

$$L_t = \mathbb{E}[\sum_{\tau=1}^{t} V^* - Q(a_\tau)]. \tag{47}$$

The maximum cumulative reward is the minimum total regret.

Denote $N_t(a)$ as the expected number of selecting action a until time step t. The greedy algorithm selects the action with the highest value, $a_t^* = \arg\max_{a \in \mathcal{A}} \hat{Q}_t(a)$, where $\hat{Q}_t(a)$ is an estimate of $Q(a)$, e.g., by Monte Carlo evaluation,

$$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{\tau=1}^{t} r_\tau 1(a_\tau = a). \tag{48}$$

The greedy algorithm may stick to a suboptimal action. However, ϵ-greedy, where $\epsilon \in (0, 1)$, can ensure a minimum regret with a constant ϵ. In ϵ-greedy, an agent selects a greedy action $a = \arg\max_{a \in \mathcal{A}} \hat{Q}(a)$, with probability $1 - \epsilon$; and selects a random action with probability ϵ.

A simple and practical idea for optimistic initialization is to initialize action value $Q(a)$ to a high value, then update it with incremental Monte Carlo evaluation,

$$\hat{Q}_t(a) = \hat{Q}_{t-1}(a) + \frac{1}{N_t(a)}(r_t - \hat{Q}_{t-1}(a)). \tag{49}$$

This encourages exploration in an early stage, but may stick to a suboptimal action.

Next we discuss upper confidence bounds (UCB) (Auer, 2002), an important result in bandit problems. Its extension, UCT for search trees (Kocsis and Szepesvári, 2006), in particular, in Monte Carlo tree search (MCTS) (Browne et al., 2012; Gelly and Silver, 2007; Gelly et al., 2012), plays important roles in many problems, including AlphaGo (Silver et al., 2016a; 2017).

We estimate an upper confidence $\hat{U}_t(a)$ for each action, to have $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with a high probability. When $N_t(a)$ is small, $\hat{U}_t(a)$ is large, i.e., the estimated value is uncertain; when $N_t(a)$ is large, $\hat{U}_t(a)$ is small, i.e., the estimated value is close to true value. We want to select an action to maximize the upper confidence bound,

$$a_t = \max_{a \in \mathcal{A}} \hat{Q}_t(a) + \hat{U}_t(a). \tag{50}$$

We need to establish a theoretical guarantee with Hoeffding's inequality theorem, which is: let $X_1, \ldots, X_t$ be i.i.d. random variables in [0,1], and let $\bar{X}_t = \frac{1}{\tau}\sum_{\tau=1}^{t} X_\tau$ be the sample mean. Then,

$$\mathbb{P}[\mathbb{E}[X] > \bar{X}_t] \leq e^{-2tu^2}. \tag{51}$$

With Hoeffding's inequality, conditioned on selecting action a, we have,

$$\mathbb{P}[Q(a) > \hat{Q}_t(a) + \hat{U}_t(a)] < e^{-2N_t(a)U_t(a)^2}. \tag{52}$$

Choose a probability $p = e^{-2N_t(a)U_t(a)^2}$, so that $Q(a) > \hat{Q}_t(a) + \hat{U}_t(a)$, i.e., the true value exceeds UCB, hence, $U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}$. Choose a schedule for p as observing more rewards, e.g., $p = t^{-4}$, hence, $U_t(a) = \sqrt{\frac{2\log p}{N_t(a)}}$. This guarantees that, as $t \to \infty$, we select optimal actions. Thus, we obtain the UCB1 algorithm,

$$a_t = \arg\max_{a \in \mathcal{A}} Q(a) + \sqrt{\frac{2\log t}{N_t(a)}}. \tag{53}$$

The UCB algorithm can achieve logarithmic asymptotic total regret, better than linear total regret achievable by ϵ-greedy and optimistic initialization (Auer, 2002).

As shown above, UCB employs $\sqrt{2\log t/N_t(a)}$ as exploration bonus to encourage less discovered actions. Model-based interval estimation with exploration bonuses (MBIE-EB) (Strehl and Littman, 2008) employs $\sqrt{1/N_t(a)}$; and Bayesian exploration bonus (BEB) (Kolter and Ng, 2009) employs $1/N_t(a)$.

In probability matching, we select an action a according to the probability that a is the optimal action with the largest value. It is optimistic under uncertainty, and uncertain actions tend to have higher probabilities of having the largest value. Thompson sampling implements probability matching, dealing with the difficulty of analytical computation with posterior distributions. Thompson sampling uses Bayes law to compute the posterior distribution, samples a reward distribution from the posterior, evaluates action value function, and, selects the action that maximizes value estimates on samples.

Gittins indices are a Bayesian model-based RL method for solving information state space bandits. It is known as Bayes-adaptive RL, and finds Bayes-optimal exploration and exploitation trade off w.r.t. a prior distribution. The computation complexity may be prohibitive for large problems.

The above discussions are in the setting of multi-armed bandits. They do not pay attention to additional information. However, such feature-based exploration vs exploitation problems are challenging (Auer et al., 2002; Langford and Zhang, 2007). Li et al. (2010) introduce contextual bandit and design algorithms based on UCB.

The techniques for multi-armed bandits are also applicable for full RL problems. A naive exploration technique, ϵ-greedy is widely used. In model-free RL, we can initialize action value function $Q(s,a) \leftarrow \frac{r_{max}}{1-\gamma}$, where r_{max} is the maximal value of reward. Brafman and Tennenholtz (2002) present R-MAX, a model-based RL, optimistically initializing all actions in all states to the maximal reward. UCB can work with model-free and model-based RL methods. Lipton et al. (2018) propose to add variance information to DQN, which is then used to guide exploration following the principle of Thompson sampling. Guez et al. (2014) propose a simulation-based search approach for Bayes-adaptive MDPs augmented with information states.

A RL agent usually uses exploration to reduce its uncertainty about the reward function and transition probabilities of the environment. In tabular cases, this uncertainty can be quantified as confidence intervals or posterior of environment parameters, which are related to the state-action visit counts. An example is MBIE-EB (Strehl and Littman, 2008). Brafman and Tennenholtz (2002), Jaksch et al. (2010), and Strehl and Littman (2008) provide theoretical guarantee for tabular cases.

Intrinsic motivation (Barto, 2013; Schmidhuber, 2010; Oudeyer and Kaplan, 2007) suggests to explore based on the concepts of curiosity and surprise, so that actions will transition to surprising states that may cause large updates to the environment dynamics models. One particular example of measuring surprise is by the change in prediction error in learning process (Schmidhuber,

1991), as studied recently in Bellemare et al. (2016). Intrinsic motivation methods do not require Markov property and tabular representation as count-based methods. Watch a video (Barto, 2017). See ICML 2018 workshop on Exploration in RL at, https://sites.google.com/view/erl-2018/home, and https://goo.gl/yxf16n for videos.

Levine (2018) discusses three classes of exploration methods in deep RL: 1) optimistic exploration methods, which estimate state visitation frequencies or novelty, typically with exploration bonuses, e.g., Bellemare et al. (2016), Fu et al. (2017), Schmidhuber (1991), and Tang et al. (2017); 2) Thompson sampling methods, which learn distribution over Q-functions or policies, then act according to samples, e.g., Osband et al. (2016); 3) information gain methods, which reason about information gain from visiting new states, e.g., Houthooft et al. (2016). All these three methods follow the principle of optimism in the face of uncertainty.

In the above, we discuss background of exploration and exploitation largely based on Lecture 9: Exploration and Exploitation in Silver (2015), as well as, the lecture about exploration in Levine (2018), Chapter 2 in (Sutton and Barto, 2018) about multi-armed bandits, and, relevant papers. Lattimore and Szepesvári (2018) is about bandit algorithms. Li (2012) surveys theoretical approaches to exploration efficiency in RL.

In the following we discuss several recent work about exploration in the setting of large scale RL problems, in particular deep RL.

COUNT-BASED METHODS

With the count-based exploration, a RL agent uses visit counts to guide its behaviour to reduce uncertainty. However, count-based methods are not directly useful in large domains. Bellemare et al. (2016) propose pseudo-count, a density model over the state space, for exploration with function approximation, to unify count-based exploration and intrinsic motivation, by introducing information gain, to relate to confidence intervals in count-based exploration, and to relate learning progress in intrinsic motivation. The authors establish pseudo-count's theoretical advantage over previous intrinsic motivation methods, implement it with a density model, use it as exploration bonuses in MBIE-EB (Strehl and Littman, 2008), in experience replay and actor-critic settings, and study its empirical performance with Atari games.

Ostrovski et al. (2017) further study the approach of pseudo-count (Bellemare et al., 2016) w.r.t. importance of density model selection, modelling assumptions, and role of mixed Monte Carlo update, and propose to use a neural density model for images, PixelCNN (van den Oord et al., 2016), for supplying pseudo-count, and combine it with various agent architectures, including DQN (Mnih et al., 2015) and Reactor (Gruslys et al., 2017). The authors observe that mixed Monte Carlo update facilitate exploration in settings with sparse rewards, like in the game of Montezuma's Revenge.

Tang et al. (2017) propose to implement the count-based exploration method by mapping states to hash codes to count occurrences with a hash table, and the counts are used for reward bonus to guide exploration. The authors experiment with simple hash functions and a learned domain-dependent hash code on both Atari games and continuous control tasks.

INTRINSIC MOTIVATION

Houthooft et al. (2016) propose variational information maximizing exploration (VIME), a curiosity-driven exploration approach to simultaneously optimizing both external reward and intrinsic surprise, for continuous state and action spaces. The authors propose to measure the information gain with variance inference, and to approximate the posterior distribution of an agent's internal belief of environment dynamics, represented with Bayesian neural networks. The authors evaluate VIME with various continuous control tasks and algorithms.

Pathak et al. (2017) study curiosity-driven exploration with intrinsic reward signal for predicting the result of its actions with self-supervised learning, and experiment with VizDoom and Super Mario Bros.

MORE WORK

Osband et al. (2016) propose bootstrapped DQN to combine deep exploration with deep neural networks to achieve efficient learning. The authors use randomized value functions to implement Thompson sampling, to enable exploration with non-linear function approximation, such as deep neural networks, and a policy is selected randomly according to its probability being optimal. The authors implement bootstrapped DQN by building multiple estimates of the action value function in parallel, and each estimate is trained with its own target network. The authors evaluate the performance of bootstrapped DQN with Atari games.

Nachum et al. (2017) propose under-appreciated reward exploration (UREX) to avoid the ineffective, undirected exploration strategies of the reward landscape, as in ϵ-greedy and entropy regularization policy gradient, and to promote directed exploration of the regions, in which the log-probability of an action sequence under the current policy under-estimates the resulting reward. UREX results from importance sampling from the optimal policy, and combines a mode seeking and a mean seeking terms to tradeoff exploration and exploitation. The authors implement UREX with minor modifications to REINFORCE, and validate it, for the first time with a RL method, on several algorithmic tasks. UREX is an effort for symbolic deep RL.

Azar et al. (2017) study the problem of provably optimal exploration for finite horizon MDPs. Fu et al. (2017) propose novelty detection with discriminative modeling for exploration. Fortunato et al. (2018) propose NoisyNet for efficient exploration by adding parametric noises to weights of deep neural networks. Jiang et al. (2017) study systematic exploration for contextual decision processes (CDPs). See also Dimakopoulou et al. (2018), Dong and Roy (2018), Gupta et al. (2018), Kumaraswamy et al. (2018), Madhavan et al. (2018), and, Osband et al. (2018).

Also note that maximum entropy RL helps exploration, as discussed in Section 4.4.

8 REPRESENTATION

Representation is fundamental to reinforcement learning, machine learning, and AI in general. For RL, it is relevant not only to function approximation for state/observation, action, value function, reward function, transition probability, but also to agent (Albrechta and Stone, 2018; Rabinowitz et al., 2018), environment, and any element in a RL problem. The "representation" in "representation learning" basically refers to the "feature" in "feature engineering". Representation learning is an approach to automatically find good features. Here we discuss "representation" in a broader perspective, i.e., about any element in a RL problem. Besides the "feature" for function approximation, we also refer "representation" to problem representation, like Markov decision process (MDP), partially observable Markov decision process (POMDP), and, predictive state representation (PSR), and, moreover, for representing knowledge, reasoning, causality, and human intelligence, either in function approximation, or in discovering new neural network architectures. We attempt to use such a notion of "representation" to unify the roles deep learning has played, is playing, and would play in various aspects of deep reinforcement learning.

When the problem is small, both state and action can be accommodated in a table, we can use a tabular representation. For large-scale problems, we need function approximation, to avoid curse of dimensionality. One approach is linear function approximation, using basis functions like polynomial bases, tile-coding, radial basis functions, Fourier basis, and proto-value functions (PVFs), etc. We also discuss representations for state distributions, in particular, successor representation, which is related to value function.

Recently, non-linear function approximations, in particular, deep neural networks, show exciting achievements. Common neural network structures include multiple layer perceptron (MLP), convolutional neural networks (CNNs), recurrent neural networks (RNNs), in particular long short time memory (LSTM) and gated recurrent unit (GRU), (variational) autoencoder, and capsules, etc. There are new neural network architectures customized for RL problems, e.g., value iteration networks (VIN), predictron, and value prediction networks (VPN), etc.

General value function (GVF) is an approach to learn, represent, and use knowledge of the world. Hierarchical representations, like options, feudal networks, and max-Q, etc. handle temporal abstraction. Relational RL integrates statistical relational learning and reasoning with RL to handle entities and relations.

There are renewed interests in deploying or designing networks for reasoning, including graph neural networks (GNN), graph networks (GN), relational networks (RNs), and compositional attention networks, etc. There are discussions about addressing issues of current machine learning with causality, and incorporating more human intelligence into artificial intelligence.

Although there have been enormous efforts for representation, since reinforcement learning is fundamentally different from supervised learning and unsupervised learning, an optimal representation for RL is probably different from generic CNN and RNN, thus it is desirable to search for an optimal representation for RL. Our hypothesis is that this would follow a holistic approach, by considering perception and control together, rather than treating them separately, e.g., by deciding a CNN to handle visual inputs, then fixing the network, and designing some procedure to find optimal weights for value function and/or policy. Learning to learn techniques as we discuss in Chapter 14 may play an important role here.

8.1 CLASSICS

In this section, we discuss classical methods for representation, as well as several papers for recent progress. We discuss (linear) function approximation, which is usually for value function and policy approximation. We then discuss representations for an RL problem description, i.e., state, transitions and reward, including, partially observable Markov decision process (POMDP), predictive state representation (PSR), and, contextual decision process (CDP). We also discuss successor representation for state visitation, and work for state-action distribution.

FUNCTION APPROXIMATION

Here we discuss linear function approximation. We discuss neural networks in Section 8.2. In linear function approximation, a value function is approximated by a linear combination of basis functions. Basis functions may take the form of polynomial bases, tile-coding, radial basis functions, Fourier basis, and proto-value functions, etc.

In tile coding, tiles partition the input space exhaustively, and each tile is a binary feature. A tiling is such a partition. Each tiling represents a feature. There are various ways to tile a space, like grid, log stripes, diagonal strips, and irregular, etc. (Sutton and Barto, 2018)

With radial basis functions (RBFs), typically, a feature i has a Gaussian response $\phi_s(i) = \exp\big(-\frac{||s-c_i||^2}{2\sigma_i^2}\big)$, where s is the state, c_i is the feature's prototypical or center state, and σ_i is the feature's width (Sutton and Barto, 2018). When using RBFs as features for a linear function approximator, we have an RBF network.

Mahadevan and Maggioni (2007) propose proto-value functions (PVFs), using "the eigenvectors of the graph Laplacian on an undirected graph formed from state transitions induced by the MDP". The authors then propose to learn PVFs and optimal policies jointly.

There are also papers with Gaussian processes (Rasmussen and Williams, 2006) and kernel methods (Schölkopf and Smola, 2001), e.g., Ghavamzadeh et al. (2016) and Ormoneit and Sen (2002).

RL PROBLEM DESCRIPTION

Partially observable Markov decision process (POMDP) (Kaelbling et al., 1998) generalizes MDP. In POMDP, an MDP determines system dynamics, with partial observability of underlying states. A POMDP maintains a probability distribution over possible states, based on observations, their probabilities, and the MDP. Hausknecht and Stone (2015) propose deep recurrent Q-learning for POMDP.

Predictive state representation (PSR) (Littman et al., 2001) utilizes vectors of predictions for action-observation sequences to represent states of dynamical systems. The predictions relate directly to observable quantities, rather than hidden states. PSRs do not have strong dependence on prior models as POMDP, and, PSRs are more compact than POMDP, in dynamic systems which linear PSRs can represent. In fact, PSRs are more general than nth-order Markov models, hidden Markov models (HMM), and POMDP (Singh et al., 2004). Recently, Downey et al. (2017) present predictive state RNNs, and Venkatraman et al. (2017) propose predictive state decoders, both of which combine PSRs with RNN to take their advantages.

Jiang et al. (2017) propose contextual decision processes (CDPs), RL problems with rich observations and function approximation, for systematic exploration. The authors introduce the Bellman rank, a complexity measure, and provide a unified framework for many problems in RL with low Bellman rank, e.g., tabular MDP, low-rank MDP, a POMDP with reactive value-functions, linear quadratic regulators (LQR), and reactive PSRs, and show that these problems are PAC-learnable (Valiant, 1984; Strehl et al., 2009; Li, 2012).

STATE AND STATE-ACTION DISTRIBUTION

Dayan (1993) introduces successor representation (SR),

$$\psi = \sum_{t=0}^{\infty} [\gamma \sum_{s'} \mathcal{P}(s'|s,\pi)]^t, \tag{54}$$

for expected discounted future state visitations, w.r.t. a given policy and a discount factor, and being independent of the reward. In the vector form,

$$\boldsymbol{\psi} = \sum_{t=0}^{\infty} (\gamma \boldsymbol{P})^t = (\boldsymbol{I} - \gamma \boldsymbol{P})^{-1}, \tag{55}$$

where $\boldsymbol{P}$ is the transition matrix, and $\boldsymbol{I}$ is the identity matrix. SR captures the dynamics of the MDP, describing where the agent will traverse in the future, independent of the reward. SR can be learned

with algorithms like TD learning. For value function, we have

$$v_\pi(s) = \mathbb{E}[R_t|s_t = s] = \mathbb{E}[r_{t+1}+\gamma R_{t+1}|s_t = s] = \mathbb{E}[r_{t+1}|s_t = s]+\gamma \sum_{s'} \mathcal{P}(s'|s, \pi)v_\pi(s'). \quad (56)$$

In vector form, we have

$$\boldsymbol{v} = \bar{\boldsymbol{r}} + \gamma \boldsymbol{P}\boldsymbol{v}, \text{so } \boldsymbol{v} = (\boldsymbol{I} - \gamma \boldsymbol{P})^{-1}\bar{\boldsymbol{r}}, \quad (57)$$

where $\bar{\boldsymbol{r}}$ is the average one-step reward from each state. Thus, we have, $\boldsymbol{v} = \boldsymbol{\psi}\bar{\boldsymbol{r}}$, decomposing the value function into environment dynamics (SR) and the reward signal. With SR, it is much easier to learn the value function. SR has wide applications in credit assignment, exploration, transfer learning, planning, imitation learning, and continual learning, etc. There are some recent papers about successor representation, e.g., Barreto et al. (2017), Kulkarni et al. (2016), Sherstan et al. (2018), and Zhang et al. (2017). See Gershman (2018) for a review.

Recently, Chen et al. (2018b) develop a bilinear representation to capture state-action distributions.

8.2 Neural Networks

In this section, we discuss representation learning, neural network architectures, challenges to CNN and RNN, memory, and a recently proposed grid-like representation. We discuss generative query network (GQN) for scene representation (Eslami et al., 2018) in Chapter 10. Deep learning, or deep neural networks, have been playing critical roles in many recent successes in AI.

Representation Learning

Representation learning is central to the success of deep learning. An ideal representation captures underlying disentangled, causal factors of data, and regularization strategies are necessary for generalization, following no free lunch theorem (Bengio et al., 2013; Goodfellow et al., 2016). We list these regularization strategies below. With *smoothness*, training data generalize to close neighbourhood in input space. *Linearity* assumes linear relationship, and may be orthogonal to smoothness. *Multiple explanatory factors* govern data generation, and motivate distributed representation, with separate directions corresponding to separate factors of variation. *Causal factors* imply that learned factors are causes of observed data, not vice versa. *Depth, or a hierarchical organization of explanatory factors* defines high level, abstract concepts with simple, hierarchical concepts. *Shared factors across tasks* enable sharing of statistical strength between tasks. *Manifolds* represent the concentration of probability mass with lower dimensionality than original space of data. *Natural clustering* identifies disconnected manifolds, each may contain a single class. *Temporal and spatial coherence* assumes that critical explanatory factors change more slowly than raw observations, thus easier to predict. *Sparsity* assumes that most inputs are described by only a few factors. And, *simplicity of factor dependencies* assumes simple dependancies among factors, e.g., marginal independence, linear dependency, or those in shallow autoencoders. Watch a talk Bengio (2018). See NIPS 2017 Workshop on Learning Disentangled Representations: from Perception to Control at `https://sites.google.com/view/disentanglenips2017`.

Neural Network Architectures

A CNN is a feedforward deep neural network, with convolutional layers, pooling layers, and fully connected layers. CNNs are designed to process data with multiple arrays, with locality and translation invariance as inductive bias (LeCun et al., 2015).

A RNN is built with a recurrent cell, and can be seen as a multilayer neural network with all layers sharing the same weights, with temporal invariance as inductive bias (LeCun et al., 2015). Long short term memory networks (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent unit (GRU) (Chung et al., 2014) are two popular RNNs, to address issues with gradient computation with long time steps.

Hinton and Salakhutdinov (2006) propose an autoencoder to reduce the dimensionality of data with neural networks. Sabour et al. (2017) and Hinton et al. (2018) propose capsules with dynamic routing, to parse the entire object into a parsing tree of capsules, each of which has a specific meaning.

There are new neural network architectures customized for RL problems, e.g., value iteration networks (VIN) (Tamar et al., 2016), predictron (Silver et al., 2016b), value prediction network (VPN) (Oh et al., 2017), imagination-augmented agents (IA2) (Weber et al., 2017), TreeQN and ATreeC (Farquhar et al., 2018), temporal difference models (TDMs) (Pong et al., 2018), MCTSnetsGuez et al. (2018), and BBQ-Networks (Lipton et al., 2018). We discuss RL with models in Chapter 6.

CHALLENGES TO CNN AND RNN

Some recent papers challenge if RNNs are a natural choice for sequence modelling. Bai et al. (2018) show empirically that CNNs outperform RNNs over a wide range of tasks. See the open source at `https://github.com/locuslab/TCN`. Miller and Hardt (2018) show that feed-forward networks approximate stable RNNs well, for both learning and inference with gradient descent, and validate theoretical results with experiments. Vaswani et al. (2017) propose a self-attention mechanism to replace recurrent and convolutional layers, for sequence transduction problems, like language modelling and machine translation.

MEMORY

Memory provides long term data storage. LSTM is a classical approach for equipping a neural network with memory, and its memory is for both storage and computation. Weston et al. (2015) propose memory networks to combine inference with a long-term memory. Graves et al. (2016) propose differentiable neural computer (DNC) to solve complex, structured problems. Wayne et al. (2018) propose memory, RL, and inference network (MERLIN) to deal with partial observability. We discuss attention and memory including above papers in Chapter 9. Below we discuss briefly neural networks equipped with memory to facilitate reasoning, e.g., relational memory core (RMC) (Santoro et al., 2018), and, compositional attention networks (Hutson, 2018).

GRID-LIKE REPRESENTATION

Banino et al. (2018) study vector-based navigation with grid-like representations. In a process of vector-based navigation, i.e., planning direct trajectories to goals, animals travel to a remembered goal, following direct routes by calculating goal-directed vectors with a Euclidean spatial metric provided by grid cells. Grid cells are also important for integrating self-motion, i.e., path integration. The authors study path integration with a recurrent network, and find emergent grid-like representations, which help improve performance of navigation with deep RL in challenging environments, and also help with mammal-like shortcut behaviors. Cueva and Wei (2018) is a concurrent work.

CNNs are popular neural networks for image processing, and induce the representation to achieve excellent results. CNNs were inspired from visual cortex. A popular representation in NLP is word2vec (Mikolov et al., 2013; Mikolov et al., 2017), which is influenced by linguistics, e.g., quoting John Rupert Firth, "You shall know a word by the company it keeps." The grid cell representation, with origin from the brain, boosts performance for navigation tasks.

8.3 KNOWLEDGE AND REASONING

Knowledge and reasoning (Brachman and Levesque, 2004; Russell and Norvig, 2009) are fundamental issues in AI. It is thus important to investigate issues about them, e.g., how to represent knowledge, like the predictive approach with general value function (GVF), or a symbolic approach with entities, properties and relations, how to incorporate knowledge in the learning system, like an inductive bias, in particular, using a knowledge base to improve learning tasks (Chen et al., 2018a; Yang and Mitchell, 2017), and how to design network architectures to help with reasoning, etc.

Bottou (2014) discuss machine learning and machine reasoning, and propose to define reasoning as the manipulation of knowledge previously acquired to answer a new question, to cover first-order logical inference, probabilistic inference, and components in a machine learning pipeline. Evans and Grefenstette (2018) propose a differentiable inductive logic framework to deal with inductive logic programming (ILP) problems with noisy data. Besold et al. (2017) discuss neural-symbolic learning and reasoning. There are books about causality (Pearl, 2009; Pearl et al., 2016; Pearl and

Mackenzie, 2018; Peters et al., 2017). Guo et al. (2018) present a survey of learning causality with data.

See NIPS 2018 Workshop on Causal Learning. See NIPS 2018 Workshop on Relational Representation Learning at https://r2learning.github.io. See NIPS 2017 Workshop on Causal Inference and Machine Learning for Intelligent Decision Making at https://sites.google.com/view/causalnips2017. See 2015 AAAI Spring Symposium Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches at https://sites.google.com/site/krr2015/.

We discuss general value function, hierarchical RL, and relational RL. We also discuss very briefly several topics, including causality, reasoning facilitated by neural networks, and incorporating human intelligence.

There are recent papers about neural approaches for algorithm induction, e.g., Balog et al. (2017); Graves et al. (2016); Liang et al. (2017a); Nachum et al. (2017); Reed and de Freitas (2016); Vinyals et al. (2015); Zaremba and Sutskever (2015).

General Value Function (GVF)

A key problem in AI is to learn, represent, and use knowledge of the world. Sutton et al. (2011) discuss that high-level representations based on first-order logic and Bayesian networks are expressive, but it is difficult to learn the knowledge and it is expensive to use such knowledge; and low-level representations like differential equations and state-transition matrices, can be learned from unsupervised data, but such representations are less expressive. The authors further discuss that value functions provide semantics for predictive knowledge and goal-oriented (control) knowledge.

Sutton et al. (2011) propose to represent knowledge with General Value Function (GVF), where policy, termination function, reward function, and terminal reward function are parameters. Schaul et al. (2015) propose Universal Value Function Approximators (UVFAs) to generalize over both states and goals. Andrychowicz et al. (2017) propose Hindsight Experience Replay (HER) to combat with the issue of sparse reward, following the idea of GVF. We discuss GVF in Section 3.3.

Hierarchical RL

Hierarchical RL (Barto and Mahadevan, 2003) is a way to learn, plan, and represent knowledge with temporal abstraction at multiple levels, with a long history, e.g., options (Sutton et al., 1999) and MAXQ (Dietterich, 2000). Hierarchical RL explores in the space of high-level goals to address issues of sparse rewards and/or long horizons. Hierarchical RL may be helpful for transfer and multi-task learning, which we discuss in Section 14.2. Hierarchical planning is a classical topic in AI (Russell and Norvig, 2009). There are some recent papers, like, hierarchical-DQN (Kulkarni et al., 2016), strategic attentive writer (Vezhnevets et al., 2016), feudal network (Vezhnevets et al., 2017), option-critic (Bacon et al., 2017), option discovery with a Laplacian framework (Machado et al., 2017), and, stochastic neural networks (Florensa et al., 2017). We discuss hierarchical RL in Chapter 11.

Relational RL

Statistical relational learning and reasoning studies uncertain relations, and manipulates structured representations of entities and their relations, with rules about how to compose them (Battaglia et al., 2018; Getoor and Taskar, 2007). Inductive logic programming (ILP) learns uncertain logic rules from positive and negative examples, entailing positive examples but not negative ones. Probabilistic ILP (Raedt et al., 2008; Manhaeve et al., 2018) is closely related to statistical relational learning. Probabilistic ILP integrates rule-based learning with statistical learning, and tackles the high complexity of ILP. Graphical models (Koller and Friedman, 2009) are important approaches for statistical relational learning.

Artificial neural networks have alternative names, including connectionism, parallel distributed processing, and neural computation (Russell and Norvig, 2009). Symbolism is about a formal language with symbols and rules, defined by mathematics and logic. Relational learning and reasoning with neural networks is an approach integrating connectionism and symbolism.

Relational RL integrates RL with statistical relational learning, and connects RL with classical AI, for knowledge representation and reasoning. Relational RL is not new. Džeroski et al. (2001) propose relational RL. Tadepalli et al. (2004) survey relational RL. Guestrin et al. (2003) introduce relational MDPs. Diuk et al. (2008) introduce objected-oriented MDPs (OO-MDPs). Recently, Battaglia et al. (2018) propose graph network (GN) to incorporate relational inductive bias, Zambaldi et al. (2018) propose deep relational RL, Keramati et al. (2018) propose strategic object oriented RL, and there are also deep learning approaches to deal with relations and/or reasoning, e.g., Battaglia et al. (2016), Chen et al. (2018a), Hutson (2018), Santoro et al. (2017), and Santoro et al. (2018). We discuss relational RL in Chapter 13.

CAUSALITY

Pearl (2018) discusses that there are three fundamental obstacles for current machine learning systems to exhibit human-level intelligence: adaptability or robustness, explainability, and understanding of cause-effect connections. The author describes a three layer causal hierarchy: association, intervention, and counterfactual. Association invokes statistical relationships, with typical questions like "What is?" and "How would seeing X change my belief in Y". Intervention considers not only seeing what is, but also changing what we see, with typical questions like "What if?" and "What if I do X?". Counterfactual requires imagination and retrospection, with typical questions like "Why?" and "What if I had acted differently?". Counterfactuals subsume interventional and associational questions, and interventional questions subsume associational questions.

Pearl (2018) proposes structural causal model, which can accomplish seven pillar tasks in automated reasoning: 1) encoding causal assumptions - transparency and testability, 2) do-calculus and the control of counfounding, 3) the algorithmization of counterfactuals, 4) mediation analysis and the assessment of direct and indirect effects, 5) adaptability, external validity and sample selection bias, 6) missing data, and, 7) causal discovery.

See some recent papers using deep learning to treat causality, e.g., Johansson et al. (2016), Hartford et al. (2017), and Lopez-Paz et al. (2017). Lattimore et al. (2016) discuss causal bandits. Tamar et al. (2018) discuss learning plannable representations with causal InfoGAN. Liu et al. (2018d) study off-policy policy evaluation inspired by causal reasoning.

REASONING

Battaglia et al. (2018) propose graph network (GN) to incorporate relational inductive bias, to attempt to achieve combinatorial generalization. GN generalizes graph neural network (GNN), e.g., Scarselli et al. (2009). Santoro et al. (2017) propose relation networks (RNs) for relational reasoning. Santoro et al. (2018) propose a relational memory core (RMC) with self-attention to handle tasks with relational reasoning. Hudson and Manning (2018) propose memory, attention, and control (MAC) recurrent cell for reasoning. Yi et al. (2018) discuss disentangling reasoning from vision and language understanding. We discuss relational RL in Chapter 13.

HUMAN INTELLIGENCE

Lake et al. (2016) discuss that we should build machines towards human-like learning and thinking. In particular, we should build causal world models, to support understanding and explanation, seeing entities rather than just raw inputs or features, rather than just pattern recognition; we should support and enrich the learned knowledge grounding in intuitive physics and intuitive psychology; we should represent, acquire, and generalize knowledge, leveraging compositionality and learning to learn, rapidly adapt to new tasks and scenarios, recombining representations, without retraining from scratch.

Lake et al. (2016) discuss that the following are key ingredients to achieve human-like learning and thinking: a) developmental start-up software, or cognitive capabilities in early development, including, a.1) intuitive physics, and, a.2) intuitive psychology; b) learning as rapid model building, including, b.1) compositionality, b.2) causality, and, b.3) learning to learn; c) thinking fast, including, c.1) approximate inference in structured models, and, c.2) model-based and model-free reinforcement learning. Watch a video Tenenbaum (2018).

We explain some of these key gradients by quoting directly from Lake et al. (2016). Intuitive physics refers to that "Infants have primitive object concepts that allow them to track objects over time and to discount physically implausible trajectories". Intuitive psychology refers to that "Infants understand that other people have mental states like goals and beliefs, and this understanding strongly constrains their learning and predictions". For causality: "In concept learning and scene understanding, causal models represent hypothetical real-world processes that produce the perceptual observations. In control and reinforcement learning, causal models represent the structure of the environment, such as modeling state-to-state transitions or action/state-to-state transitions."

Botvinick et al. (2017) discuss about one additional ingredient, autonomy, so that agents can build and exploit their own internal models, with minimal human manual engineering.

PART II: IMPORTANT MECHANISMS

In this part, we discuss important mechanisms for the development of (deep) reinforcement learning, including attention and memory in Chapter 9, unsupervised learning in Chapter 10, hierarchical RL in Chapter 11, relational RL in Chapter 13, multi-agent RL in Chapter 12, and, learning to learn in Chapter 14.

Note that we do not discuss some mechanisms, like Bayesian RL (Ghavamzadeh et al., 2015), and semi-supervised RL (Audiffren et al., 2015; Cheng et al., 2016; Dai et al., 2017; Finn et al., 2017b; Kingma et al., 2014; Papernot et al., 2017; Yang et al., 2017; Zhu and Goldberg, 2009).

9 ATTENTION AND MEMORY

Attention is a mechanism to focus on the salient parts. Memory provides long term data storage. Attention can be an approach for memory addressing.

A soft attention mechanism, e.g., Bahdanau et al. (2015), utilizes a weighted addressing scheme to all memory locations, or a distribution over memory locations, can be trained with backpropagation. A hard attention mechanism, e.g., Zaremba and Sutskever (2015), utilizes a pointer to address a memory location, following the way conventional computers accessing memory, and can be trained with reinforcement learning, in particular, policy gradient. Attention can help with visualization about where a model is attending to, e.g., in machine translation and image captioning. Most papers follow a soft attention mechanism. There are endeavours for hard attention (Liang et al., 2017a; Malinowski et al., 2018; Xu et al., 2015; Zaremba and Sutskever, 2015).

See Olah and Carter (2016) and Britz (2016) for discussions about attention and memory; the former discusses neural Turing machine (Graves et al., 2014) etc., and the latter discusses sequence-to-sequence model (Bahdanau et al., 2015), etc.

In the following, we discuss several papers about attention and/or memory.

See also Ba et al. (2014; 2016); Danihelka et al. (2016); Duan et al. (2017); Eslami et al. (2016); Gregor et al. (2015); Jaderberg et al. (2015); Kaiser and Bengio (2016); Kadlec et al. (2016); Oh et al. (2016); Oquab et al. (2015); Yang et al. (2016); Zagoruyko and Komodakis (2017); Zaremba and Sutskever (2015).

ATTENTION

Cho et al. (2014) and Sutskever et al. (2014) propose the sequence to sequence approach by using two RNNs to encode a sentence to a fix-length vector and then decode the vector into a target sentence. To address the issues with encoding the whole sentence into a fix-length vector in the basic sequence to sequence approach, Bahdanau et al. (2015) introduce a soft-attention technique, i.e., weighted sum of annotations to which an encoder maps the source sentence, to learn to jointly align and translate, by soft-searching for most relevant parts of the source sentence, and predicting a target word with these parts and previously generated target words.

Mnih et al. (2014) introduce the recurrent attention model (RAM) to focus on selected sequence of regions or locations from an image or video for image classification and object detection, to reduce computational cost for handling large video or images. The authors utilize REINFORCE to train the model, to overcome the issue that the model is non-differentiable, and experiment on an image classification task and a dynamic visual control problem.

Xu et al. (2015) integrate attention to image captioning, inspired by the papers in neural machine translation (Bahdanau et al., 2015) and object recognition (Mnih et al., 2014; Ba et al., 2014). The authors utilize a CNN to encode the image, and an LSTM with attention to generate a caption. The authors propose a soft deterministic attention mechanism and a hard stochastic attention mechanism. The authors show the effectiveness of attention with caption generation tasks on Flickr8k, Flickr30k, and MS COCO datasets.

Vaswani et al. (2017) propose Transformer, using self-attention to replace recurrent and convolutional layers, for sequence transduction problems, like language modelling and machine translation. Transformer utilizes a scaled dot-product attention, to map a query and key-value pairs to an output, and computes a set of queries as matrices simultaneously to improve efficiency. Transformer further implements a multi-head attention by transforming queries, keys, and values with different, learned linear projections respectively, performing the attention function in parallel, then concatenating results and yielding final values. Transformer follows the encoder-decoder architecture. The encoder is composed of a stack of six identical layers, with two sub-layers, a multi-head self-attention mechanism, then, a position-wise fully connected feed-forward network, with residual connection around each sub-layer, followed by layer normalization. The decoder is the same as the encoder, with an additional multi-head attention sub-layer between the two sub-layers, which takes inputs from the output of the encoder stack and the output from previous multi-head attention sub-layer. Transformer implements positional encoding to account for the order of the sequence. The authors evaluate Transformer on two machine translation tasks, achieve good results

w.r.t. BLEU score, and show that an attention mechanism has better time efficiency and is more parallelizable than recurrent models. Dehghani et al. (2018) extend Transformer. See open source at `https://github.com/tensorflow/tensor2tensor`, in particular, for Dehghani et al. (2018) at `https://goo.gl/72gvdq`. Tang et al. (2018b) study the hypothesis that self-attention and CNNs, rather than RNNs, can extract semantic features to improve long range dependences in texts with NLP tasks.

MEMORY

Weston et al. (2015) propose memory networks to combine inference with a long-term memory, which could be read from and written to, and train these two components to use them jointly. The authors present a specific implementation of the general framework on the task of question answering (QA), where the memory works as a dynamic knowledge base, and evaluate on a large-scale QA task and a smaller yet complex one.

Sukhbaatar et al. (2015) extend Weston et al. (2015) with a recurrent attention model over a large external memory, train in an end-to-end way, and experiment with question answering and language modelling tasks. See open source at `https://github.com/facebook/MemNN`.

Graves et al. (2016) propose differentiable neural computer (DNC), in which, a neural network can read from and write to an external memory, so that DNC can solve complex, structured problems, which a neural network without read-write memory can not solve. DNC minimizes memory allocation interference and enables long-term storage. Similar to a conventional computer, in a DNC, the neural network is the controller and the external memory is the random-access memory, and a DNC represents and manipulates complex data structures with the memory. Differently, a DNC learns such representation and manipulation end-to-end with gradient descent from data in a goal-directed manner. When trained with supervised learning, a DNC can solve synthetic question answering problems, for reasoning and inference in natural language. Moreover, it can solve the shortest path finding problem between two stops in transportation networks and the relationship inference problem in a family tree. When trained with reinforcement learning, a DNC can solve a moving blocks puzzle with changing goals specified by symbol sequences. DNC outperforms normal neural network like LSTM or DNC's precursor neural Turing machine (Graves et al., 2014). With harder problems, an LSTM may simply fail. Although these experiments are relatively small-scale, we expect to see further improvements and applications of DNC. See a blog at `https://deepmind.com/blog/differentiable-neural-computers/`.

Wayne et al. (2018) propose memory, RL, and inference network (MERLIN) to deal with partial observability, by equipping with extensive memory, and more importantly, formatting memory in the right way for storing right information trained with unsupervised predictive modelling. The author evaluate MERLIN on behavioural tasks in psychology and neurobiology, which may have high dimension sensory input and long duration of experiences.

10 UNSUPERVISED LEARNING

Unsupervised learning takes advantage of the massive amount of data without labels, and would be a critical mechanism to achieve artificial general intelligence.

Unsupervised learning is categorized into non-probabilistic models, like sparse coding, autoencoders, k-means etc, and probabilistic (generative) models, where density functions are concerned, either explicitly or implicitly. Among probabilistic (generative) models with explicit density functions, some are with tractable models, like fully observable belief nets, neural autoregressive distribution estimators, and PixelRNN, etc; some are with non-tractable models, like Botlzmann machines, variational autoencoders, Helmhotz machines, etc. For probabilistic (generative) models with implicit density functions, we have generative adversarial networks (GANs), moment matching networks, etc. See Salakhutdinov (2016) for more details.

LeCun (2018) summarizes the development of deep learning, and outlooks the future of AI, highlighting the role of world models and self-supervised learning. [5]

Self-supervised learning is a special type of unsupervised learning, in which, no labels are given; however, labels are created from the data. Unsupervised auxiliary learning (Jaderberg et al., 2017; Mirowski et al., 2017), GANs, and Aytar et al. (2018), as we discuss below, can be regarded as self-supervised learning. Pathak et al. (2017) propose curiosity-driven exploration by self-supervised prediction. Watch two talks, Efros (2017) and Gupta (2017), about self-supervised learning.

Goel et al. (2018) conduct unsupervised video object segmentation for deep RL. Mirowski et al. (2018) study learning to navigate in cities without a map. Hsu et al. (2018) study unsupervised learning with meta-learning. See also Artetxe et al. (2018), Le et al. (2012), Liu et al. (2017b), Nair et al. (2018), and van den Oord et al. (2018).

In the following, we discuss unsupervised auxiliary learning (Jaderberg et al., 2017; Mirowski et al., 2017), which, together with Horde (Sutton et al., 2011), are approaches to take advantages of possible non-reward training signals in environments. We also discuss generative adversarial networks (Goodfellow et al., 2014), generative query network (GQN) for scene representation (Eslami et al., 2018), and, playing hard exploration games by watching YouTube (Aytar et al., 2018).

UNSUPERVISED AUXILIARY LEARNING

Environments may contain abundant possible training signals, which may help to expedite achieving the main goal of maximizing the accumulative rewards, e.g., pixel changes may imply important events, and auxiliary reward tasks may help to achieve a good representation of rewarding states. This may be even helpful when the extrinsic rewards are rarely observed.

Jaderberg et al. (2017) propose unsupervised reinforcement and auxiliary learning (UNREAL) to improve learning efficiency by maximizing pseudo-reward functions, besides the usual cumulative reward, while sharing a common representation. UNREAL is composed of four components: base agent, pixel control, reward prediction, and value function replay. The base agent is a CNN-LSTM agent, and is trained on-policy with A3C (Mnih et al., 2016). Experiences of observations, rewards and actions are stored in a replay buffer, for being used by auxiliary tasks. The auxiliary policies use the base CNN and LSTM, together with a deconvolutional network, to maximize changes in pixel intensity of different regions of the input images. The reward prediction module predicts short-term extrinsic reward in next frame by observing the last three frames, to tackle the issue of reward sparsity. Value function replay further trains the value function. UNREAL

[5] LeCun (2018) uses the cake metaphor, as in his NIPS 2016 invited talk titled Predictive Learning. In this metaphor, "pure" reinforcement learning, as the single cherry on the cake, "predicts a scalar reward given once in a while", with very low feedback information content; supervised learning, as the icing of the cake, "predicts a category or a few numbers for each input", with medium feedback information content; and, self-supervised learning, as cake genoise, "predicts any part of its input for any observed part", or "predicts future frames in videos", with high but stochastic feedback information content ("self-supervised learning" replacing "unsupervised/predictive learning" in his NIPS 2016 talk). As one response from the RL community, Pieter Abbeel presents a cake with many cherries in Abbeel (2017a), as a metaphor that RL methods can also have high information content, e.g., Hindsight Experience Replay (HER) (Andrychowicz et al., 2017) and Universal Value Function Approximators (UVFAs) (Schaul et al., 2015).

has a shared representation among signals, while Horde trains each value function separately with distinct weights. The authors show that UNREAL improves A3C's performance on Atari games, and performs well on 3D Labyrinth game. See a blog at `https://deepmind.com/blog/reinforcement-learning-unsupervised-auxiliary-tasks/`.

Mirowski et al. (2017) obtain navigation ability by solving a RL problem maximizing cumulative reward and jointly considering unsupervised tasks to improve data efficiency and task performance. The authors address the sparse reward issues by augmenting the loss with two auxiliary tasks, 1) unsupervised reconstruction of a low-dimensional depth map for representation learning to aid obstacle avoidance and short-term trajectory planning; 2) self-supervised loop closure classification task within a local trajectory. The authors incorporate a stacked LSTM to use memory at different time scales for dynamic elements in the environments. The proposed agent learns to navigate in complex 3D mazes end-to-end from raw sensory inputs, and performs similarly to human level, even when start/goal locations change frequently. In this approach, navigation is a by-product of the goal-directed RL optimization problem, in contrast to conventional approaches such as simultaneous localization and mapping (SLAM), where explicit position inference and mapping are used for navigation.

GENERATIVE ADVERSARIAL NETS

Goodfellow et al. (2014) propose generative adversarial nets (GANs) to estimate generative models via an adversarial process by training two models simultaneously, a generative model G to capture the data distribution, and a discriminative model D to estimate the probability that a sample comes from the training data but not the generative model G.

Goodfellow et al. (2014) model G and D with multilayer perceptrons: $G(z : \boldsymbol{\theta}_g)$ and $D(x : \boldsymbol{\theta}_d)$, where $\boldsymbol{\theta}_g$ and $\boldsymbol{\theta}_d$ are parameters, x are data points, and z are input noise variables. Define a prior on input noise variable $p_z(z)$. G is a differentiable function and $D(x)$ outputs a scalar as the probability that x comes from the training data rather than p_g, the generative distribution we want to learn.

D will be trained to maximize the probability of assigning labels correctly to samples from both training data and G. Simultaneously, G will be trained to minimize such classification accuracy, $\log(1 - D(G(z)))$. As a result, D and G form the two-player minimax game as follows:

$$\min_G \max_D E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))]. \tag{58}$$

Goodfellow et al. (2014) show that as G and D are given enough capacity, generative adversarial nets can recover the data generating distribution, and provide a training algorithm with backpropagation by minibatch stochastic gradient descent.

GANs are notoriously hard to train. See Arjovsky et al. (2017) for Wasserstein GAN (WGAN) as a stable GANs model. Gulrajani et al. (2017) propose to improve stability of WGAN by penalizing the norm of the gradient of the discriminator w.r.t. its input, instead of clipping weights as in Arjovsky et al. (2017). Mao et al. (2016) propose Least Squares GANs (LSGANs), another stable model. Berthelot et al. (2017) propose BEGAN to improve WGAN by an equilibrium enforcing model, and set a new milestone in visual quality for image generation. Bellemare et al. (2017) propose Cramér GAN to satisfy three machine learning properties of probability divergences: sum invariance, scale sensitivity, and unbiased sample gradients. Hu et al. (2017) unified GANs and Variational Autoencoders (VAEs).

Lucic et al. (2018) conduct a large-scale empirical study on GANs models and evaluation measures, and observe that, by fine-tuning hyperparameters and random restarts, most models perform similarly. The authors propose more data sets which enable computing of precision and recall. The authors further observe that the evaluated models do not outperform the original GAN algorithm. Kurach et al. (2018) discuss the state of the art of GANs in a practical perspective. The authors reproduce representative algorithms, discuss common issues, open-source their code on Github, and provide pre-trained models on TensorFlow Hub. Brock et al. (2018) study image synthesis.

We discuss generative adversarial imitation learning (Ho and Ermon, 2016; Li et al., 2017) in Chapter 5. Finn et al. (2016a) establish connections between GANs, inverse RL, and energy-based models. Pfau and Vinyals (2016) establish connections between GANs and actor-critic algorithms.

See Goodfellow (2017) for summary of his NIPS 2016 Tutorial on GANs. GANs have received much attention and many work have been appearing after the publication of Goodfellow (2017). See CVPR 2018 tutorial on GANs at `https://sites.google.com/view/cvpr2018tutorialongans/`, with several talks by several speakers.

GENERATIVE QUERY NETWORK

Eslami et al. (2018) propose generative query network (GQN) for scene representation, to obtain a concise description of a 3D scene from 2D visual sensory data, with a representation network and a generator network trained jointly in an end-to-end fashion, without human semantic labels, e.g., object classes, object locations, scene types, or part labels, and domain knowledge.

In GQN, observations from different 2D viewpoints for 3D scenes are fed into the representation network, to obtain a neural scene representation by summing observations' representations element-wise. The neural scene representation is then fed into the generation network, which is a recurrent latent variable model, to make prediction about the scene from a different viewpoint. GQN is trained with many scenes, with various number of observations, and with back-propagation. GQN attempts to obtain a concise scene representation, to capture the scene contents, e.g., the identities, positions, colors, and object counts, etc., and to make a generator successful for predictions, by maximizing the likelihood to generate ground-truth images from query viewpoints. Variational approximations are used to deal with the intractability of latent variables.

Experiments show that representations learned by GQN with the properties of viewpoint invariance, compositionality, factorization, "scene algebra", similar to that of word embedding algebra, and decreasing Bayesian surprise with more observations for both full and partial observability. Bayesian surprise refers to the KL divergence between conditional prior and posterior. Robot arm reaching experiments show that the GQN representation helps with data efficiency and robust control.

PLAYING GAMES BY WATCHING YOUTUBE

Aytar et al. (2018) propose to play hard exploration Atari games, including Montezuma's Revenge, Pitfall! and Private Eye, by watching YouTube. YouTube videos are usually noisy and unaligned, without the frame-by-frame alignment between demonstrations, and the information of exact action and reward sequences in demonstrator's observation trajectory, which are the properties of demonstrations required by previous imitation learning, e.g., Hester et al. (2018) as we discuss in Chapter 5.

Aytar et al. (2018) overcome these limitations by a one-shot imitation method in two steps. First, a common representation is learned from unaligned videos from multiple sources, with two self-supervised objectives: temporal distance classification (TDC) and cross-model temporal distance classification (CMC). In self-supervision, an auxiliary task is proposed to solve among all domains simultaneously, for a network to attempt to learn a common representation. In TDC, temporal distances between two frames in a single video sequence are predicted, to help learn a representation of the environment dynamics. In CMC, a representation is learned to correlate visual and audio observations, and to highlight critical game events. Furthermore, a new measure of cycle-consistency is proposed to evaluate the quality of the learned representation. Second, a single YouTube video is embedded in such representation, and a reward function is built, so that an agent can learn to imitate human game play.

Experiments using the distributed A3C RL algorithm IMPALA (Espeholt et al., 2018) show breakthrough results on these three hard Atari games.

11 HIERARCHICAL RL

Hierarchical RL (Barto and Mahadevan, 2003) is a way to learn, plan, and represent knowledge with temporal abstraction at multiple levels, with a long history, e.g., options (Sutton et al., 1999), MAXQ (Dietterich, 2000), hierarchical abstract machines (Parr and Russell, 1998), and dynamic movement primitives (Schaal, 2006). Hierarchical RL is an approach for issues of sparse rewards and/or long horizons, with exploration in the space of high-level goals. The modular structure of hierarchical RL approaches is usually conducive to transfer and multi-task learning, which we discussed in Section 14.2. The concepts of sub-goal, option, skill, and, macro-action are related. Hierarchical planning is a classical topic in AI (Russell and Norvig, 2009).

Here we introduce options briefly. Markov decision process (MDP) is defined by the 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, for the state space, the action space, the state transition probability, the reward function, and the discount factor. An option o consists of three components: 1) an initiation set of states $\mathcal{I}_o \subseteq \mathcal{S}$, 2) a policy $\pi_o : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, guiding the behaviour of an option, such that $\pi_o(a|s)$ is the probability of taking action a in state s when following option o, and, 3) a termination condition $\beta_o : \mathcal{S} \rightarrow [0, 1]$, roughly determining the length of an option, such that $\beta_o(s)$ is the probability of terminating the option o upon entering state s. A policy-over-options calls an option o. During the execution of the option o, the agent selects an action until a termination condition is met. An option may call another option. $P(s'|s, o)$ is the probability of next state s' conditioned on that option o executes from state s. $r_o(s)$ is the expected return during the execution of option o. Introducing options over an MDP constitutes a semi-Markov decision process (SMDP). It can be shown that learning and planning algorithms from MDPs can transfer to options. It can be shown that the reward model for options is equivalent to a value function, and a Bellman equation can be written for it, so RL algorithms can be used to learn it. This also applies to the transition model for options.

Usually options are learned with provided sub-goals and pseudo-rewards, and good performance is shown for Atari games, e.g. with hierarchical-DQN (h-DQN) (Kulkarni et al., 2016), and for MineCraft, e.g., with Tessler et al. (2017). Automatic options discovery receives much attention recently, e.g., strategic attentive writer (STRAW) (Vezhnevets et al., 2016), feudal network (FuN) (Vezhnevets et al., 2017), option-critic (Bacon et al., 2017), option discovery with a Laplacian framework (Machado et al., 2017), and, stochastic neural networks (Florensa et al., 2017).

Hierarchical RL follows the general algorithm design principle of divide and conquer, so that hard goals, e.g. those with sparse long-term rewards are replaced with easy sub-goals, e.g. those with dense short-term rewards, and RL algorithms, e.g., policy-based or value-based, combined with representations, are utilized to solve easy sub-goals, and finally to achieve hard goals.

Watch recent talks on hierarchical RL, e.g., Silver (2017), Precup (2018). See NIPS 2017 workshop on Hierarchical RL, at `https://sites.google.com/view/hrlnips2017`, and videos at `https://goo.gl/h9Mz1a`.

We discuss several recent papers in the following. See also Kompella et al. (2017), Le et al. (2018), Nachum et al. (2018), Peng et al. (2017a), Sharma et al. (2017), Tang et al. (2018a), Tessler et al. (2017), and Yao et al. (2014).

HIERARCHICAL DQN

Kulkarni et al. (2016) propose hierarchical-DQN (h-DQN) by organizing goal-driven intrinsically motivated deep RL modules hierarchically to work at different time-scales. h-DQN integrats a top level action value function and a lower level action value function. The former learns a policy over intrinsic sub-goals, or options (Sutton et al., 1999). And the latter learns policies over raw actions to satisfy the objective of each given sub-goal. In particular, h-DQN has a two-stage hierarchy with a meta-controller and a controller. The meta-controller receives state s and select a goal g. The controller then selects an action a conditioned on state s and goal g, the goal g does not change until it is achieved or a termination condition is met. The internal critic evaluates if a goal has been achieved, and produces a reward $r(g)$ to the controller, e.g., a binary internal reward 1 for achieving the goal, and 0 otherwise. The objectives for the meta-controller and controller are to maximize cumulative extrinsic and intrinsic rewards, respectively. The authors evaluate h-DQN on a discrete

stochastic decision process, and a hard Atari game, Montezuma's Revenge. See the open source at https://github.com/mrkulk/hierarchical-deep-RL

FEUDAL NETWORKS

Vezhnevets et al. (2017) propose feudal networks (FuNs) for hierarchical RL, inspired by feudal RL (Dayan and Hinton, 1993). In FuNs, a Manager module discovers and sets abstract sub-goals and operates at a long time scale, and a Worker module selects atom actions at every time step of the environment to achieve the sub-goal set by the Manager. FuNs decouple end-to-end learning across multiple levels at different time scales, by separating the Manager module from the Worker module, to facilitate long time scale credit assignment and emergence of sub-policies for sub-goals set by the Manager. In FuNs, sub-goals are formulated as directions in the latent space, so that the Manager selects a subgoal direction to maximize reward, and the Worker selects actions to maximize cosine similarity to the direction of the subgoal set by the Manager. The authors utilize a dilated LSTM to design the Manager to allow backpropagation through long time steps, and experiment FuNs with a water maze and Atari games.

OPTION-CRITIC

Bacon et al. (2017) derive policy gradient theorems for options, and propose an option-critic architecture to learn both intra-option policies and termination conditions gradually, at the same time with the policy-over-options, combining options discovery with options learning. The option-critic architecture works with linear and non-linear function approximations, with discrete or continuous state and action spaces, and without rewards or sub-goals. The authors experiment with a four-room domain, a pinball domain, and Atari games. See the open source at https://github.com/jeanharb/option_critic.

Harutyunyan et al. (2018) study the dilemma between efficiency of long options and flexibility of short ones in the option-critic architecture. The authors decouple the behaviour and target termination conditions, similar to off-policy learning for policies, and propose to cast options learning as multi-step off-policy learning, The authors show benefits of learning short options from longer ones, by analysis and with experiments.

Riemer et al. (2018) further study the option-critic architecture.

OPTION DISCOVERY WITH A LAPLACIAN FRAMEWORK

Machado et al. (2017) propose to discover options with proto-value functions (PVFs) (Mahadevan and Maggioni, 2007), which are well-known for representation in MDPs and define options implicitly. The authors introduce the concepts of eigen-purpose and eigen-behavior. An eigen-purpose is an intrinsic reward function, to motivate an agent to traverse in principle directions of the learned representation of state space. An eigen-behavior is the optimal policy for an intrinsic reward function. The authors discover task-independent options, since the eigen-purposes are obtained without reward information. The authors observe that some options are not helpful for exploration, although they improve the efficiency of planning. The authors further show that the options they discover improve exploration, since these options operate at different time scales, and they can be sequenced easily. The authors experiment with tabular domains and Atari games.

STRATEGIC ATTENTIVE WRITER

Vezhnevets et al. (2016) propose strategic attentive writer (STRAW), a deep recurrent neural network architecture, for learning high-level temporally abstract macro-actions in an end-to-end manner based on observations from the environment. Macro-actions are sequences of actions commonly occurring. STRAW builds a multi-step action plan, updated periodically based on observing rewards, and learns for how long to commit to the plan by following it without replanning. STRAW learns to discover macro-actions automatically from data, in contrast to the manual approach in previous work. Vezhnevets et al. (2016) validate STRAW on next character prediction in text, 2D maze navigation, and Atari games.

STOCHASTIC NEURAL NETWORKS

Florensa et al. (2017) propose to pre-train a large span of skills using stochastic neural networks with an information-theoretic regularizer, then on top of these skills, to train high-level policies for downstream tasks. Pre-training is based on a proxy reward signal, which is a form of intrinsic motivation to explore agent's own capabilities, requiring minimal domain knowledge about the downstream tasks. Their method combines hierarchical methods with intrinsic motivation, and the pre-training follows an unsupervised way.

12 MULTI-AGENT RL

Multi-agent RL (MARL) is the integration of multi-agent systems (Horling and Lesser, 2004; Shoham and Leyton-Brown, 2009; Stone and Veloso, 2000) with RL. It is at the intersection of game theory (Leyton-Brown and Shoham, 2008) and RL/AI.

Besides issues in RL like sparse rewards and sample efficiency, there are new issues like multiple equilibria,[6] and even fundamental issues like what is the question for multi-agent learning, and whether convergence to an equilibrium is an appropriate goal, etc. Consequently, multi-agent learning is challenging both technically and conceptually, and demands clear understanding of the problem to be solved, the criteria for evaluation, and coherent research agendas (Shoham et al., 2007).

In a fully centralized approach, when global state and joint action information are available, estimating the joint Q action value function becomes possible. This can address the nonstationary issue. However, it encounters the issue of curse of dimensionality when the number of agents grows. Another issue is that it may be hard to extract decentralized policies, for an agent to make decisions based on its own observation.

Littman (1994) employ stochastic games as a framework for MARL, propose minimax-Q learning for zero-sum games, and show convergence under certain conditions. Hu and Wellman (2003) propose Nash Q-learning for general-sum games and show its convergence with a unique Nash equilibrium. Bowling and Veloso (2002) propose the win or learn fast (WoLF) principle to vary the learning rate to tackle the issues with learning a moving target, and show convergence with self-play in certain iterated matrix games. These papers, together with Foerster et al. (2018a), Lowe et al. (2017), and Usunier et al. (2017), follow centralized approaches.

Tan (1993) introduces independent Q-learning (IQL) for MARL, where each agent learns a Q action value function independently. For Q-learning to be stable and convergent, the environment would be stationary. This is usually not the case for multi-agent systems, where an agent would change its policy according to other agents, and the environment is usually nonstationary or even adversarial. Independent approaches to MARL may not converge. Foerster et al. (2017) and Omidshafiei et al. (2017) propose to stabilize independent approaches.

Oliehoek et al. (2008) introduce the paradigm of centralized training for decentralized execution. We discuss several papers following this scheme below.

Along with the success of RL in single agent problems, like, Mnih et al. (2015), Jaderberg et al. (2017), Schulman et al. (2015), Nachum et al. (2018), and two-player games, like Silver et al. (2016a; 2017); Moravčík et al. (2017), recently, we see some progress in multi-agent RL problems, like Jaderberg et al. (2018) for Quake III Arena Capture the Flag, Sun et al. (2018) and Pang et al. (2018) for StarCraft II, and OpenAI Five for Dota 2.

Zambaldi et al. (2018) investigate StarCraft II mini-games with relational deep RL, as discussed in Chapter 13. Bansal et al. (2018) investigate the emergent complex skills via multi-agent competition. Foerster et al. (2018b) propose learning with opponent-learning awareness, so that each agent considers the learning process of other agents in the environment. Hoshen (2017) present vertex attention interaction network (VAIN), for multi-agent predictive modelling, with an attentional neural network. Mhamdi et al. (2017) introduce dynamic safe interruptibility for MARL, in joint action learners and independent learners scenarios. Perolat et al. (2017) propose to use MARL for the common pool resource appropriation problem. Hu et al. (2018b) propose opponent-guided tactic learning for StarCraft micromanagement. Song et al. (2018) extend generative adversarial imitation learning (GAIL) (Ho and Ermon, 2016) to multi-agent settings. Lanctot et al. (2018) study actor-critic policy optimization in partially observable multi-agent settings. See also Hughes et al. (2018), Wai et al. (2018), and Zhou et al. (2018)

Multi-agent systems have many applications, e.g., as we will discuss, games in Chapter 15, robotics in Chapter 16, energy in Section 20.3, transportation in Section 20.4, and compute systems in Section 20.5.

[6] Chen and Deng (2006) show that finding a Nash equilibrium in a two-player game is PPAD-complete, i.e., unless every problem in PPAD is solvable in polynomial time, there is not a fully polynomial-time approximation algorithm for finding a Nash equilibrium in a two-player game. PPAD is a complexity class for polynomial parity arguments on directed graphs.

Busoniu et al. (2008) and Ghavamzadeh et al. (2006) are surveys for multi-agent RL. Parkes and Wellman (2015) is a survey about economic reasoning and AI.

In the following, we discuss centralized training for decentralized execution, several issues in game theory, and games.

Centralized Training for Decentralized Execution

A centralized critic can learn from all available state information conditioned on joint actions, and each agent learns its policy from its own observation action history. The centralized critic is only used during learning, and only the decentralized actor is needed during execution. In the following, several recently propose approaches use StarCraft as the experimental testbed, e.g., Peng et al. (2017b), Foerster et al. (2018a), and Rashid et al. (2018).

Foerster et al. (2018a) propose the counterfactual multi-agent (COMA) actor-critic method. In COMA, policies are optimized with decentralized actors, and Q-function is estimated with a centralized critic, using a counterfactual baseline to marginalize out one agent's action, and fixing other agents' actions, for the purpose of multi-agent credit assignment.

Some papers propose communication mechanisms in MARL (Foerster et al., 2016; Sukhbaatar et al., 2016). Peng et al. (2017b) require communication.

Peng et al. (2017b) propose a multiagent actor-critic framework, with a bidirectionally-coordinated network to form coordination among multiple agents in a team, deploying the concept of dynamic grouping and parameter sharing for better scalability. In the testbed of StarCraft, without human demonstration or labelled data as supervision, the proposed approach learns strategies for coordination similar to the level of experienced human players, like move without collision, hit and run, cover attack, and focus fire without overkill.

It is desirable to design an algorithm between the two extremes of independent RL and fully centralized RL approaches. One way is to decompose Q function. Sunehag et al. (2017) and Rashid et al. (2018) fall into this category.

Sunehag et al. (2017) propose value-decomposition networks (VDN) to represent the centralized action value function Q as a sum of value functions of individual agents. In VDN, each agent trains its value function based on its observations and actions, and a decentralized policy is derived from its action value function.

Rashid et al. (2018) propose QMIX, so that each agent network represents an individual action value function, and a mixing network combines them into a centralized action value function, with a non-linear approach, in contrast to the simple sum in VDN (Sunehag et al., 2017). Such factored representation allows complex centralized action value function, extraction of decentralized policies with linear time individual argmax operations, and scalability.

Issues in Game Theory

Heinrich and Silver (2016) propose neural fictitious self-play (NFSP) to combine fictitious self-play with deep RL to learn approximate Nash equilibria for games of imperfect information in a scalable end-to-end approach without prior domain knowledge. NFSP is evaluated on two-player zero-sum games. In Leduc poker, NFSP approaches a Nash equilibrium, while common RL methods diverges. In Limit Texas Hold'em, a real-world scale imperfect-information game, NFSP performs similarly to state-of-the-art, superhuman algorithms which are based on domain expertise.

Lanctot et al. (2017) observe that independent RL, in which each agent learns by interacting with the environment, oblivious to other agents, can overfit the learned policies to other agents' policies. The authors propose policy-space response oracle (PSRO), and its approximation, deep cognitive hierarchies (DCH), to compute best responses to a mixture of policies using deep RL, and to compute new meta-strategy distributions using empirical game-theoretic analysis. PSRO/DCH generalizes previous algorithms, like independent RL, iterative best response, double oracle, and fictitious play. The authors present an implementation with centralized training for decentralized execution, as discussed below. The authors experiment with grid world coordination, a partially observable game,

and Leduc Poker (with a six-card deck), a competitive imperfect information game, and show reduced joint-policy correlation (JPC), a new metric to quantify the effect of overfitting.

Social dilemmas, e.g., prisoner's dilemma, reveal the conflicts between collective and individual rationality. Cooperation is usually beneficial for all. However, parasitic behaviours like free-riding may result in the tragedy of the commons, which makes cooperation unstable. The formalism of matrix game social dilemmas (MGSD) is a popular approach. However, as discussed in Leibo et al. (2017), MGSD does not consider several important aspects of real world social dilemmas: they are temporally extended, "cooperation and defection are labels that apply to policies implementing strategic decisions", "cooperative may be a graded quantity", cooperation and defection may not happen fully simultaneously, since information about the starting of a strategy by one player would influence the other player, and, decisions are mandatory although with only partial information about the world and other players.

Leibo et al. (2017) propose a sequential social dilemma (SSD) with MARL to tackle these issues. The authors conduct empirical game theoretic analyses of two games, fruit gathering and wolfpack hunting, and show that, when treating cooperation and defection as one-shot decisions as in MGSD, they have empirical payoff matrices as prisoner's dilemma. However, gathering and wolfpack are two different games, with opposite behaviours in the emergence and stability of cooperation. SSDs can capture the differences between these two games, using a factor to promote cooperation in gathering and to discourage cooperation in wolfpack. The sequential structure of SSDs results in complex model to compute or to learn equilibria. The authors propose to apply DQN to find equilibria for SSDs.

GAMES

Jaderberg et al. (2018) approach Capture the Flag, a 3D multi-player first-person video game, in an end-to-end manner using raw inputs including pixels and game points, with techniques of population based training, optimization of internal reward, and temporally hierarchical RL, and achieve human-level performance, for the first time for multi-agent RL problems.

Jaderberg et al. (2018) propose to train a diverse population of agents, to form two teams against each other, and to train each agent independently in a decentralized way, only through interaction with the environment, without knowledge of environment model, other agents, and human policy prior, and without communication with other agents. Each agent learn an internal reward signal, to generate internal goals, such as capturing a flag, to complement the sparse game winning reward. The authors propose a two-tier optimization process. The inner optimization maximizes agents' expected discounted future internal rewards. The outer optimization solves a meta-game, to maximize the meta-reward of winning the match, w.r.t. internal reward functions and hyperparameters, with meta transition dynamics provided by the inner optimization. The inner optimization is solved with RL. The outer optimization is solved with population based training (PBT) (Jaderberg et al., 2017), an online evolutionary method adapting internal rewards and hyperparameters and performing model selection by agent mutation, i.e., replacing under-performing agents with better ones. Auxiliary signals (Jaderberg et al., 2017) and differentiable neural computer memory (Graves et al., 2016) are also used to improve performance. RL agents are trained asynchronously from thousands of concurrent matches on randomly generated maps.

The authors design an agent to achieve strong capacity and avoiding several common RL issues, with the integration of learning and evolution. Learning from a diverse population of agents on random maps helps achieve skills generalizable to variability of maps, number of players, and choice of teammates, and stability in partially observable multi-agent environments. Learning an internal reward signal helps tackle the sparse reward problem and further the credit assignment issue. The multi-timescale representation helps with memory and long term temporal reasoning for high level strategies. The authors choose PBT instead of self play, since self play may be unstable in multi-agent RL environments, and needs more manipulation to support concurrent training for better scalability.

Experiments show that an agent can learn a disentangled representation to encode various knowledge of game situations, like conjunctions of flag status, re-spawn state, and room type, associating with activation of some individual neurones, and behaving like humans e.g., in navigating, following, and defending. Such knowledge is acquired through RL training, rather than from explicit models.

Experiments also show that the generalizable skills for tasks with random maps are supported by rich representation of spatial environments, induced by the temporal hierarchy and explicit memory module. Experiments show human level performance, and a survey shows that the agents are more collaborative than human participants. It appears that the training was conducted with less than 2000 commodity computers.

The authors mention the limitations of the current work: "the difficulty of maintaining diversity in agent populations, the greedy nature of the meta-optimisation performed by PBT, and the variance from temporal credit assignment in the proposed RL updates". See a blog at `https://deepmind.com/blog/capture-the-flag/`.

Sun et al. (2018) and Pang et al. (2018) have beaten full-game built-in AI in StarCraft II. OpenAI Five designs a Dota 2 agent for 5v5 plays, with a common RL algorithm, Proximal Policy Optimization (PPO) and self play, and beat human players. However, huge computation is involved, with 256 GPUs and 128,000 CPU cores. See `https://openai.com/five/`.

13 RELATIONAL RL

Integrating reinforcement learning and relational learning (Getoor and Taskar, 2007) is a promising approach to problem solving in AI. Relational RL makes connections between RL and classical AI, for knowledge representation and reasoning, which we discuss briefly in Section 8.3.

Džeroski et al. (2001) propose relational RL. Tadepalli et al. (2004) give an overview of relational RL, and identify several challenges: suitable function approximation to represent relational knowledge, generalization across objects, transferability across tasks, run-time planning and reasoning, and, incorporating prior knowledge. Tadepalli et al. (2004) further propose relational RL as a solution to these challenges. Guestrin et al. (2003) introduce relational MDPs. Diuk et al. (2008) introduce objected-oriented MDPs (OO-MDPs), a close approach to relational RL. See more work, e.g., Mrowca et al. (2018), Palm et al. (2018), and Santoro et al. (2018).

See NIPS 2018 Workshop on Relational Representation Learning at `https://r2learning.github.io`.

We discuss some papers about relational learning and relational RL below.

RELATIONAL LEARNING

Battaglia et al. (2018) first discuss ways to incorporate relational inductive bias with deep learning. In fully connected networks (FC), entities are units, and their relations are all to all, thus have only weak relational inductive bias. In convolutional neural networks (CNNs), entities are still units, or grid elements, e.g. pixels in images, and relations are local. CNNs impose locality and translation invariance as relational inductive bias. In recurrent neural networks (RNN), entities include input and hidden state at each time step, and relations are the mapping from previous hidden state and current input to the hidden state of this step. This mapping is reuse over time steps, thus temporal invariance is the relational inductive bias for RNN.

Battaglia et al. (2018) then propose graph network (GN) to incorporate relational inductive bias, to attempt to achieve combinatorial generalization, i.e., the capacity to use known elements to build new inferences, predictions and behaviours. GN can operate on arbitrary relational structure, having explicit representation of entities (nodes) and relationships (edges), grounding in data. Node and edge permutations are invariant in GN. GN generalizes previous graph neural networks, e.g., Scarselli et al. (2009).

Santoro et al. (2017) propose relation networks (RNs) for relational reasoning in a plug-and-play way in neural networks. The authors experiment RN-augmented networks on visual question answering, text-based question answering, and reasoning about dynamic physical systems. Santoro et al. (2018) propose a relational memory core (RMC), a memory module, to handle tasks with relational reasoning, using self attention (Vaswani et al., 2017) to deal with memory interaction. The authors test RMC on RL tasks, program evaluation, and language modelling. Battaglia et al. (2016) propose interaction network to learning about objects, relations and physics, and evaluate it with n-body problems, rigid-body collision, and non-rigid dynamics.

Chen et al. (2018a) propose a framework for iterative visual reasoning, beyond current recognition systems with CNNs. It is composed of a local module to store previous beliefs, a global model for graph reasoning. It refines estimates by rolling out these models iteratively, and cross-feeding predictions to each other. The graph model consists of: 1) a knowledge graph, with nodes and edges to represent classes and their relationships respectively; 2) a region graph of the current image, with nodes and edges as regions in the images and their spatial relationships; 3) an assignment graph, assigning regions to classes. An attention mechanism is used to combine both local and global modules for final predictions.

Hudson and Manning (2018) propose memory, attention, and control (MAC) cell, with three units, namely, control, read, and write units, to construct recurrent compositional attention networks for reasoning, by imposing structural constraints in the operation of and interaction between cells, for the purpose of interpretability, generalization, computation efficiency, and data efficiency. Hudson and Manning (2018) evaluate their proposed approach on a visual question answering (VQA) task with the CLEVR data set. Watch a video at `https://www.youtube.com/watch?v=jpNLp9SnTF8`.

RELATIONAL RL

Zambaldi et al. (2018) propose to improve sample efficiency, generalization capacity, and interpretability of deep RL with relational reinforcement learning (Džeroski et al., 2001). The proposed network architecture consists of FC, CNNs, and a relational module, which uses self-attention (Vaswani et al., 2017) to reason about interactions between entities, and to facilitate model-free policy optimization. Zambaldi et al. (2018) construct a navigation and planning task, BOX-World, to test the capacity of relational reasoning, and show good performance. Zambaldi et al. (2018) also experiment on StarCraft II mini-games in the the StarCraft II Learning Environment (SC2LE) (Vinyals et al., 2017), and achieve good performance on six mini-games. Note that Sun et al. (2018) and Pang et al. (2018) have beaten full-game built-in AI in StarCraft II.

Wang et al. (2018b) propose NerveNet, resembling the neural nervous system to a graph, to learn structured policy for continuous control. A policy is defined using the graph neural networks (GNN) (Scarselli et al., 2009), in which, information is propagated over the agent structure, and actions are predicted for different parts of the agent. The authors evaluate NerveNet in OpenAI Gym environment, and on transfer learning and multi-task learning tasks. See `http://www.cs.toronto.edu/~tingwuwang/nervenet.html` for open source and demo.

Keramati et al. (2018) propose strategic object oriented RL (SOORL) for model-learning with automatic model selection, and planning with strategic exploration. The authors achieve positive reward in Pitfall!, a hard Atari game. However, we note that, concurrently, Aytar et al. (2018) achieve much better results in Pitfall! and two other hard Atari games with an approach of self-supervision+RL, albeit relational and object oriented mechanisms are worth more efforts.

Yang et al. (2018) propose planning-execution-observation-RL (PEORL) to integrate hierarchical RL with symbolic planning for dynamic, uncertain environments.

Zambaldi et al. (2018) utilize pixel feature vectors resulting from CNNs as entities, a problem agnostic approach. Keramati et al. (2018) utilize bounding boxes to detect object. Yang et al. (2018) work with manually crafted symbolic knowledge. The performance would be further improved with an advanced reasoning technique about images to find entities and relations, e.g., Chen et al. (2018a), Santoro et al. (2017), Santoro et al. (2018), or, Battaglia et al. (2016), etc.

14 LEARNING TO LEARN

Learning to learn, a.k.a. meta-learning, is learning about some aspects of learning. It includes concepts as broad as transfer learning, multi-task learning, one/few/zero-shot learning, learning to optimize, learning to reinforcement learn, learning combinatorial optimization, hyper-parameter learning, neural architecture design, automated machine learning (AutoML), continual learning, etc. Learning to learn is a core ingredient to achieve strong AI (Lake et al., 2016), and has a long history, e.g., Ellis (1965), Schmidhuber (1987), Bengio et al. (1991), Sutton (1992), Thrun and Pratt (1998), Hochreiter et al. (2001), and Brazdil et al. (2009).

Li and Malik (2017) and Li and Malik (2017), along with the blog at `http://bair.berkeley.edu/blog/2017/09/12/learning-to-optimize-with-rl/`, divide various learning to learn methods into three categories: learning what to learn, learning which model to learn, and, learning how to learn. The authors mention that, "roughly speaking, 'learning to learn' simply means learning something about learning". The authors discuss that the term of learning to learn has the root in the idea of metacognition by Aristotle in 350 BC (`http://classics.mit.edu/Aristotle/soul.html`), which describes "the phenomenon that humans not only reason, but also reason about their own process of reasoning". In the category of learning what to learn, the aim is to learn values for base-model parameters, gaining the meta-knowledge of commonalities across the family of related tasks, and to make the base-learner useful for those tasks (Thrun and Pratt, 1998). Examples in this category include methods for transfer learning, multi-task learning and few-shot learning. In the category of learning which model to learn, the aim is to learn which base-model is most suitable for a task (Brazdil et al., 2009), gaining the meta-knowledge of correlations of performance between various base-models, by investigating their expressiveness and searchability. This learns the outcome of learning. In the category of learning how to learn, the aim is to learn the process of learning, gaining the meta-knowledge of commonalities in learning algorithms behaviours. There are three components for learning how to learn: the base-model, the base-algorithm to train the base-model, and the meta-algorithm to learn the base-algorithm. The goal of learning how to learn is to design the meta-algorithm to learn the base-algorithm, which trains the base-model. Bengio et al. (1991), Andrychowicz et al. (2016), Li and Malik (2017), and Li and Malik (2017) fall into this category. Fang et al. (2017) study learning how to active learn. Wang et al. (2018c) study how to learn MCMC proposals. Zhang et al. (2018b) study learning to multitask. Negrinho et al. (2018) study learning beam search policies. Hsu et al. (2018) study unsupervised learning with meta-learning.

Finn et al. (2017a), along with the blog at `https://github.com/cbfinn/maml`, summarize that there are three categories of methods for learning to learn, namely, recurrent models, metric learning, and learning optimizers. In the approach of recurrent models, a recurrent model, e.g. an LSTM, is trained to take in data points, e.g., (image, label) pairs for an image classification task, sequentially from the dataset, and then processes new data inputs from the task. The meta-learner usually uses gradient descent to train the learner, and the learner uses the recurrent network to process new data. Santoro et al. (2016), Mishra et al. (2018), Duan et al. (2016), and Wang et al. (2016) fall into this category. In the approach of metric learning, a metric space is learned to make learning efficient, mostly for few-shot classification. Koch et al. (2015), Vinyals et al. (2016), and Snell et al. (2017) fall into this category. In the approach of learning optimizers, an optimizer is learned, using a meta-learner network to learn to update the learner network to make the learner learn a task effectively. Wichrowska et al. (2017), Andrychowicz et al. (2016), Li and Malik (2017), and Li and Malik (2017) fall into this category. Motivated by the success of using transfer learning for initializing computer vision network weights with the pre-trained ImageNet weights (Donahue et al., 2014), Finn et al. (2017a) propose model-agnostic meta-learning (MAML) to optimize an initial representation for a learning algorithm, so that the parameters can be fine-tuned effectively from a few examples.

Duan (2017) gives a brief review of meta-learning. The author discusses meta-learning for supervised learning, including metric-based models, optimization-based models, and fully generic models, and other applications. The author also discusses meta-learning for control, and proposes to learn reinforcement learning algorithms (Duan et al., 2016), and one-shot imitation learning (Duan et al., 2017).

Combinatorial optimization is critical for many areas, like social networks, telecommunications, and transportation. Many combinatorial optimization problems are NP-hard, and algorithms follow three approaches: exact algorithms, approximate algorithms, and heuristics, and all of them require specialized knowledge and human efforts for trail-and-error. Dai et al. (2017) propose to automate combinatorial optimization using deep RL with graph embedding (Dai et al., 2016).

We discuss learning to plan, including, value iteration networks (VIN) (Tamar et al., 2016), and predictron (Silver et al., 2016b), and learning to search, MCTSnets (Guez et al., 2018), in Chapter 6.

See NIPS 2018 Workshop on Meta-Learning at `http://metalearning.ml/2018/`.

Continual learning (Chen and Liu, 2016; Kirkpatrick et al., 2017; Lopez-Paz and Ranzato, 2017; Xu and Zhu, 2018) is important for achieving general intelligence. See Singh (2017) for a tutorial about continual learning. See NIPS 2018 Workshop on Continual Learning at `https://sites.google.com/view/continual2018`. See ICML 2018 Workshop on Lifelong Learning: A Reinforcement Learning Approach at `https://sites.google.com/view/llarla2018/home`.

We discuss few/one/zero-shot learning in Section 14.1, transfer and multi-task learning in Section 14.2, learning to optimize in Section 14.3, learning reinforcement learn in Section 14.4, learning combinatorial optimization in Section 14.5, and AutoML in Section 14.6.

14.1 Few/One/Zero-Shot Learning

As discussed in (Finn et al., 2017a), the aim of few-shot meta-learning is to train a model adaptive to a new task quickly, using only a few data samples and training iterations. Finn et al. (2017a) propose model-agnostic meta-learning (MAML) to optimize an initial representation for a learning algorithm, so that the parameters can be fine-tuned effectively from a few examples. To illustrate MAML, consider a model f_θ with parameter θ. The model parameter θ becomes θ_i', when adapting to a new task $\mathcal{T}_i$. We compute the new parameter θ_i' with one gradient descent update,

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta), \tag{59}$$

or more updates, on task $\mathcal{T}_i$. Here α is the step size. We train model parameters by optimizing the meta-objective,

$$\min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) = \min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)}), \tag{60}$$

the performance of $f_{\theta_i'}$, w.r.t. θ across tasks sampled from $p(\mathcal{T})$. Note, the meta-optimization is performed over the old parameters θ, and the objective is computed using the new parameters θ'. This aims to optimize the model parameters so that one or a few gradient steps on a new task will produce maximally effective behaviour on that task. We perform the meta-optimization across tasks via SGD, and update θ as,

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}), \tag{61}$$

where β is the meta step size. The intuition underlying MAML is that some internal representations are effective for adaptation, and the goal of MAML is to find model parameters sensitive to changes in the task, so that after we draw a task from $p(\mathcal{T})$, when the direction of the gradient of the loss function changes, small parameter changes will significantly improve the loss function.

MAML works for both supervised learning and reinforcement learning. Experiments show good results on few-shot classification, regression, and policy gradients RL with neural network policies. See a blog about learning to learn and MAML at `http://bair.berkeley.edu/blog/2017/07/18/learning-to-learn/`

Finn and Levine (2018) show that deep representation integrated with gradient descent has sufficient capacity to approximate any learning algorithms, i.e., the universality of meta-learning, and show empirically that gradient-based meta-learning found learning strategies with better generalization capacity than recurrent models. Grant et al. (2018) treat gradient-based meta-learning as hierarchical Bayes. Finn et al. (2018) study probabilistic MAML. Yoon et al. (2018) study Bayesian MAML.

Al-Shedivat et al. (2018) propose to use the framework of learning to learn for continuous adaptation in non-stationary and competitive environments, by treating a non-stationary environment as a sequence of stationary tasks. The authors develope a gradient-based meta-learning algorithm adaptive in dynamically changing and adversarial scenarios based on MAML (Finn et al., 2017a), by anticipating changes in environment and updating policies accordingly. The proposed approach attempt to handle Markovian dynamics on two level of hierarchy: at the upper level for dynamics of tasks, and at the lower level for MDPs representing particular tasks. The authors evaluate the performance of the proposed approach 1) on a single-agent multi-leg robots locomotion task in MuJoCo physics simulator, with handcrafted nonstationarity, by selecting a pair of legs to scale down the torques applied to joints, until fully paralyzed, and, 2) on iterated adaptation games in RoboSumo, which is in a 3D environment, with simulated physics, having pairs of agents to compete, and not only non-stationary but also adversarial. The authors assume that trajectories from the current task contain some information about the next task, so that tasks become dependant sequentially. The proposed algorithm may not take advantage of more data, and it could diverge when there are large distributional changes from iteration to iteration.

Lake et al. (2015) propose an one-shot concept learning model, for handwritten characters in particular, with probabilistic program induction. Koch et al. (2015) propose siamese neural networks with metric learning for one-shot image recognition. Vinyals et al. (2016) design matching networks for one-shot classification. Duan et al. (2017) propose a model for one-shot imitation learning with attention for robotics. Johnson et al. (2017) present zero-shot translation for Google's multilingual neural machine translation system. Kaiser et al. (2017b) design a large scale memory module for life-long one-shot learning to remember rare events. Kansky et al. (2017) propose Schema Networks for zero-shot transfer with a generative causal model of intuitive physics. Snell et al. (2017) propose prototypical networks for few/zero-shot classification by learning a metric space to compute distances to prototype representations of each class. George et al. (2017) propose a generative vision model to train with high data efficiency, breaking text-based CAPTCHA. Liu et al. (2018c) study generalization of zero-shot learning with deep calibration network.

14.2 Transfer/Multi-task Learning

Transfer learning is about transferring knowledge learned from different domains, possibly with different feature spaces and/or different data distributions (Taylor and Stone, 2009; Pan and Yang, 2010; Weiss et al., 2016). As reviewed in Pan and Yang (2010), transfer learning can be inductive, transductive, or unsupervised. Inductive transfer learning includes self-taught learning and multi-task learning. Transductive transfer learning includes domain adaptation and sample selection bias/covariance shift. Taylor and Stone (2009) compare RL transfer learning algorithms w.r.t. the following performance metrics: jumpstart, asymptotic performance, total reward, transfer ratio, time to threshold, and against the following dimensions: task difference assumption, source task selection, task mapping, transferred knowledge, and allowed learners. Multitask learning (Caruana, 1997; Zhang et al., 2018a; Ruder, 2017) learns related tasks with a shared representation in parallel, leveraging information in related tasks as an inductive bias, to improve generalization, and to help improve learning for all tasks. The modular structure of hierarchical RL approaches is usually conducive to transfer and multi-task learning, which we discussed in Chapter 11.

Whye Teh et al. (2017) propose Distral, distill & transfer learning, for joint training of multiple tasks, by sharing a distilled policy, trained to be the centroid of policies for all tasks, to capture common behavioural structure across tasks, and training each task policy to be close to the shared policy. The design of Distral is to overcome issues in transfer and multi-task learning, that in the approach of share neural network parameters, gradients from different tasks may interfere each other negatively, and that one task may dominate the learning of the shared model, due to different reward functions of different tasks. Whye Teh et al. (2017) design Distral following the techniques of distillation (Bucila et al., 2006; Hinton et al., 2014), and soft Q-learning (Haarnoja et al., 2017; Nachum et al., 2017), a.k.a., G-learning (Fox et al., 2016). The authors observe that, the distillation arises naturally, when optimize task models towards a distilled model, when using KL divergence as a regularization. Moreover, the distilled model serves as a regularizer for task models training, and it may help transferability by regularizing neural networks in a space more semantically meaningful, e.g., for policies, rather than for network parameters. Distral can be instantiated in several forms, and it outperforms empirically the baseline A3C algorithms in grid world and complex 3D environments.

Barreto et al. (2017) propose a transfer framework for tasks with different reward functions but the same environment dynamics, based on two ideas, namely, successor features and generalized policy improvement. Successor features are a value function representation decoupling environment dynamics from rewards, extending the successor representation (Dayan, 1993), as discussed in Section 8.1, for continuous tasks with function approximation. Generalized policy improvement operates on multiple policies, rather than one policy as in the policy improvement in dynamic programming. The authors integrate these two ideas for transfer learning among tasks, and establish two theorems, one for performance guarantee of a task before any learning, and another for performance guarantee of a task if it had seen similar tasks. The author design a method based on these ideas and analysis, and show good performance on navigation tasks and a task to control a simulated robotic arm.

Gupta et al. (2017a) formulate the multi-skill problem for two agents to learn multiple skills, define the common representation using which to map states and to project the execution of skills, and design an algorithm for two agents to transfer the informative feature space maximally to transfer new skills, with similarity loss metric, autoencoder, and reinforcement learning. The authors validate their proposed approach with two simulated robotic manipulation tasks.

As a practical example of transfer learning in computer vision, Kornblith et al. (2018) investigate the transferability of ImageNet architectures and features, for 13 classification models on 12 image classification tasks, in three settings: as fixed feature extractors, fine-tuning, and training from random initialization. The authors observe that, ImageNet classification network architectures generalize well across datasets, but fixed ImageNet features do not transfer well.

See recent work in transfer learning/multi-task learning e.g., Andreas et al. (2017), Dong et al. (2015), Kaiser et al. (2017a), Kansky et al. (2017), Killian et al. (2017) Long et al. (2015), Long et al. (2016), Long et al. (2017), Mahajan et al. (2018), Maurer et al. (2016), McCann et al. (2017), Mo et al. (2018), Parisotto et al. (2016), Papernot et al. (2017), Pérez-D'Arpino and Shah (2017), Rajendran et al. (2017), Sener et al. (2018), Smith et al. (2017), Sohn et al. (2018), Yosinski et al. (2014), and, Zhao et al. (2018a).

See NIPS 2015 workshop on Transfer and Multi-Task Learning: Trends and New Perspectives at `https://sites.google.com/site/tlworkshop2015/`.

We will discuss sim-to-real transfer learning in robotics in Chapter 16.1.

14.3 Learning to Optimize

Li and Malik (2017) and Li and Malik (2017) propose to automate unconstrained continuous optimization algorithms with RL, in particular, guided policy search (Levine et al., 2016). Algorithm 11 presents a general structure of optimization algorithms. Li and Malik (2017) and Li and Malik (2017) formulate a RL problem as follows: 1) the state is the current iterate, $x^{(i)}$, and may also include some features along the historical optimization trajectory, like the history of gradients, iterates, and objective values; 2) the action is the step vector, Δx, which updates the iterate $x^{(i)}$; and, 3) the policy is the update formula $\phi(\cdot)$, which depends on the current iterate, and the history of gradients, iterates, and objective values. Thus, learning the policy is equivalent to learning the optimization algorithm. One possible cost function is the objective function value at the current iterate. A RL algorithm in this problem does not have access to the state transition model. See a blog at `http://bair.berkeley.edu/blog/2017/09/12/learning-to-optimize-with-rl/`. See also Andrychowicz et al. (2016) and Bello et al. (2017).

14.4 Learning Reinforcement Learn

Xu et al. (2018) investigate a fundamental problem in RL to discover an optimal form of return, and propose a gradient-based meta-learning algorithm to learn the return function, by tuning meta-parameters of the return function, e.g., the discount factor, γ, the bootstrapping parameter, λ, etc., in an online fashion, when interacting with the environment. This is in contrast to many recent work in learning to learn that are in a setting of transfer/multi-task learning. Experiments on Atari games show good results. The technique would be general for other components of the return function, the learning update rule, and hyperparameter tunning.

Input: objective function f

$x^{(0)} \leftarrow$ random point in the domain of f **for** $i = 1, 2, 3, \ldots$ **do**
  $\Delta x \leftarrow \phi(f, \{x^{(0)}, \ldots, x^{(i-1)}\})$ **if** *stopping condition is met* **then**
    return $x^{(i-1)}$
  end
  $x^{(i)} \leftarrow x^{(i-1)} + \Delta x$
end

Algorithm 11: General structure of optimization algorithms, adapted from Li and Malik (2017)

Wang et al. (2018a) investigate that prefrontal cortex works as a meta-reinforcement learning system. Learning to learn, or meta-learning, is related to the phenomenon that our brain can do so much with so little. A hypothesis is that we learn on two timescales: learning on specific examples in short term, and learning abstract skills or rules over long term. Kahneman (2011) describes two modes of thought: one, fast, instinctive and emotional; another, slower, more deliberative, and more logical. See a blog at `https://deepmind.com/blog/prefrontal-cortex-meta-reinforcement-learning-system/`.

Duan et al. (2016) and Wang et al. (2016) propose to learn a flexible RNN model to handle a family of RL tasks, to improve sample efficiency, learn new tasks in a few samples, and benefit from prior knowledge. The agent is modelled with RNN, with inputs of observations, rewards, actions and termination flags. The weights of RNN are trained with RL, in particular, TRPO in Duan et al. (2016) and A3C in Wang et al. (2016). Duan et al. (2016) and Wang et al. (2016) achieve similar performance as specific RL algorithms for various problems.

Houthooft et al. (2018) propose evolved policy gradients with meta-learning. Gupta et al. (2018) propose meta-RL of structured exploration strategies. Stadie et al. (2018) study the importance of sampling in meta-RL.

14.5 Learning Combinatorial Optimization

Dai et al. (2017) propose to automate algorithm design for combinatorial optimization problems on graphs using deep RL with graph embedding (Dai et al., 2016), by learning heuristics on problem instances from a distribution $\mathbb{D}$ of a graph optimization problem, to generalize to unseen instances from $\mathbb{D}$.

The authors propose to construct a feasible solution following a greedy algorithm design pattern, by successively adding nodes based on the graph structure and the current partial solution. A deep graph embedding, structure2vec (Dai et al., 2016), is used to represent nodes, considering their properties in the graph, and helps generalize the algorithm to different graph sizes.

The graph embedding is also used to represent state, action, value function, and policy. A state is a sequence of nodes on a graph. An action is selecting a node that is not part of the current state. The reward is the change in the cost function after taking an action, i.e., adding a node, and transition to a new state. A deterministic greedy policy is used based on the action value function. For example, in the Minimum Vertex Cover (MVC) problem, we are to find a subset of nodes in a graph, so that every edge is covered and the number of nodes selected is minimized. In MVC, a state is the subset of nodes selected so far, an action is to add a node to the current subset, the reward is -1 for each action, and the termination condition is all edges are covered. The authors propose to learn a greedy policy parameterized by the graph embedding network using n-step fitted Q-learning.

Dai et al. (2017) evaluate the proposed approach on three graph combinatorial optimization problems: Minimum Vertex Cover (MVC), Maximum Cut (MAXCUT), and Traveling Salesman Problem (TSP), and compare with pointer networks with actor-critic (Bello et al., 2016), and strong baseline approximate and heuristics algorithms for each problem respectively. Dai et al. (2017) achieve good performance on both synthetic and real graphs. See the open source at `https://github.com/Hanjun-Dai/graph_comb_opt`.

Vinyals et al. (2015) propose pointer networks to learn the conditional probability of an output sequence of discrete tokens, corresponding to positions in an input sequence, by generating output sequence with attention as a pointer to select an element of the input sequence, and evaluate performance on three combinatorial optimization problems, finding planar convex hulls, computing Delaunay triangulations, and the planar Travelling Salesman Problem. Pointer networks are graph-agnostic, in contrast to the graph embedding in Dai et al. (2017).

14.6 AUTOML

AutoML is about automating the process of machine learning. See a website for AutoML at `http://automl.chalearn.org`. See NIPS 2018 AutoML for Lifelong Machine Learning Competition at `https://www.4paradigm.com/competition/nips2018`. See also a usable machine learning project Bailis et al. (2017).

Neural network architecture design is one particular task of AutoML. Neural architecture design is a notorious, nontrivial engineering issue. Neural architecture search provides a promising avenue to explore. See a survey (Elsken et al., 2018).

Zoph and Le (2017) propose neural architecture search to generate neural networks architectures with an RNN trained by RL, in particular, REINFORCE, searching from scratch in variable-length architecture space, to maximize the expected accuracy of the generated architectures on a validation set. In the RL formulation, a controller generates hyperparameters as a sequence of tokens, which are actions chosen from hyperparameters spaces. Each gradient update to the policy parameters corresponds to training one generated network to convergence. An accuracy on a validation set is the reward signal. The neural architecture search can generate convolutional layers, with skip connections or branching layers, and recurrent cell architectures. The authors design a parameter server approach to speed up training. Comparing with state-of-the-art methods, the proposed approach achieves competitive results for an image classification task with CIFAR-10 dataset, and better results for a language modeling task with Penn Treebank.

Zoph et al. (2017) propose to transfer the architectural building block learned with the neural architecture search (Zoph and Le, 2017) on small dataset to large dataset for scalable image recognition. Baker et al. (2017) propose a meta-learning approach, using Q-learning with ϵ-greedy exploration and experience replay, to generate CNN architectures automatically for a given learning task. Zhong et al. (2017) propose to construct network blocks to reduce the search space of network design, trained by Q-learning. See also Liu et al. (2017), Liu et al. (2017), Real et al. (2017), and Real et al. (2018). Note that Real et al. (2018) show that evolutionary approaches can match or surpass human-crafted and RL-designed image neural network classifiers.

Jin et al. (2018) propose a Bayesian approach for efficient search. See Cai et al. (2018a) for an approach with limited computation resources (200 GPU hours). See also Chen et al. (2018a), Kandasamy et al. (2018), Luo et al. (2018), and Wong et al. (2018).

There are recent works exploring new neural architectures (manually). Vaswani et al. (2017) propose a new archichitecture for translation that replaces CNN and RNN with attention and positional encoding. Kaiser et al. (2017a) propose to train a single model, MultiModel, which is composed of convolutional layers, an attention mechanism, and sparsely-gated layers, to learn multiple tasks from various domains, including image classification, image captioning and machine translation. Wang et al. (2016b) propose the dueling network architecture to estimate state value function and associated advantage function, to combine them to estimate action value function for faster convergence. Tamar et al. (2016) introduce value iteration networks (VIN), a fully differentiable CNN planning module to approximate the value iteration algorithm, to learn to plan. Silver et al. (2016b) propose the predictron to integrate learning and planning into one end-to-end training procedure with raw input in Markov reward process. These neural architectures were designed manually. It would be interesting to see if learning to learn can help automate such neural architecture design.

Neural architecture design has already had industrial impact. Google, among others, is working on AutoML, in particular, AutoML Vision, and extends AutoML to NLP and contact center, etc. See blogs about Google AutoML at `http://goo.gl/ijBjUr`, `http://goo.gl/irCvD6`, and, `http://goo.gl/VUzCNt`. See Auto-Keras at `https://autokeras.com`.

There are recent interesting work in AutoML. He et al. (2018) propose AutoML for model compression. Cubuk et al. (2018) propose AutoAugment to automate data augmentation for images. Chen et al. (2018c) study learning to optimize tensor programs.

See 2018 International Workshop on Automatic Machine Learning (collocated with the Federated AI Meeting, ICML, IJCAI, AMAS, and ICCBR) at `https://sites.google.com/site/automl2018icml/`.

One limitation of neural architecture design is that the network components are manually designed, and it is not clear if AI has the creativity to discover new components, e.g., discovering residual connections before ResNets were designed.

PART III: APPLICATIONS

Reinforcement learning has a wide range of applications. We discuss games in Chapter 15 and robotics in Chapter 16, two classical RL application areas. Games are important testbeds for RL/AI. Robotics will be critical in the era of AI. Natural language processing (NLP) follows in Chapter 17, which enjoys wide and deep applications of RL recently. Next we discuss computer vision in Chapter 18, in which, there are efforts for integration with language. In Chapter 19, we discuss finance and business management, which have natural problems for RL. We discuss more applications in Chapter 20, healthcare in Section 20.1, education in Section 20.2. energy in Section 20.3, transportation in Section 20.4, and computer systems in Section 20.5. We attempt to put reinforcement learning in the wide context of science, engineering, and art in Section 20.6,

Reinforcement learning is widely utilized in operations research (Powell, 2011), e.g., supply chain, inventory management, resource management, etc; we do not list it as an application area — it is implicitly a component in application areas like energy and transportation. We do not list smart city, an important application area of AI, as it includes several application areas here: healthcare, education, energy, transportation, etc.

These application areas build on RL techniques as discussed in previous chapters, and may overlap with each other, e.g., a robot may need skills from application areas like computer vision and NLP.

RL is usually for sequential decision making. However, some problems, seemingly non-sequential on surface, like neural network architecture design (Zoph and Le, 2017), and, model compression (He et al., 2018), have been approached by RL. Creativity would push the frontiers of deep RL further w.r.t. core elements, important mechanisms, and applications. RL is probably helpful, if a problem can be regarded as or transformed to a sequential decision making problem, and states, actions, maybe rewards, can be constructed. Many problems with manual design of strategies may be automated, and RL is a potential solution method. We illustrate deep RL applications in Figure 4. We maintain a blog, Reinforcement Learning Applications, at `https://medium.com/@yuxili/`.

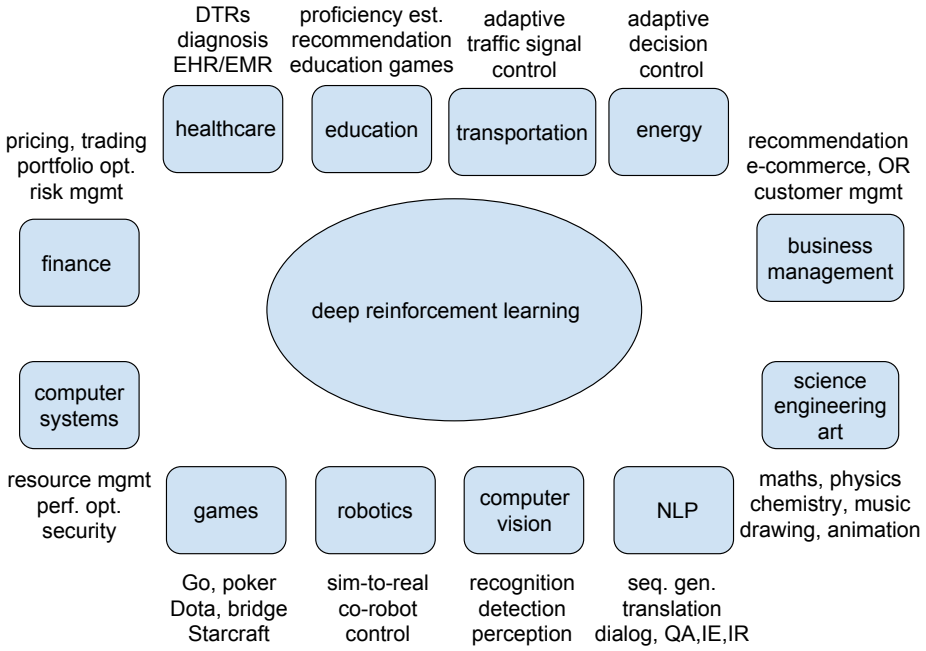

Figure 4: Deep Reinforcement Learning Applications

15 GAMES

Games provide excellent testbeds for AI algorithms, dated back to the ages of Alan Turing, Claude Shannon, and John von Neumann. In games, we have good or even perfect simulators, and we can generate unlimited data. We have seen great achievements of human-level or super-human performance in computer games, e.g.,

- Chinook (Schaeffer, 1997; Schaeffer et al., 2007) for Checkers,
- Deep Blue (Campbell et al., 2002) for chess,
- Logistello (Buro, 1999) for Othello,
- TD-Gammon (Tesauro, 1994) for Backgammon,
- GIB (Ginsberg, 2001) for contract bridge,
- MoHex (Huang et al., 2013; Gao et al., 2017) for Hex,
- DQN (Mnih et al., 2016) and Aytar et al. (2018) for Atari 2600 games,
- AlphaGo (Silver et al., 2016a) and AlphaGo Zero (Silver et al., 2017) for Go,
- Alpha Zero (Silver et al., 2017) for chess, shogi, and Go,
- Cepheus (Bowling et al., 2015), DeepStack (Moravčík et al., 2017), and Libratus (Brown and Sandholm, 2017a;b) for heads-up Texas Hold'em Poker,
- Jaderberg et al. (2018) for Quake III Arena Capture the Flag,
- OpenAI Five, for Dota 2 at 5v5, `https://openai.com/five/`,
- Zambaldi et al. (2018), Sun et al. (2018), and Pang et al. (2018) for StarCraft II.

Schaeffer (1997), Buro (1999), Ginsberg (2001), and Campbell et al. (2002) employ heuristic search techniques (Russell and Norvig, 2009), in particular, alpha-beta search. Tesauro (1994), Mnih et al. (2016), Silver et al. (2016a), Silver et al. (2017), Silver et al. (2017), Jaderberg et al. (2018), OpenAI Five, Zambaldi et al. (2018), Sun et al. (2018), and Pang et al. (2018) are powered with (deep) reinforcement learning; the "Alpha series" are also equipped with Monte Carlo tree search (MCTS), a heuristic search technique. Aytar et al. (2018) follows a self-supervised approach, together with deep RL. Huang et al. (2013) employ MCTS, and Gao et al. (2017) extend it with deep learning. Bowling et al. (2015) and Moravčík et al. (2017), handle imperfect information with counterfactual regret minimization (CFR), and follow generalized policy iteration. Brown and Sandholm (2017a;b) utilize classical search techniques.

As deep RL has achieved human-level or superhuman performance for many two-play games, multi-player games are at the frontier for scientific discovery of AI, with outstanding achievements already achieved for games like Quake III Arena Capture the Flag and Dota 2 5v5.

We discuss three categories of games, namely, board games in Section 15.1, card games in Section 15.2, and video games in Section 15.3, loosely for two-player perfect information zero-sum games, imperfect information zero-sum games, and games with video frames as inputs, mostly with partial observability or imperfect information, respectively.

The above classification is not exhaustive, e.g., it does not include single agent, non-board, non-card, non-video games, like Rubik's Cube (McAleer et al., 2018). Computer games have much wider topics, e.g., storytelling (Thue et al., 2007).

See Yannakakis and Togelius (2018) for a book on AI and games. See Justesen et al. (2017) for a survey about applying deep (reinforcement) learning to video games. See Ontañón et al. (2013) for a survey about Starcraft. Check AIIDE and CIG Starcraft AI Competitions, and its history at `https://www.cs.mun.ca/~dchurchill/starcraftaicomp/history.shtml`. See Lin et al. (2017) for a StarCraft dataset.

15.1 BOARD GAMES

Board games like chess, Go and, Backgammon, are classical testbeds for RL/AI algorithms. In such games, players have prefect information of two players. Tesauro (1994) approach Backgammon using neural networks to approximate value function learned with TD learning, and achieve human level performance.

COMPUTER GO

The challenge of solving computer Go comes from not only the gigantic search space of size 250^{150}, an astronomical number, but also the hardness of position evaluation (Müller, 2002), which was successfully used in solving many other games, like chess and Backgammon.

AlphaGo (Silver et al., 2016a), a computer Go program, won the human European Go champion, 5 games to 0, in October 2015, and became the first computer Go program to won a human professional Go player without handicaps on a full-sized 19 $\times$ 19 board. Soon after that in March 2016, AlphaGo defeated Lee Sedol, an 18-time world champion Go player, 4 games to 1, making headline news worldwide. This set a landmark in AI. AlphaGo defeated Ke Jie 3:0 in May 2017. AlphaGo Zero (Silver et al., 2017) further improved previous versions by learning a superhuman computer Go program without human knowledge. Alpha Zero (Silver et al., 2017) generalized the learning framework in AlphaGo Zero to more domains. See blogs for AlphaGo at `https://deepmind.com/research/alphago/`, and for AlphaGo Zero at `https://deepmind.com/blog/alphago-zero-learning-scratch/`.

Tian and Zhu (2016) also investigate computer Go. See Facebook open source ELF OpenGo, `https://github.com/pytorch/ELF/`, and a blog, `https://research.fb.com/facebook-open-sources-elf-opengo/`.

ALPHAGO: TRAINING PIPELINE AND MCTS

We discuss briefly how AlphaGo works based on Silver et al. (2016a) and Sutton and Barto (2018). See Sutton and Barto (2018) for an intuitive description of AlphaGo.

AlphaGo is built with techniques of deep convolutional neural networks, supervised learning, reinforcement learning, and Monte Carlo tree search (MCTS) (Browne et al., 2012; Gelly and Silver, 2007; Gelly et al., 2012). AlphaGo is composed of two phases: neural network training pipeline and MCTS. The training pipeline phase includes training a supervised learning (SL) policy network from expert moves, a fast rollout policy, a RL policy network, and a RL value network.

The SL policy network has convolutional layers, ReLU nonlinearities, and an output softmax layer representing probability distribution over legal moves. The inputs to the CNN are 19 $\times$ 19 $\times$ 48 image stacks, where 19 is the dimension of a Go board and 48 is the number of features. State-action pairs are sampled from expert moves to train the network with stochastic gradient ascent to maximize the likelihood of the move selected in a given state. The fast rollout policy uses a linear softmax with small pattern features.

The RL policy network improves SL policy network, with the same network architecture, and the weights of SL policy network as initial weights, and policy gradient for training. The reward function is +1 for winning and -1 for losing in the terminal states, and 0 otherwise. Games are played between the current policy network and a random, previous iteration of the policy network, to stabilize the learning and to avoid overfitting. Weights are updated by stochastic gradient ascent to maximize the expected outcome.

The RL value network still has the same network architecture as SL policy network, except the output is a single scalar predicting the value of a position. The value network is learned in a Monte Carlo policy evaluation approach. To tackle the overfitting problem caused by strongly correlated successive positions in games, data are generated by self-play between the RL policy network and itself until game termination. The weights are trained by regression on state-outcome pairs, using stochastic gradient descent to minimize the mean squared error between the prediction and the corresponding outcome.

In MCTS phase, AlphaGo selects moves by a lookahead search. It builds a partial game tree starting from the current state, in the following stages: 1) select a promising node to explore further, 2) expand a leaf node guided by the SL policy network and collected statistics, 3) evaluate a leaf node with a mixture of the RL value network and the rollout policy, 4) backup evaluations to update the action values. A move is then selected.

ALPHAGO ZERO

AlphaGo Zero can be understood as following a generalized policy iteration scheme, incorporating MCTS inside the training loop to perform both policy improvement and policy evaluation. MCTS may be regarded as a policy improvement operator. It outputs move probabilities stronger than raw probabilities of the neural network. Self-play with search may be regarded as a policy evaluation operator. It uses MCTS to select moves, and game winners as samples of value function. Then the policy iteration procedure updates the neural network's weights to match the move probabilities and value more closely with the improved search probabilities and self-play winner, and conduct self-play with updated neural network weights in the next iteration to make the search stronger.

The features of AlphaGo Zero (Silver et al., 2017), comparing with AlphaGo (Silver et al., 2016a), are: 1) it learns from random play, with self-play RL, without human data or supervision; 2) it uses black and white stones from the board as input, without any manual feature engineering; 3) it uses a single neural network to represent both policy and value, rather than separate policy network and value network; and 4) it utilizes the neural network for position evaluation and move sampling for MCTS, and it does not perform Monte Carlo rollouts. AlphaGo Zero deploys several recent achievements in neural networks: residual convolutional neural networks (ResNets), batch normalization, and rectifier nonlinearities.

AlphaGo Zero has three main components in its self-play training pipeline executed in parallel asynchronously: 1) optimize neural network weights from recent self-play data continually; 2) evaluate players continually; 3) use the strongest player to generate new self-play data.

When AlphaGo Zero playing a game against an opponent, MCTS searches from the current state, with the trained neural network weights, to generate move probabilities, and then selects a move.

We present a brief, conceptual pseudo code in Algorithm 12 for training in AlphaGo Zero, conducive for easier understanding.

ALPHA ZERO

Alpha Zero (Silver et al., 2017) generalizes the learning framework in AlphaGo Zero to more domains, in particular, perfect information two-player zero-sum games, with a general reinforcement learning algorithm, learning with no human knowledge except the game rules, and achieves superhuman performance for the games of chess, shogi, and Go.

DISCUSSIONS

AlphaGo Zero is a reinforcement learning algorithm. It is neither supervised learning nor unsupervised learning. The game score is a reward signal, not a supervision label. Optimizing the loss function l is supervised learning. However, it performs policy evaluation and policy improvement, as one iteration in generalized policy iteration.

AlphaGo Zero is not only a heuristic search algorithm. AlphaGo Zero follows a generalized policy iteration procedure, in which, heuristic search, in particular, MCTS, plays a critical role, but within the scheme of reinforcement learning generalized policy iteration, as illustrated in the pseudo code in Algorithm 12. MCTS can be viewed as a policy improvement operator.

AlphaGo Zero has attained a superhuman level perfromance. It may confirm that human professionals have developed effective strategies. However, it does not need to mimic human professional plays. Thus it does not need to predict their moves correctly.

The inputs to AlphaGo Zero include the raw board representation of the position, its history, and the colour to play as 19×19 images; game rules; a game scoring function; invariance of game rules under rotation and reflection, and invariance to colour transposition except for komi.

AlphaGo Zero utilizes 64 GPU workers and 19 CPU parameter servers for training, around 2000 TPUs for data generation, and 4 TPUs for game playing. The computation cost is probably too formidable for researchers with average computation resources to replicate AlphaGo Zero. ELF OpenGo is a reimplementation of AlphaGoZero/AlphaZero using ELF (Tian et al., 2017), at `https://facebook.ai/developers/tools/elf`.

Input: the raw board representation of the position, its history, and the colour to play as 19×19 images; game rules; a game scoring function; invariance of game rules under rotation and reflection, and invariance to colour transposition except for komi
Output: policy (move probabilities) p, value v

initialize neural network weights θ_0 randomly
//AlphaGo Zero follows a generalized policy iteration procedure
for *each iteration* i **do**

// termination conditions:
// 1. both players pass
// 2. the search value drops below a resignation threshold
// 3. the game exceeds a maximum length

initialize s_0
for *each step* t*, until termination at step* T **do**

// MCTS can be viewed as a policy improvement operator
// search algorithm: asynchronous policy and value MCTS algorithm (APV-MCTS)
// execute an MCTS search $\pi_t = \alpha_{\theta_{i-1}}(s_t)$ with previous neural network $f_{\theta_{i-1}}$
// each edge (s, a) in the search tree stores a prior probability $P(s, a)$, a visit count $N(s, a)$,
// and an action value $Q(s, a)$
while *computational resource remains* **do**
select: each simulation traverses the tree by selecting the edge with maximum upper confidence bound $Q(s, a) + U(s, a)$, where $U(s, a) \propto P(s, a)/(1 + N(s, a))$
expand and evaluate: the leaf node is expanded and the associated position s is evaluated by the neural network, $(P(s, \cdot), V(s)) = f_{\theta_i}(s)$; the vector of P values are stored in the outgoing edges from s
backup: each edge (s, a) traversed in the simulation is updated to increment its visit count $N(s, a)$, and to update its action value to the mean evaluation over these simulations, $Q(s, a) = 1/N(s, a) \sum_{s'|s,a \to s'} V(s')$, where $s'|s, a \to s'$ indicates that a simulation eventually reached s' after taking move a from position s
end
// self-play with search can be viewed as a policy evaluation operator: select each move with the improved MCTS-based policy, use the game winner as a sample of the value
play: once the search is complete, search probabilities $\pi \propto N^{1/\tau}$ are returned, where N is the visit count of each move from root and τ is a parameter controlling temperature; play a move by sampling the search probabilities π_t, transition to next state s_{t+1}
end

score the game to give a final reward $r_T \in \{-1, +1\}$
for *each step* t *in the last game* **do**
$z_t \leftarrow \pm r_T$, the game winner from the perspective of the current player
store data as (s_t, π_t, z_t)
end
sample data (s, π, z) uniformly among all time-steps of the last iteration(s) of self-play

//train neural network weights θ_i
//optimizing loss function l performs both policy evaluation, via $(z - v)^2$, and policy improvement, via $-\pi^T \log p$, in a single step
adjust the neural network $(p, v) = f_{\theta_i}(s)$:
to minimize the error between the predicted value v and the self-play winner z, and
to maximize similarity of neural network move probabilities p to search probabilities π
specifically, adjust the parameters θ by gradient descent on a loss function
$(p, v) = f_{\theta_i}(s)$ and $l = (z - v)^2 - \pi^T \log p + c\|\theta_i\|^2$
l sums over the mean-squared error and cross-entropy losses, respectively
c is a parameter controlling the level of $L2$ weight regularization to prevent overfitting

evaluate the checkpoint every 1000 training steps to decide if replacing the current best player (neural network weights) for generating next batch of self-play games
end

Algorithm 12: AlphaGo Zero training pseudo code, based on Silver et al. (2017)

AlphaGo Zero requires huge amount of data for training, so it is still a big data issue. However, the data can be generated by self play, with a perfect model or precise game rules.

Due to the perfect model or precise game rules for computer Go, AlphaGo algorithms have their limitations. For example, in healthcare, robotics and self driving problems, it is usually hard to collect a large amount of data, and it is hard or impossible to have a close enough or even perfect model. As such, it is nontrivial to directly apply AlphaGo Zero algorithms to such applications.

On the other hand, AlphaGo algorithms, especially, the underlying techniques, namely, deep learning, reinforcement learning, Monte Carlo tree search, and self-play, have many applications. Silver et al. (2016a) and Silver et al. (2017) recommend the following applications: general game-playing (in particular, video games), classical planning, partially observed planning, scheduling, constraint satisfaction, robotics, industrial control, and online recommendation systems. AlphaGo Zero blog at `https://deepmind.com/blog/alphago-zero-learning-scratch/` mentions the following structured problems: protein folding, reducing energy consumption, and searching for revolutionary new materials. [7] Several of these, like planning, scheduling, constraint satisfaction, are constraint programming problems (Rossi et al., 2006).

AlphaGo has made tremendous progress, and sets a landmark in AI. However, we are still far away from attaining artificial general intelligence (AGI).

It is interesting to see how strong a deep neural network in AlphaGo can become, i.e., to approximate optimal value function and policy, and how soon a very strong computer Go program would be available on a mobile phone.

15.2 Card Games

Variants of card games, including Texas Hold'em Poker, majiang/mahjong, Skat, etc., are imperfect information games, which, as one type of game theory problems, have many applications, e.g., security and medical decision support (Chen and Bowling, 2012). It is interesting to see more progress of deep RL in such applications, and the full version of Texas Hold'em.

Heads-up Limit Hold'em Poker is essentially solved (Bowling et al., 2015) with counterfactual regret minimization (CFR), which is an iterative method to approximate a Nash equilibrium of an extensive-form game with repeated self-play between two regret-minimizing algorithms.

DeepStack

Recently, significant progress has been made for Heads-up No-Limit Hold'em Poker (Moravčík et al., 2017), the DeepStack computer program defeated professional poker players for the first time. DeepStack utilizes the recursive reasoning of CFR to handle information asymmetry, focusing computation on specific situations arising when making decisions and use of value functions trained automatically, with little domain knowledge or human expert games, without abstraction and offline computation of complete strategies as before. DeepStack follows generalized policy iteration. Watch a talk by Michael Bowling at `https://vimeo.com/212288252`.

It is desirable to see extension of DeepStack to multi-player settings. The current study of Poker limits to a single hand, while there are usually many hands in Poker. It is thus desirable to investigate

[7] There is a blog titled "AlphaGo, in context" in May 2017 by Andrej Karpathy, after AlphaGo defeated Ke Jie, at`https://medium.com/@karpathy/alphago-in-context-c47718cb95a5`. The author characterizes properties of Computer Go as: fully deterministic, fully observable, discrete action space, accessible perfect simulator, relatively short episode/game, clear and fast evaluation conducive for many trail-and-errors, and huge datasets of human play games, to illustrate the narrowness of AlphaGo. (AlphaGo Zero invalidates the last property, "huge datasets of human play games".) It is true that computer Go has limitations in the problem setting and thus for potential applications, and is far from artificial general intelligence. However, we see the success of AlphaGo, in particular, AlphaGo Zero, as the triumph of AI, in particular, the underlying techniques, i.e., deep learning, reinforcement learning, Monte Carlo tree search, and self-play; these techniques are present in many recent achievements in AI. AlphaGo techniques will shed light on classical AI areas, like planning, scheduling, and constraint satisfaction (Silver et al., 2016a), and new areas for AI, like retrosynthesis (Segler et al., 2018). Reportedly, the success of AlphaGo's conquering titanic search space inspired quantum physicists to solve the quantum many-body problem (Carleo and Troyer, 2017).

sequential decision making in cases of multiple hands, and probably treating tournaments and cash games differently.

15.3 VIDEO GAMES

Video games are those with video frames as inputs to RL/AI agents. In video games, information may be perfect or imperfect, and game theory may be deployed or not.

We discuss algorithms for Atari 2600 games, in particular, DQN and its extensions, mostly in Chapter 3, when we talk about value-based methods. We discuss algorithms for StarCraft in Chapter 12, when we talk about multi-agent RL. We discuss Zambaldi et al. (2018), which investigated StarCraft II mini-games with relational deep RL, in Chapter 13. Sun et al. (2018) and Pang et al. (2018) have beaten full-game built-in AI in StarCraft II. We discuss Jaderberg et al. (2018), a human-level agent for Quake III Arena Capture the Flag in Chapter 12. We discuss Mnih et al. (2016) in Section 4.2, and Jaderberg et al. (2017) and Mirowski et al. (2017) in Section 10, which use Labyrinth as the testbed. Oh et al. (2016) and Tessler et al. (2017) study Minecraft. Chen and Yi (2017) and Firoiu et al. (2017) study Super Smash Bros.

Wu and Tian (2017) deploy A3C to train an agent in a partially observable 3D environment, Doom, from recent four raw frames and game variables, to predict next action and value function, following the curriculum learning (Bengio et al., 2009) approach of starting with simple tasks and gradually transition to harder ones. It is nontrivial to apply A3C to such 3D games directly, partly due to sparse and long term reward. The authors won the champion in Track 1 of ViZDoom Competition by a large margin. Dosovitskiy and Koltun (2017) approach the problem of sensorimotor control in immersive environments with supervised learning. Lample and Chaplot (2017) also study Doom.

16 ROBOTICS

Robotics is a classical application area for reinforcement learning. Robots have wide applications, e.g., manufacture, supply chain, healthcare, etc.

Robotics pose challenges to reinforcement learning, including dimensionality, real world examples, under-modelling (models not capturing all details of system dynamics) and model uncertainty, and, reward and goal specification (Kober et al., 2013). RL provides tractable approaches to robotics, through representation, including state-action discretization, value function approximation, and pre-structured policies; through prior knowledge, including demonstration, task structuring, and directing exploration; and through models, including mental rehearsal, which deals with simulation bias, real world stochasticity, and optimization efficiency with simulation samples, and approaches for learned forward models (Kober et al., 2013).

Watch Abbeel (2017a) for a talk on deep learning for robotics. See Kober et al. (2013) for a survey of RL in robotics, Deisenroth et al. (2013) for a survey on policy search for robotics, and Argall et al. (2009) for a survey of robot learning from demonstration. See the journal Science Robotics. It is interesting to note that from NIPS 2016 Invited Talk titled Dynamic Legged Robots by Marc Raibert, Boston Dynamics robots did not use machine learning. See NIPS 2018 Workshop on Imitation Learning and its Challenges in Robotics at `https://sites.google.com/view/nips18-ilr`.

Ng et al. (2004) study autonomous helicopter. Reddy et al. (2018) study glider soaring. Mirowski et al. (2017), Banino et al. (2018), and Wayne et al. (2018) propose methods for learning to navigate. Liu and Tomizuka (2016; 2017) study how to make robots and people to collaborate to achieve both flexibility and productivity in production lines. See a blog at `http://bair.berkeley.edu/blog/2017/12/12/corobots/`.

In the following, we discuss sim-to-real, imitation learning, value-based learning, policy-based learning, and model-based learning for robotics. Then we discuss autonomous driving vehicles, a special type of robots.

16.1 SIM-TO-REAL

It is easier to train a robot in simulation than in reality. Most RL algorithms are sample intensive. And exploration may cause risky policies to the robot and/or the environment. However, a simulator usually can not precisely reflect the reality. How to bridge the gap between simulation and reality, i.e., sim-to-real, is critical and challenging in robotics. Sim-to-real is a special type of transfer learning, as discussed in Section 14.2.

Peng et al. (2017c) propose to use dynamics randomization to train recurrent policies in simulation, and deploy the policies directly on a physical robot, achieving good performance on an object pushing task, without calibration. This work does not consider visual observation.

OpenAI (2018) propose to learn dexterity of in-hand manipulation to perform object reorientation for a physical Shadow Dexterous Hand, using Proximal Policy Optimization (PPO), with dynamics randomization in simulation, and transferring the learned policy directly to physical hand, sharing code base with OpenAI Five for playing Dota 2. See a blog with video at `https://blog.openai.com/learning-dexterity/`. Chen et al. (2018b), Popov et al. (2017) and Pérez-D'Arpino and Shah (2017) also study dexterity learning. See also `https://bair.berkeley.edu/blog/2018/08/31/dexterous-manip/`.

Rusu et al. (2017) propose to use progressive networks (Rusu et al., 2016) to bridge the reality gap. In progressive networks, lateral connections connect layers of network columns learned previously to each new column, to support compositionality, and to support transfer learning and domain adaptation. In a progressive network, columns may not have the same capacity or structure, so that the column for simulation can have sufficient capacity, and the column for reality may have lower capacity, which can be initialized from the column trained with simulation, to encourage exploration and fast learning from scarce real data. Rusu et al. (2017) use MuJoCo physics simulator to train the first column, for a reaching task with the modelled Jaco; and use real Jaco to train the second column with RGB images, to be deployed on a real robot. The authors also propose to handle dynamic tasks,

e.g., dynamic targets, by adding a third column and proprioceptive features, i.e., features for joint angles and velocities for arms and fingers.

Sadeghi et al. (2018) propose a convolutional recurrent neural network to learn perception and control for servoing a robot arm to desired objects, invariant to viewpoints. The proposed method uses the memory of past movements to select actions to reach the target, rather than assuming known dynamics or requiring calibration. The authors propose to learn the controller with simulated demonstration trajectories and RL. The supervised demonstration learning usually learns a myopic policy, with distance to goal as the objective; and RL helps learn a policy consider long term effect, by evaluating the action value function to assess if the goal can be reached. The visual layers are adapted with a small amount of realistic images for better transfer performance. The work assumes that the robot can move to an object directly, without planning for a sophisticated policy, e.g., with obstacles.

See also Bousmalis et al. (2017), and a blog, `https://research.googleblog.com/2017/10/closing-simulation-to-reality-gap-for.html`

16.2 Imitation Learning

Finn et al. (2016b) study inverse optimal cost, or inverse reinforcement learning in control. The authors propose to utilize nonlinear cost functions, such as neural networks, to impose structures on the cost for informative features and effective regularization, and approximate MaxEnt (Ziebart et al., 2008) with samples for learning with unknown dynamics in high-dimensional continuous environments.

Duan et al. (2017) propose one-shot imitation learning, in the setting with many tasks, as supervised learning. The authors train a neural network with pairs of demonstrations for a subset of tasks, with input as the first demonstration and a state sampled from the second demonstration, and predicted the action for the sampled state. The authors utilize the soft attention to process sequence of states and actions of a demonstration, and vector components for block locations, as in the block stacking experiments, for better generalization to unseen conditions and tasks in training data. See videos at `https://sites.google.com/view/nips2017-one-shot-imitation/`.

Finn et al. (2017c) and Yu et al. (2018) propose one-shot meta-imitation learning methods to build vision-based policies fine-tuned end-to-end from one demonstration, using model-agnostic meta-learning (MAML) (Finn et al., 2017a) for pre-training on a diverse range of demonstrations from other environments. Usually learning from raw pixels requires a large amount of data. MAML optimizes an initial representation for a learning algorithm, to build a rich prior in the meta-learning phase, allowing the parameters to adapt to new tasks with a few examples. We discuss MAML in Section 14.1. In Finn et al. (2017c), demonstrations come from a teleoperated robot. In Yu et al. (2018), demonstrations come from a human, posing a challenging issue of domain shift. Yu et al. (2018) learn how to learn from both human and teleoperated demonstrations, and could adapt to a new task with one human demonstration. Both Finn et al. (2017c) and Yu et al. (2018) experiment with both simulation and physical robots. See a blog at `http://bair.berkeley.edu/blog/2018/06/28/daml/`. The open source for Finn et al. (2017c) is at `https://github.com/tianheyu927/mil`.

16.3 Value-based Learning

Here we discuss two papers using successor representation (SR). We introduce that a value function can be decomposed into environment dynamics (SR) and reward signal in Chapter 8. We discuss general value function (GVF) in Section 3.3, e.g., Sutton et al. (2011), universal function approximators (UVFAs) (Schaul et al., 2015), and, hindsight experience replay (HER) (Andrychowicz et al., 2017). Sutton et al. (2011) and Andrychowicz et al. (2017) experiment with robots. Barreto et al. (2017) extend successor representation to successor features for continuous tasks with function approximation, as discussed in Section 14.2.

Zhang et al. (2017) propose a deep RL approach with successor features for robot navigation tasks, for transferring knowledge obtained from previous navigation tasks to new ones, without localization, mapping or planning. The authors experiment with both simulation and physical robots.

Sherstan et al. (2018) propose to accelerate construction of knowledge represented by GVFs with successor representation in a continual learning setting, so that learning new GVFs are sped up with previous learned GVFs.

16.4 Policy-based Learning

Levine et al. (2016) propose to train perception and control systems jointly end-to-end, to map raw image observations directly to torques at the robot's motors. The authors introduce guided policy search (GPS) to train policies represented by CNNs, by transforming policy search into supervised learning to achieve data efficiency, with training data provided by a trajectory-centric RL method operating under unknown dynamics. GPS alternates between trajectory-centric RL and supervised learning, to obtain the training data coming from the policy's own state distribution, to address the issue that supervised learning usually does not achieve good, long-horizon performance. GPS utilizes pre-training to reduce the amount of experience data to train visuomotor policies. Good performance is achieved on a range of real-world manipulation tasks requiring localization, visual tracking, and handling complex contact dynamics, and simulated comparisons with previous policy search methods. As the authors mention, "this is the first method that can train deep visuomotor policies for complex, high-dimensional manipulation skills with direct torque control".

Yahya et al. (2017) propose a distributed and asynchronous guided policy search for a vision-based door opening task with four robots.

Zhu et al. (2017b) propose target-driven visual navigation in indoor scenes with deep RL by treating the policy as a function of both the goal and the current state in an actor-critic model, to generalize over targets, similar to general value function. The authors design the house of interactions (AI2-THOR), a simulation framework with high-quality 3D scenes and physics engine, which allows visual interactions for agents, to generate large amount of training data. The authors qualitatively compare AI2-THOR with other simulators, like ALE, VizDoom, UETorch, Project Malmo, SceneNet, TORCS, SYTHNIA, and, Virtual KITTI.

16.5 Model-based Learning

Gu et al. (2016) propose normalized advantage functions (NAF) to enable experience replay with Q-learning for continuous task, and to refit local linear models iteratively. Gu et al. (2017a) propose asynchronous NAF algorithm, with safety constraints, for 3D manipulation in simulation and door opening for real robots.

Finn and Levine (2017) propose to combine model predictive control (MPC) with action-conditioned video prediction, in a self-supervised approach without labeled training data, for physical robotic manipulation of previously unseen objects.

Chebotar et al. (2017) focus on time-varying linear-Gaussian policies, and integrated a model-based linear quadratic regulator (LQR) algorithm with a model-free path integral policy improvement algorithm. To generalize the method for arbitrary parameterized policies such as deep neural networks, the authors combined the proposed approach with guided policy search (GPS) (Levine et al., 2016).

Lee et al. (2017) study visual servoing by combining features learned from object classification, predictive dynamic models, and fitted Q-iteration algorithm.

16.6 Autonomous Driving Vehicles

Autonomous driving is an important topic of intelligent transportation systems as we discuss in Chapter 20.4. O'Kelly et al. (2018) propose to test autonomous vehicles rare-event simulation. Fridman et al. (2018) propose DeepTraffic, a micro-traffic simulator, for deep RL. Yu et al. (2018) release BDD100K, a large-scale diverse driving video database. The data is available at, `http://bdd-data.berkeley.edu`. The blog is at, `http://bair.berkeley.edu/blog/2018/05/30/bdd/`. See also Bojarski et al. (2016), Bojarski et al. (2017), Zhou and Tuzel (2018). See a website for Tesla Autopilot Miles at, `https://hcai.mit.edu/tesla-autopilot-miles/`.

We argue that the current science, engineering, and technology, including AI, are not ready for road test of fully autonomous driving vehicles yet. One issue is adversarial examples, as we discuss in Section 20.5. From the Insurance Institute for Highway Safety (IIHS) website, `https://www.iihs.org/iihs/topics/t/general-statistics/fatalityfacts/state-by-state-overview`, we learn that human drivers in US encounter roughly one death per 100 million vehicle miles. Consider, for an autonomous system, how many decisions to make, and what level of accuracy are required, for just one second. Simulators are probably far from reality; and road test data collected so far are far from for justification of statistically significant level claims. Road tests are actually experiments, with humans involved. For a "self-driving vehicle" to run on roads, a feasible approach is to require a driver to sit on the driver's seat, pay attention to the driving, and take control of the vehicle at any time if not always. For fully self driving vehicles, we propose to conduct "road tests" in a closed environment, using robots as pedestrians, etc., until AI has sufficient intelligence, e.g., understanding scenes, with common sense, etc.

17 NATURAL LANGUAGE PROCESSING

Natural language processing (NLP) learns, understands, and produces human language content using computational techniques (Hirschberg and Manning, 2015).

There are many interesting topics in NLP. we discuss sequence generation in Section 17.1, machine translation in Section 17.2, and, dialogue systems in Section 17.3. We list some NLP topics in the following. We also list the integration of computer vision with NLP, like visual captioning, visual dialog, and visual relationship and attribute detection, which we discuss in Section 18.4.

- language tree-structure learning, e.g., Socher et al. (2011; 2013); Yogatama et al. (2017);
- semantic parsing, e.g., Liang et al. (2017b);
- question answering, e.g., Shen et al. (2017), Trischler et al. (2016), Xiong et al. (2017a), Wang et al. (2017b), Choi et al. (2017)
- summarization, e.g., Chopra et al. (2016); Paulus et al. (2017); Zhang and Lapata (2017)
- sentiment analysis (Liu, 2012; Zhang et al., 2018)
- information retrieval (Manning et al., 2008), e.g., and Mitra and Craswell (2017), Wang et al. (2017a), Zhang et al. (2016a)
- information extraction, e.g., Narasimhan et al. (2016);
- visual captioning, e.g., Wang et al. (2018e); Xu et al. (2015); Liu et al. (2016); Lu et al. (2016); Ren et al. (2017); Pasunuru and Bansal (2017); Wang et al. (2018f);
- visual dialog, e.g., Das et al. (2017); Strub et al. (2017);
- visual relationship and attribute detection, e.g., Liang et al. (2017c);
- visual question answering, e.g., Hudson and Manning (2018)
- popular Reddit threads prediction, e.g., He et al. (2016c)
- automatic query reformulation, e.g., Nogueira and Cho (2017);
- language to executable program, e.g., Guu et al. (2017);
- knowledge graph reasoning, e.g., Xiong et al. (2017c);
- text games, e.g., Wang et al. (2016a), He et al. (2016b), Narasimhan et al. (2015);
- semi-supervised learning, co-training, e.g., Wu et al. (2018).

Deep learning has been permeating into many subareas in NLP, and helping make significant progress. It appears that NLP is still a field more about synergy than competition, for deep learning vs. non-deep learning algorithms, and for approaches based on no domain knowledge vs. with linguistics knowledge. Some non-deep learning algorithms are effective and perform well, e.g., word2vec (Mikolov et al., 2013; Mikolov et al., 2017) and fastText (Joulin et al., 2017), and many papers study syntax and semantics of languages, with a recent example in semantic role labeling (He et al., 2017). Some deep learning approaches to NLP problems incorporate explicitly or implicitly linguistics knowledge, e.g., Socher et al. (2011; 2013); Yogatama et al. (2017). Manning (2017) discusses computational linguistics and deep learning. See ACL 2018 Workshop on Relevance of Linguistic Structure in Neural NLP, at `https://sites.google.com/view/relsnnlp`.

McCann et al. (2018) propose natural language decathlon (decaNLP), an NLP benchmark suitable for multitask, transfer, and continual learning. See the website, `http://decanlp.com`. Devlin et al. (2018) propose Bidirectional Encoder Representations from Transformers (BERT) for language representation model pre-training.

Melis et al. (2018) investigate the evaluation in neural language models, and observe that standard LSTM outperforms recent models. Bai et al. (2018) show empirically that CNNs outperforms RNNs over a wide range of tasks.

See Jurafsky and Martin (2017), Goldberg (2017), Deng and Liu (2018) for books on NLP; Hirschberg and Manning (2015); Cho (2015); Young et al. (2017) for surveys on NLP; Deng and Li (2013); Gao et al. (2018a); Hinton et al. (2012); He and Deng (2013); Young et al. (2013) for surveys

on dialogue systems; Neubig (2017) for a tutorial on neural machine translation and sequence-to-sequence models, Agarwal et al. (2018) for a tutorial on end to-end goal-oriented question answering systems, and, Monroe (2017) for a gentle introduction to translation.

See three ACL 2018 tutorials: Wang et al. (2018d) for deep reinforcement learning for NLP, Gao et al. (2018b) for neural approaches to conversational AI (see also Gao et al. (2018a)), and, Anderson et al. (2018) for connecting language and vision to actions.

See several workshops: NIPS Workshop on Conversational AI: Today's Practice and Tomorrow's Potential, in 2018 at `http://alborz-geramifard.com/workshops/nips18-Conversational-AI/`, and, in 2017 at `http://alborz-geramifard.com/workshops/nips17-Conversational-AI/`; NIPS 2018 Workshop on Wordplay: Reinforcement and Language Learning in Text-based Games; NIPS 2016 Workshop on End-to-end Learning for Speech and Audio Processing, at `https://sites.google.com/site/nips2016endtoendspeechaudio/`; and, NIPS 2015 Workshop on Machine Learning for Spoken Language Understanding and Interactions, at `http://slunips2015.wixsite.com/slunips2015`.

17.1 Sequence Generation

A sequence may take the form of text, music, and molecule, etc. Sequence generation techniques may be applicable to multiple domains, e.g., Jaques et al. (2017) experiment with musical melody and computational molecular generation. Here we focus on text generation, which is the basis for many NLP problems, like conversational response generation, machine translation, abstractive summarization, etc.

Text generation models are usually based on n-gram, feed-forward neural networks, or recurrent neural networks, trained to predict next word given the previous ground truth words as inputs; then in testing, the trained models are used to generate a sequence word by word, using the generated words as inputs. The errors will accumulate on the way, causing the exposure bias issue. Moreover, these models are trained with word level losses, e.g., cross entropy, to maximize the probability of next word; however, the models are evaluated on different metrics like BLEU.

Ranzato et al. (2016) propose mixed incremental cross-entropy reinforce (MIXER) for sequence prediction, with incremental learning and a loss function combining both REINFORCE and cross-entropy. MIXER is a sequence level training algorithm, aligning training and testing objective, such as BLEU, rather than predicting the next word as in previous papers.

Bahdanau et al. (2017) propose an actor-critic algorithm for sequence prediction, to improve Ranzato et al. (2016). The authors utilize a critic network to predict the value of a token, i.e., the expected score following the sequence prediction policy, defined by an actor network, trained by the predicted value of tokens. Some techniques are deployed to improve performance: SARSA rather than Monter-Carlo method to lessen the variance in estimating value functions; target network for stability; sampling prediction from a delayed actor whose weights are updated more slowly than the actor to be trained, to avoid the feedback loop when actor and critic need to be trained based on the output of each other; and, reward shaping to avoid the issue of sparse training signal.

Yu et al. (2017) propose SeqGAN, sequence generative adversarial nets with policy gradient, integrating the adversarial scheme in Goodfellow et al. (2014).

17.2 Machine Translation

Neural machine translation (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015) utilizes end-to-end deep learning for machine translation, and becomes dominant, against the traditional statistical machine translation techniques. In sequence to sequence model (Cho et al., 2014; Sutskever et al., 2014), an input sequence of symbol representation is encoded to a fix-length vector, which is then decoded to symbols one by one, in an auto-regressive way, using symbols generated previously as additional input. Bahdanau et al. (2015) introduce a soft-attention technique to address the issues with encoding the whole sentence into a fix-length vector.

He et al. (2016a) propose dual learning mechanism to tackle the data hunger issue in machine translation, inspired by the observation that the information feedback between the primal, translation from language A to language B, and the dual, translation from B to A, can help improve both translation models, with a policy gradient method, using the language model likelihood as the reward signal. Experiments show, with only 10% bilingual data for warm start and monolingual data, the dual learning approach perform comparably with previous neural machine translation methods with full bilingual data in English to French tasks. The dual learning mechanism may be extended to other tasks, if a task has a dual form, e.g., speech recognition and text to speech, image caption and image generation, question answering and question generation, search and keyword extraction, etc. Xia et al. (2018) study model-level dual learning.

See Wu et al. (2016), Johnson et al. (2017) for Google's Neural Machine Translation System, Gehring et al. (2017) for convolutional sequence to sequence learning for fast neural machine translation, Klein et al. (2017) for OpenNMT, an open source neural machine translation system, at `http://opennmt.net`, Cheng et al. (2016) for semi-supervised learning for neural machine translation, Wu et al. (2017c) for adversarial neural machine translation, Vaswani et al. (2017) for a new approach for translation replacing ConvNets and RNN with self attention and positional encoding, open source at `https://github.com/tensorflow/tensor2tensor`, Dehghani et al. (2018) for an extension of Vaswani et al. (2017) at `https://goo.gl/72gvdq`, Artetxe et al. (2018) for an unsupervised approach to machine translation, and, Zhang et al. (2017) for an open source toolkit for neural machine translation.

17.3 Dialogue Systems

In dialogue systems, conversational agents, or chatbots, human and computer interacts with a natural language. We intentionally remove "spoken" before "dialogue systems" to accommodate both spoken and written language user interface (UI). Jurafsky and Martin (2017) categorize dialogue systems as task-oriented dialog agents and chatbots; the former are set up to have short conversations to help complete particular tasks; the latter are set up to mimic human-human interactions with extended conversations, sometimes with entertainment value. As in Deng (2017), there are four categories: social chatbots, infobots (interactive question answering), task completion bots (task-oriented or goal-oriented) and personal assistant bots. We have seen generation one dialogue systems: symbolic rule/template based, and generation two: data driven with (shallow) learning. We are now experiencing generation three: data driven with deep learning, in which reinforcement learning usually plays an important role. A dialogue system usually include the following modules: (spoken) language understanding, dialogue manager (dialogue state tracker and dialogue policy learning), and a natural language generation (Young et al., 2013). In task-oriented systems, there is usually a knowledge base to query. A deep learning approach, as usual, attempts to make the learning of the system parameters end-to-end. See Deng (2017) for more details. See a survey paper on applying machine learning to speech recognition (Deng and Li, 2013).

Dhingra et al. (2017) propose KB-InfoBot, a goal-oriented dialogue system for multi-turn information access. KB-InfoBot is trained end-to-end using RL from user feedback with differentiable operations, including those for accessing external knowledge database (KB). In previous work, e.g., Li et al. (2017) and Wen et al. (2017), a dialogue system accesses real world knowledge from KB by symbolic, SQL-like operations, which is non-differentiable and disables the dialogue system from fully end-to-end trainable. KB-InfoBot achieves the differentiability by inducing a soft posterior distribution over the KB entries to indicate which ones the user is interested in. The authors design a modified version of the episodic REINFORCE algorithm to explore and learn both the policy to select dialogue acts and the posterior over the KB entries for correct retrievals.The authors deploy imitation learning from rule-based belief trackers and policy to warm up the system.

Su et al. (2016) propose an on-line learning framework to train the dialogue policy jointly with the reward model via active learning with a Gaussian process model, to tackle the issue that it is unreliable and costly to use explicit user feedback as the reward signal. The authors show empirically that the proposed framework reduces manual data annotations significantly and mitigates noisy user feedback in dialogue policy learning.

Li et al. (2016) propose to use deep RL to generate dialogues to model future reward for better informativity, coherence, and ease of answering, to attempt to address the issues in the sequence

to sequence models based on Sutskever et al. (2014): the myopia and misalignment of maximizing the probability of generating a response given the previous dialogue turn, and the infinite loop of repetitive responses. The authors design a reward function to reflect the above desirable properties, and deploy policy gradient to optimize the long term reward. It would be interesting to investigate the reward model with the approach in Su et al. (2016) or with inverse RL and imitation learning as discussed in Chapter 5, although Su et al. (2016) mention that such methods are costly, and humans may not act optimally.

Tang et al. (2018a) propose subtask discovery for hierarchical dialogue policy learning based on a dynamic programming approach to segmentation, extending Peng et al. (2017a), which assumes subtasks are defined by experts. Williams et al. (2017) propose to combine an RNN with domain knowledge to improve data efficiency of dialog training. Lewis et al. (2017) study end-to-end learning for negotiation dialogues; open source at `https://github.com/facebookresearch/end-to-end-negotiator`. Zhang et al. (2016b) study end-to-end speech recognition with CNNs. Xiong et al. (2017) describe Microsoft's conversational speech recognition system in 2017. Zhou et al. (2018) propose an emotional chatting machine. Li et al. (2017) present an end-to-end task-completion neural dialogue system with parameters learned by supervised and reinforcement learning. See the open source at `http://github.com/MiuLab/TC-Bot`. See Serban et al. (2018) for a survey of corpora for building dialogue systems.

See more recent papers: Asri et al. (2016), Bordes et al. (2017), Chen et al. (2016b), Fatemi et al. (2016), Kandasamy et al. (2017), Li et al. (2017a), Li et al. (2017b), Lipton et al. (2018), Mesnil et al. (2015), Mo et al. (2018), Saon et al. (2016), She and Chai (2017), Xiong et al. (2017b), Zhao and Eskenazi (2016).

18 COMPUTER VISION

Computer vision is about how computers gain understanding from digital images or videos. Computer vision has been making rapid progress recently, and deep learning plays an important role. We discuss briefly recent progress of computer vision below.

Krizhevsky et al. (2012) propose AlexNet, almost halving the error rate of an ImagetNet competition task, and ignite this wave of deep learning/AI. He et al. (2016d) propose residual nets (ResNets) to ease the training of very deep neural networks by adding shortcut connections to learn residual functions with reference to the layer inputs. Fast R-CNN (Girshick, 2015), Faster R-CNN (Ren et al., 2015), and Mask R-CNN (He et al., 2017) are proposed for image segmentation. Facebook AI Research (FAIR) open source Detectron for object detection algorithms, `https://research.fb.com/downloads/detectron/`.

Generative adversarial networks (GANs) (Goodfellow et al., 2014; Goodfellow, 2017) attracts lots of attention recently. There are fundamental work to improve the stability of learning GANs, e.g., Wasserstein GAN (WGAN) (Arjovsky et al., 2017), Gulrajani et al. (2017), and Least Squares GANs (LSGANs) (Mao et al., 2016). Many proposals in GANs are using computer vision testbeds, e.g., CycleGAN (Zhu et al., 2017a), DualGAN (Yi et al., 2017), and Shrivastava et al. (2017).

For disentangled factor learning, many papers use computer vision testbeds. Diederik P Kingma (2014) propose variational autoencoders (VAEs). Kulkarni et al. (2015) propose deep convolution inverse graphics network (DC-IGN), which follows a semi-supervised way. Chen et al. (2016a) propose InfoGAN, an information-theoretic extension to the GANs, following an unsupervised way. Higgins et al. (2017) propose β-VAE to automatically discover interpretable, disentangled, factorised, latent representations from raw images in an unsupervised way. When $\beta = 1$, β-VAE is the same as VAEs. Eslami et al. (2016) propose the framework of Attend-Infer-Repeat for efficient inference in structured image models to reason about objects explicitly. Zhou et al. (2015) show that object detectors emerge from learning to recognize scenes, without supervised labels for objects.

Reinforcement learning is an effective tool for many computer vision problems, like classification, e.g. Mnih et al. (2014), detection, e.g. Caicedo and Lazebnik (2015), captioning, e.g. Xu et al. (2015), etc. RL is an important ingredient for interactive perception (Bohg et al., 2017), where perception and interaction with the environment would be helpful to each other, in tasks like object segmentation, articulation model estimation, object dynamics learning, haptic property estimation, object recognition or categorization, multimodal object model learning, object pose estimation, grasp planning, and manipulation skill learning. See Rhinehart et al. (2018) for a tutorial on inverse reinforcement learning for computer vision.

Malik (2018) discusses that there are great achievements in the fields of vision, motor control, and language semantic reasoning, and it is time to investigate them together. Zhang and Zhu (2018) survey visual interpretability for deep learning. Lucid, at `https://github.com/tensorflow/lucid`, is an open source for interpretability, implementing feature visualization techniques in Olah et al. (2017). Olah et al. (2018) discuss building blocks of interpretability.

In the following, we discuss recognition in Section 18.1, motion analysis in Section 18.2, scene understanding in Section 18.3, integration with NLP in Section 18.4, visual control in Section 18.5, and interactive perception in Section 18.6.

We list more topics about applying deep RL to computer vision as follows: Liu et al. (2017) for semantic parsing of large-scale 3D point clouds, Devrim Kaba et al. (2017) for view planning, which is a set cover problem, Cao et al. (2017) for face hallucination, i.e., generating a high-resolution face image from a low-resolution input image, Brunner et al. (2018) for learning to read maps, Cubuk et al. (2018) for data augmentation for images, Bhatti et al. (2016) for SLAM-augmented DQN.

18.1 RECOGNITION

RL can improve efficiency for image classification by focusing only on salient parts. For visual object localization and detection, RL can improve efficiency over approaches with exhaustive spatial hypothesis search and sliding windows, and strike a balance between sampling more regions for better accuracy and stopping the search when sufficient confidence is obtained about the target's location.

Mnih et al. (2014) introduce the recurrent attention model (RAM), which we discuss in Chapter 9. Caicedo and Lazebnik (2015) propose an active detection model for object localization with DQN, by deforming a bounding box with transformation actions to determine the most specific location for target objects. Jie et al. (2016) propose a tree-structure RL approach to search for objects sequentially, considering both the current observation and previous search paths, by maximizing the long-term reward associated with localization accuracy over all objects with DQN. Mathe et al. (2016) propose to use policy search for visual object detection. Kong et al. (2017) deploy collaborative multi-agent RL with inter-agent communication for joint object search. Welleck et al. (2017) propose a hierarchical visual architecture with an attention mechanism for multi-label image classification. Rao et al. (2017) propose an attention-aware deep RL method for video face recognition. Krull et al. (2017) study 6D object pose estimation.

18.2 Motion Analysis

In tracking, an agent needs to follow a moving object. Supančič and Ramanan (2017) propose online decision-making process for tracking, formulate it as a partially observable decision-making process (POMDP), and learn policies with deep RL algorithms, to decide where to look for the object, when to reinitialize, and when to update the appearance model for the object, where image frames may be ambiguous and computational budget may be constrained. Yun et al. (2017) also study visual tracking with deep RL.

Rhinehart and Kitani (2017) propose to discover agent rewards for K-futures online (DARKO) to model and forecast first-person camera wearer's long-term goals, together with states, transitions, and rewards from streaming data, with inverse RL.

18.3 Scene Understanding

Wu et al. (2017b) study the problem of scene understanding, and attempt to obtain a compact, expressive, and interpretable representation to encode scene information like objects, their categories, poses, positions, etc, in a semi-supervised way. In contrast to encoder-decoder based neural architectures as in previous work, Wu et al. (2017b) propose to replace the decoder with a deterministic rendering function, to map a structured and disentangled scene description, scene XML, to an image; consequently, the encoder transforms an image to the scene XML by inverting the rendering operation, a.k.a., de-rendering. The authors deploy a variant of REINFORCE to overcome the non-differentiability issue of graphics rendering engines.

Wu et al. (2017a) propose a paradigm with three major components, a convolutional perception module, a physics engine, and a graphics engine, to understand physical scenes without human annotations. The perception module recovers a physical world representation by inverting the graphics engine, inferring the physical object state for each segment proposal in input and combining them. The generative physics and graphics engines then run forward with the world representation to reconstruct the visual data. The authors show results on both neural, differentiable and more mature but non-differentiable physics engines.

We discuss generative query network (GQN) (Eslami et al., 2018) in Chapter 10. Chen et al. (2018a) propose a framework for iterative visual reasoning, which we discuss in Chapter 13. There are recent papers about physics learning, e.g., Agrawal et al. (2016); Battaglia et al. (2016); Denil et al. (2017); Watters et al. (2017); Wu et al. (2015).

18.4 Integration with NLP

Some papers integrate computer vision with natural language processing (NLP). Xu et al. (2015) integrate attention to image captioning. See also Liu et al. (2016), Lu et al. (2016), Rennie et al. (2017), and Ren et al. (2017) for image captioning. See Pasunuru and Bansal (2017); Wang et al. (2018f) for video captioning. Strub et al. (2017) propose end-to-end optimization with deep RL for goal-driven and visually grounded dialogue systems for the GuessWhat?! game. Das et al. (2017) propose to learn cooperative visual dialog agents with deep RL. Wang et al. (2018e) propose to use inverse RL for visual storytelling. See also Kottur et al. (2017). See Liang et al. (2017c) for visual relationship and attribute detection.

18.5 VISUAL CONTROL

Visual control is about deriving a policy from visual inputs, e.g., in games (Mnih et al., 2015; Silver et al., 2016a; 2017; Oh et al., 2015; Wu and Tian, 2017; Dosovitskiy and Koltun, 2017; Lample and Chaplot, 2017; Jaderberg et al., 2017), robotics (Finn and Levine, 2017; Gupta et al., 2017b; Lee et al., 2017; Levine et al., 2016; Mirowski et al., 2017; Zhu et al., 2017b), and self-driving vehicles (Bojarski et al., 2016; Bojarski et al., 2017; Zhou and Tuzel, 2018).

For a visual control problem in computer vision, there should be some ingredients of, by, for computer vision, but not just use a CNN or some deep neural network to take image or video as input, without further handling with computer vision techniques, e.g., DQN (Mnih et al., 2015) and AlphaGo (Silver et al., 2016a; 2017).

18.6 INTERACTIVE PERCEPTION

Reinforcement learning is an important ingredient for interactive perception (Bohg et al., 2017). Jayaraman and Grauman (2018) propose a deep RL approach with recurrent neural network for active visual completion, to hallucinate unobserved parts based on a small number of observations. The authors attempt to answer the question of how to make decisions about what to observe to acquire more information in visual perception, without labeled data, rather than making inference decisions based on labeled observations. The look-around decisions are rewarded based on the accuracy of the predictions of unobserved views, in particular, the distance between view predictions and their ground truth for all viewpoints and all time steps. The authors propose a task agnostic approach for active visual completion, and, consider two tasks: panoramic natural scenes and 3D object shapes, for illustration. The authors also discuss generalization and transferability of their proposed approach to new tasks and environments.

19 FINANCE AND BUSINESS MANAGEMENT

Machine learning naturally has wide applications in finance, e.g., in fundamental analysis, behavioural finance, technical analysis, financial engineering, financial technology (FinTech), etc. Reinforcement learning is a natural solution to some sequential decision making finance problems, like option pricing, trading, and multi-period portfolio optimization, etc. RL also has many applications in business management, like ads, recommendation, customer management, and marketing, etc.

Financial engineering is a discipline rooting in finance, computer science and mathematics (Hull, 2014; Luenberger, 1997). Derivative pricing is an essential issue in financial engineering. The values of financial derivatives depend on the values of underlying assets. Options are the most fundamental derivatives. An option gives the holder the right, but not the obligation, to buy or sell an asset at a certain price by a certain time. Portfolio optimization is about how to allocate assets so as to trade off between return and risk.

There are two schools in finance, Efficient Markets Hypothesis (EMH) and Behavioral Finance (Lo, 2004). According to EMH, "prices fully reflect all available information" and are determined by the market equilibrium. However, psychologists and economists have found a number of behavioral biases that are native in human decision-making under uncertainty. For example, Amos Tversky and Daniel Kahneman demonstrate the phenomenon of loss aversion, in which people tend to strongly prefer avoiding losses to acquiring gains (Kahneman, 2011). Prashanth et al. (2016) investigate prospect theory with reinforcement learning. Behavioral finance justifies technical analysis (Murphy, 1999) to some extend. Lo et al. (2000) propose to use nonparametric kernel regression for automatic technical pattern recognition. Lo (2004) proposes the Adaptive Market Hypothesis to reconcile EMH and behavioral finance, where the markets are in the evolutionary process of competition, mutation, reproduction and natural selection. RL may play an important role in this fundamental market paradigm.

Financial technology (FinTech) has been attracting lots of attention, especially after the notion of big data and data science. FinTech employs machine learning techniques to deal with issues like fraud detection (Phua et al., 2010), and consumer credit risk (Khandani et al., 2010), etc.

We will discuss applications of deep learning and reinforcement learning to finance and business management. We only pick a couple of papers for discussions, and do not include many relevant papers. Machine learning techniques, like support vector machines (SVM), decision trees, etc, have also been applied to finance and business management. We can check them from the reference in the papers we discuss.

It is nontrivial for finance and economics academia to accept machine learning methods like neural networks. One factor is that neural networks, among many machine learning methods, are black box; however, interpretability is desirable in finance. Doshi-Velez and Kim (2017) and Lipton (2018) discuss issues of interpretability. Zhang and Zhu (2018) is a survey about interpretability in computer vision. National Bereau Economic Research (NBER) organizes a meeting on Economics of Artificial Intelligence; see `http://conference.nber.org/conferences/2018/AIf18/summary.html`. See Mullainathan (2017) for a lecture in American Finance Association (AFA) 2017 annual meeting about machine learning and prediction in economics and finance. In 2018 ASSA Annual Meeting, at `https://www.aeaweb.org/conference/2018`, AFA as part of it, we see many sessions with the keywords "artificial intelligence" and/or "machine learning". We see this as that AI and machine learning are starting to permeate to the mainstream of the field of finance and economics. It would be natural for economics and finance to marry reinforcement learning, machine learning, and AI, considering that quantitative approaches for economics and finance share foundations of optimization, statistics, and probability with RL/ML/AI, and, behavioural approaches for economics and finance share foundations of psychology, neuroscience, and cognitive science with RL/ML/AI. We will see more and more applications of RL/ML/AI in finance, economics, and social sciences in general. See NIPS 2018 Workshop on Challenges and Opportunities for AI in Financial Services: the Impact of Fairness, Explainability, Accuracy, and Privacy.

Before discussing applications of RL to finance and business management, we introduce finance applications with deep learning. Deep learning has a wide applications in finance, e.g., company fundamentals prediction (Alberg and Lipton, 2017), macroeconomic indicator forecasting (Cook and Hall, 2017), and limit order books (Sirignano, 2016), etc.

Heaton et al. (2016) introduce deep learning for finance, in particular, deep portfolios, and argue that deep learning methods may be more powerful than the current standard methods in finance., e.g., the simplistic traditional financial economics linear factor models and statistical arbitrage asset management techniques. The authors show the power of deep learning with a case study of smart indexing for the biotechnology IBB index with a four step algorithm: 1) auto-encoding, find the market-map to auto-encode the input variables with itself and to create a more efficient representation of the input variables; 2) calibrating, find the portfolio-map to create a portfolio from the input variables to approximate an objective; 3) validating, balance the errors in the auto-encoding and calibrating steps; and 4) verifying, choose market-map and portfolio-map to satisfy the validating step. The authors make an interesting observation that the univariate activation functions such as ReLU, i.e., $max(0, x)$, where x is a variable, in deep learning can be interpreted as compositions of financial call and put options on linear combination of input assets.

Bao et al. (2017) investigate the problem of stock price forecasting by combining wavelet transforms (WT), stacked auto-encoders (SAEs) and long-short term memory (LSTM): WT for decomposing stock price time series to reduce noises, SAEs for generating high-level features for stock price prediction, and LSTM for stock price forecasting by taking the denoised features. The authors evaluate the performance of the proposed method with six market indices and their corresponding index futures, together with a buy-and-sell trading strategy, using three performance metrics: Mean absolute percentage error (MAPE), correlation coefficient (R) and Theil's inequality coefficient (Theil U), and show promising results in both predictive accuracy and profitability performance.

19.1 Option Pricing

Options are fundamental financial instruments, dating back to the ancient time. A challenging problem is option pricing, especially for American type options, which can be exercised before the maturity date. For European options, which can only be exercised at the maturity date, prices can be calculated by the Black-Scholes formula in certain cases. The key to American option pricing (Longstaff and Schwartz, 2001; Tsitsiklis and Van Roy, 2001; Li et al., 2009) is to calculate the conditional expected value of continuation. This is an optimal stopping problem. Hull (2014) provides an introduction to options and other derivatives and their pricing methods; and Glasserman (2004) provides a book length treatment for Monte Carlo methods in financial engineering. The least squares Monte Carlo (LSM) method in Longstaff and Schwartz (2001), following approximate dynamic programming, is a standard approach in the finance literature for pricing American options.

19.2 Portfolio Optimization

Mean-variance analysis by Markowitz is a classical approach to portfolio optimization in one period (Luenberger, 1997). Dynamic portfolio optimization in multi-period renews its attraction recently (Campbell and Viceira, 2002; Brandt et al., 2005), following the recent empirical evidence of return predictability (Pastor and Stambaugh, 2009), and with the consideration of practical issues including parameter and model uncertainty, transaction cost and background risks (Brandt et al., 2005). Brandt et al. (2005) and Neuneier (1997) deploy the backward dynamic programming approach in Longstaff and Schwartz (2001) for the dynamic portfolio problem. It is possible to apply reinforcement learning methods for it.

Moody and Saffell (2001) learns to trade via direct reinforcement, without any forecasting. Deng et al. (2016) extend it with deep neural networks. It may be beneficial to take advantage of return predictability in RL methods.

It is critical to control the risk when forming portfolios. Value-at-Risk (VaR) is a popular risk measure; while conditional VaR (CVaR) has desirable mathematical properties (Hull, 2014). Yu et al. (2009) provide formulations for VaR and CVaR with relaxed probability distributions by worst-case analysis. Deep (reinforcement) learning would provide better solutions in some issues in risk management. The generalization to continuous state and action spaces is an indispensable step for such methods to be applied to dynamic portfolio optimization.

19.3 BUSINESS MANAGEMENT

Li et al. (2010) formulate personalized news articles recommendation as a contextual bandit problem, to learn an algorithm to select articles sequentially for users based on contextual information of users and articles, such as historical activities of users and descriptive information and categories of content, and to take user-click feedback to adapt selection policy to maximize total user clicks in the long run.

Theocharous et al. (2015) formulate a personalized ads recommendation systems as a RL problem to maximize life-time value (LTV) with theoretical guarantees. This is in contrast to a myopic solution with supervised learning or contextual bandit formulation, usually with the performance metric of click through rate (CTR). As the models are hard to learn, the authors deploy a model-free approach to compute a lower-bound on the expected return of a policy to address the off-policy evaluation problem, i.e., how to evaluate a RL policy without deployment.

Jiang and Li (2016) study off-policy value evaluation by extending the doubly robust estimator for bandits. The proposed method helps safety in policy improvements and applies to both shallow and deep RL. One experiment is about maximizing lifetime value of customers. Silver et al. (2013) propose concurrent reinforcement learning for the customer interaction problem. See Cai et al. (2018b) for mechanism design for fraudulent behaviour in e-commerce, Hu et al. (2018a) for ranking in e-commerce search engine, Hu et al. (2018c) for incentive mechanism design in crowdsourcing, Lattimore et al. (2018) for ranking, Nazari et al. (2018) for vehicle routing in operations research, Shi et al. (2018) for visualization of online retail environment for RL, and, Zhao et al. (2018b), Zhao et al. (2018c) and Zheng et al. (2018a) for recommendation. See Zhang et al. (2017) for a survey on recommendation.

See Section 16.7 Personalized Web Services in Sutton and Barto (2018) for a detailed and intuitive description of some topics discussed here.

20 MORE APPLICATIONS

In this chapter, we discuss more reinforcement learning applications: healthcare in Section 20.1, education in Section 20.2. energy in Section 20.3, transportation in Section 20.4, computer systems in Section 20.5, and, science, engineering and art in Section 20.6.

20.1 HEALTHCARE

There are many opportunities and challenges in healthcare for machine learning (Miotto et al., 2017; Saria, 2014). Personalized medicine is getting popular in healthcare. It systematically optimizes patients' health care, in particular, for chronic conditions and cancers using individual patient information, potentially from electronic health/medical record (EHR/EMR). Li et al. (2018b) propose a hybrid retrieval-generation reinforced agent for medical image report generation. Rajkomar et al. (2018) investigate applying deep learning to EHR data. Rotmensch et al. (2017) learn a healthcare knowledge graph from EHR. Fauw et al. (2018) apply deep learning for diagnosis and referral in retinal disease; see a blog at `https://deepmind.com/blog/moorfields-major-milestone/`. Gheiratmand et al. (2017) study network-based patterns of schizophrenia. Rajpurkar et al. (2018) introduce a large dataset of musculoskeletal radiographs. See a tutorial on deep reinforcement learning for medical imaging at `https://www.hvnguyen.com/deepreinforcementlearning`. See Liu and Sun (2017) for a tutorial on deep learning for health care applications. See a course on Machine Learning for Healthcare, at `https://mlhc17mit.github.io`.

Dynamic treatment regimes (DTRs) or adaptive treatment strategies are sequential decision making problems. Some issues in DTRs are not in standard RL. Shortreed et al. (2011) tackle the missing data problem, and design methods to quantify the evidence of the learned optimal policy. Goldberg and Kosorok (2012) propose methods for censored data (patients may drop out during the trial) and flexible number of stages. See Chakraborty and Murphy (2014) for a recent survey, and Kosorok and Moodie (2015) for an edited book about recent progress in DTRs. Currently Q-learning is the RL method in DTRs. Ling et al. (2017) apply deep RL to the problem of inferring patient phenotypes. Liu et al. (2018d) study off-policy policy evaluation and its application to sepsis treatment. Kallus and Zhou (2018) study confounding-robust policy improvement and its application to acute ischaemic stroke treatment. Peng et al. (2018c) study disease diagnosis.

Some recent conferences and workshops at the intersection of machine learning and healthcare are: Machine Learning for Healthcare, `https://www.mlforhc.org`; NIPS 2018 Workshop on Machine Learning for Health (ML4H): Moving beyond supervised learning in healthcare; NIPS 2017 Workshop on Machine Learning for Health (ML4H), `https://ml4health.github.io/2017/`; NIPS 2016 Workshop on Machine Learning for Health (ML4H), `http://www.nipsml4hc.ws`; NIPS 2015 Workshop on Machine Learning in Healthcare, `https://sites.google.com/site/nipsmlhc15/`. See an issue of Nature Biomedical Engineering on machine learning at `https://www.nature.com/natbiomedeng/volumes/2/issues/10`. See Hernandez and Greenwald (2018), a WSJ article about IBM Watson dilemma.

20.2 EDUCATION

Northcutt (2017) presents tutorial-style slides for AI in online education with an emphasis on personalization. The author presents a framework for AI in online education, including the active/passive course, content, and student, and give examples as below. Active AI refers to changes in course or experience; passive AI refers to unseen estimation or modeling.

- active student: cognitive tutor
- passive student: proficiency estimation (IRT)
- active content: content recommendation engine
- passive content: estimating points of confusion in instructional videos
- active course: auto-generate new courses from pieces of other courses, students interact; measure outcomes, and iterate
- passive course: estimate optimal course prerequisite structure for a new field

Northcutt (2017) present 18 problems, some with solutions, some with ideas.

One problem is representation, which is about how to represent courses as vectors, how to measure similarity of two courses, videos, and students, how to recommend content to students, and how to match students with other students, etc.

One approach to obtain representation is to use embeddings, on courses, content, or students. High-dimensional feature matrices can be used to capture the interactions between courses, content, and students. Dimension reduction can be done using PCA, SVD, etc. or hidden layers of a neural network. With such embedded dense low-dimensional representations, we can generate new courses from existing courses, pair student, make inference, and structure content, etc.

We can employ a recommendation engine for MOOCs using embeddings, with a Siamese neural network architecture, one network to represent students, and another network for content, and produce representations so that $cosine(student, content)$ measures the goodness of $content$ for $student$.

Northcutt (2017) also discuss the following problems, detect struggling, rampant cheating, stats collaboration, how to personalize, independent treatment effect (ITE)/counterfactuals, trajectory prediction, how to order content, adaptive learning, Google Scholar, majority bias in forums, feature extraction, cognitive state, content likability, points of confusion, cognitive modeling, human intelligence vs. artificial intelligence, and the next edX. See (Northcutt, 2017) for more details.

There are some recent work in education using RL.

Mandel et al. (2014) propose an offline policy evaluation method, by combining importance sampling with cross-validation, to investigate generalization of representations. The authors propose a feature compaction algorithm for high-dimension problems, which benefit from both PCA and neural networks. Furthermore, the authors apply the method to an educational game, optimizing engagement by learning concept selection.

Liu et al. (2014) propose UCB-Explore, based on multi-armed bandit algorithm UCB1, to automaticaly allocate experimental samples, to balance between learning the effectiveness of each experimental condition and users' test performances, by explicitly specifying the tradeoff between these two objectives. The authors compare UCB-Explore with other multi-armed bandit algorithms like UCB1 and ϵ-greedy with simulation on an educational game.

Upadhyay et al. (2018) apply reinforcement learning to marked temporal point processes with an application in personalized education. See also Li et al. (2018a) for applying RL to temporal point processes. Oudeyer et al. (2016) discuss theory and applications of intrinsic motivation, curiosity, and learning in educational technologies.

Watch a video titled Reinforcement Learning with People (Brunskill, 2017). See ACL 2018 Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2018) with a Shared Task for Chinese Grammatical Error Diagnosis (CGED), at `https://sites.google.com/view/nlptea2018/`.

20.3 ENERGY

A smart grid is a power grid utilizing modern information technologies to create an intelligent electricity delivery network for electricity generation, transmission, distribution, consumption, and control (Fang et al., 2012). An important aspect is adaptive control (Anderson et al., 2011). Glavic et al. (2017) review application of RL for electric power system decision and control. See Platt (2017) for a talk about energy. Here we briefly discuss demand response (Wen et al., 2015; Ruelens et al., 2016).

Demand response systems motivate users to dynamically adapt electrical demands in response to changes in grid signals, like electricity price, temperature, and weather, etc. With suitable electricity prices, load of peak consumption may be rescheduled/lessened, to improve efficiency, reduce costs, and reduce risks. Wen et al. (2015) propose to design a fully automated energy management system with model-free reinforcement learning, so that it doesn't need to specify a disutility function to model users' dissatisfaction with job rescheduling. The authors decompose the RL formulation over devices, so that the computational complexity grows linearly with the number of devices, and conduct simulations using Q-learning. Ruelens et al. (2016) tackle the demand response problem

with batch RL. Wen et al. (2015) take the exogenous prices as states, and Ruelens et al. (2016) utilize the average as feature extractor to construct states.

20.4 TRANSPORTATION

Intelligent transportation systems (Bazzan and Klügl, 2014) apply advanced information technologies for tackling issues in transport networks, like congestion, safety, efficiency, etc., to make transport networks, vehicles and users smart.

Autonomous driving vehicles is an important topic of intelligent transportation systems. We discuss it in Section 16.6.

See NIPS Workshops on Machine Learning for Intelligent Transportation Systems, in 2018 at `https://sites.google.com/site/nips2018mlits/`, in 2017 at `https://sites.google.com/site/nips2017mlits/`, and, in 2016 at `https://sites.google.com/site/nips2016intelligenttrans/`.

ADAPTIVE CONTROL

An important issue in intelligent transportation systems is adaptive control. El-Tantawy et al. (2013) propose to model the adaptive traffic signal control problem as a multiple player stochastic game, and solve it with the approach of multi-agent RL (Shoham et al., 2007; Busoniu et al., 2008). Multi-agent RL integrates single agent RL with game theory, facing challenges of stability, nonstationarity, and curse of dimensionality. El-Tantawy et al. (2013) approach the issue of coordination by considering agents at neighbouring intersections. The authors validate their proposed approach with simulations, and real traffic data from the City of Toronto. El-Tantawy et al. (2013) don't explore function approximation. See van der Pol and Oliehoek (2017) for a recent work, and Mannion et al. (2016) for an experimental review, about applying RL to adaptive traffic signal control. Belletti et al. (2018) study expert level control of ramp metering based on multi-task deep reinforcement learning.

20.5 COMPUTER SYSTEMS

Computer systems are indispensable in our daily life and work, e.g., mobile phones, computers, and cloud computing. Control and optimization problems abound in computer systems, e,g., Mestres et al. (2017) propose knowledge-defined networks, Gavrilovska et al. (2013) review learning and reasoning techniques in cognitive radio networks, and Haykin (2005) discuss issues in cognitive radio, like channel state prediction and resource allocation. We also note that Internet of Things (IoT)(Xu et al., 2014) and wireless sensor networks (Alsheikh et al., 2014) play important roles in robotics and autonomous driving as discussed in Chapter 16, in energy systems as discussed in Section 20.3, and in transportation as discussed in Section 20.4. See Zhang et al. (2018) for a recent survey about applying deep learning and reinforcement learning to issues in mobile and wireless networking. Mukwevho and Celik (2018) discuss fault tolerance in cloud computing.

Kraska et al. (2018) propose learned indexes, by treating B-Tree, Hash, BitMap, etc. as models, and use neural networks to learn such models, by using the signal of learned model of structure or sort order of lookup keys to predict the existence or position of records. Experiments show promising results. Wang and O'Boyle (2018) study compiler optimization. Reichstaller and Knapp (2017) study software testing. Krishnan et al. (2018) study SQL join queries optimization. Faust et al. (2018) study sorting. See recent papers about neural approaches for program synthesis, e.g., Balog et al. (2017); Liang et al. (2017a; 2018); Nachum et al. (2017); Parisotto et al. (2017); Reed and de Freitas (2016); Vinyals et al. (2015); Zaremba and Sutskever (2015); Zhang et al. (2018).

See SysML conference, at the intersection of system and machine learning, at `https://www.sysml.cc`. See NIPS 2018 Workshop on Security in Machine Learning.

PERFORMANCE OPTIMIZATION

Mirhoseini et al. (2017) propose to optimize device placement for Tensorflow computational graphs with RL. The authors deploy a seuqence-to-sequence model to predict how to place subsets of operations in a Tensorflow graph on available devices, using the execution time of the predicted

placement as reward signal for REINFORCE algorithm. The proposed method finds placements of Tensorflow operations on devices for Inception-V3, recurrent neural language model and neural machine translation, yielding shorter execution time than those placements designed by human experts. Computation burden is one concern for a RL approach to search directly in the solution space of a combinatorial problem. We discuss learning combinatorial optimization in Section 14.5. Gao et al. (2018c) also study the problem of device placement.

Mao et al. (2016) study resource management in systems and networking with deep RL. The authors propose to tackle multi-resource cluster scheduling with policy gradient, in an online manner with dynamic job arrivals, optimizing various objectives like average job slowdown or completion time. The authors validate their proposed approach with simulation.

Liu et al. (2017a) propose a hierarchical framework to tackle resource allocation and power management in cloud computing with deep RL. The authors decompose the problem as a global tier for virtual machines resource allocation and a local tier for servers power management. The authors validate their proposed approach with actual Google cluster traces. Such hierarchical framework/decomposition approach was to reduce state/action space, and to enable distributed operation of power management.

Google deploy machine learning for data centre power management, reducing energy consumption by 40%. See blogs at `http://goo.gl/4PHcos` and `http://goo.gl/N3Aoxm`. Lazic et al. (2018) study data center cooling with model-predictive control (MPC).

Optimizing memory control is discussed in Sutton and Barto (2018).

SECURITY

There is a long history applying machine learning (ML) techniques to system security issues, e.g., Chandola et al. (2009) survey ML techniques for anomaly detection, and, Sommer and Paxson (2010) discuss issues in using ML techniques for network intrusion detection.

Adversarial machine learning is about learning in the presence of adversaries. It is concerned with the training stage, when facing data poisoning issues, and learning wrong models hard to detect. It is also concerned with the inference stage, when facing adversarial examples, and making wrong decisions. Adversarial ML is a critical for some ML applications, like autonomous driving.

Adversarial ML is an emerging field, in this wave of deep learning, after researchers find adversarial examples to deep learning algorithms, e.g., Szegedy et al. (2013) show that various images, like a truck, a building, or a dog, after being added small noises, are all classified by AlexNet as "ostrich, Struthio camelus". Goodfellow et al. (2015) also show a fast adversarial example generation method, so that an image of panda, after being added a small vector, is classified as a gibbon by GoogLeNet. Eykholt et al. (2018) show that physical images, like stop signs, yield signs, left turn signs etc., after being perturbed by adding black or white stickers, are misclassified by the state of art deep neural networks as speed limit 45 signs. Evtimov et al. (2017) discuss attacks to physical images, `http://bair.berkeley.edu/blog/2017/12/30/yolo-attack/`. RL algorithms are also vulnerable to adversarial attacks, e.g., Huang et al. (2017) and Havens et al. (2018), as well as multi-armed bandit algorithms, e.g., Jun et al. (2018). See a blog at `http://rll.berkeley.edu/adversarial/`.

Adversarial ML is an active area, with fierce competition between the design of attack and defense algorithms. Athalye et al. (2018) show that seven of nine defense techniques, shortly after their papers being accepted to ICLR 2018, cause the issue of obfuscated gradients and are vulnerable to their attacks. See their open source at `https://github.com/anishathalye/obfuscated-gradients` for implementations of their attack and the studied defense methods.

Anderson et al. (2018) propose a black-box attack approach against static portable executable (PE) anti-malware engines with reinforcement learning, which produces functional evasive malware samples to help improve anti-malware engines. The performance still needs improvements. See the open source at `https://github.com/endgameinc/gym-malware`.

See Song (2018) for a tutorial on AI and security. See Kantarcioglu and Xi (2016) for a tutorial on adversarial data mining. See Yuan et al. (2017) for a survey on attacks and defenses for deep

learning. Papernot et al. (2016) present CleverHans, a software library for reference implementations adversarial ML algorithms.

20.6 SCIENCE, ENGINEERING AND ART

Reinforcement learning, deep learning, machine learning, and AI in general, have very wide interactions with science, engineering and art. We see that RL and areas in science, engineering and art influence each other, i.e., RL/AI has applications in these areas, with new observations or even new algorithms and new principles; and intuitions and principles from these areas help further development of RL/AI. For example, Sutton and Barto (2018) discuss the interplay between RL and neuroscience and psychology; and in Chapter 14, we discuss learning to learn new algorithms. Here we focus on applications of RL/AI to these areas.

Sutton and Barto (2018) treat dynamic programming (DP) and Markov decision processes (MDPs) as foundations for RL, and also devote two chapters for neuroscience and psychology, respectively. There are books discussing approximate dynamic programming, MDPs, operations research, optimal control, as well as the underlying optimization, statistics, and probability, e.g.,Bertsekas and Tsitsiklis (1996), Bertsekas (2012), Szepesvári (2010), and Powell (2011). There are strong relationships between these areas with RL. [8] Powell (2010) discusses merging AI and operations research to solve high-dimensional stochastic optimization problems with approximate dynamic programming. Lake et al. (2016) discuss incorporating human intelligence into the current DL/RL/AI systems. Hassabis et al. (2017) discuss the connection between neuroscience and RL/AI. Kriegeskorte and Douglas (2018) surveys cognitive computational neuroscience.

RL/AI is relevant to many areas, e.g., mathematics, chemistry, physics (Cranmer, 2016), biology (Mahmud et al., 2018), music, drawing (Xie et al., 2012; Ha and Eck, 2018), character animation (Peng et al., 2018a;b), dancing (Chan et al., 2018), storytelling (Thue et al., 2007), etc. DeVries et al. (2018) study earthquakes with deep learning. Some topics, e.g., music, drawing, storytelling, are at the intersection of science and art.

We discuss games in Chapter 15, robotics in Chapter 16, computer vision in Chapter 18, natural language processing (NLP) in Chapter 17, and, computer systems in Section 20.5, as areas in computer science. We put many topics in computer science like indexing (Kraska et al., 2018), compiler optimization (Wang and O'Boyle, 2018), software testing (Reichstaller and Knapp, 2017), SQL join queries optimization (Krishnan et al., 2018), and sorting (Faust et al., 2018) etc. in computer systems in Section 20.5. See recent papers about neural approaches for program synthesis, e.g., Balog et al. (2017); Liang et al. (2017a; 2018); Nachum et al. (2017); Parisotto et al. (2017); Reed and de Freitas (2016); Vinyals et al. (2015); Zaremba and Sutskever (2015); Zhang et al. (2018).

We discuss finance and business management in Section 19, healthcare in Section 20.1, and, education in Section 20.2, as areas in social science. We discuss energy in Section 20.3, and transportation in Section 20.4, as areas in engineering. [9]

Imagination is critical for creative activities, like science, engineering and art. Mahadevan (2018b) discuss imagination machines as a new challenge for AI. See Mahadevan (2018a) for a tutorial on this topic.

Quantum machine learning is about designing machine learning algorithms on quantum computing architectures, at the interaction of theoretical computer science, machine learning, and physics. Biamonte et al. (2017) survey quantum machine learning, including quantum reinforcement learning.

We list some workshops in the following.

- NIPS 2015 Workshop on Quantum Machine Learning at `https://www.microsoft.com/en-us/research/event/quantum-machine-learning/`
- Machine Learning for Science Workshop at LBNL at `https://sites.google.com/lbl.gov/ml4sci/`

[8] Check for a special issue of IEEE Transactions on Neural Networks and Learning Systems on Deep Reinforcement Learning and Adaptive Dynamic Programming, published in June 2018, `https://ieeexplore.ieee.org/document/8353782/`.

[9] Computer vision and computer systems are also in engineering.

- Machine Learning for Physics and the Physics of Learning at
  `http://www.ipam.ucla.edu/programs/long-programs/machine-learning-for-physics-and-the-physics-of-learning/`
- NIPS 2018 Workshop Machine Learning for Molecules and Materials at `http://www.quantum-machine.org/workshops/nips2018draft/`
- NIPS Workshop on Machine Learning for Creativity and Design
  - in 2018 at `https://nips2018creativity.github.io/`
  - in 2017 at `https://nips2017creativity.github.io`

In this section, we attempt to put reinforcement learning in the wide context of science, engineering, and art. We have already touched many aspects in previous chapters/sections. Here we only discuss a small sample of the aspects we have not discussed before.

20.6.1 CHEMISTRY

Retrosynthesis is a chemistry technique to transform a target molecule into simpler precursors recursively. Segler et al. (2018) propose to combine Monte Carlo tree search (MCTS) with symbolic rules for automatic retrosynthesis. Three deep neural networks, namely, an expansion policy network to guide the search with transformations extracted automatically, a filter network to feasibility of the proposed reactions, and a rollout policy network to sample transformations to estimate the value of a position, are trained with almost all reactions published in organic chemistry. The proposed approach improves previous computer-aided synthesis planning systems significantly. Segler et al. (2018) follow the approach of AlphaGo in Silver et al. (2016a). It is interesting to study if the approach of AlphaGo Zero (Silver et al., 2017), in particular, generalized policy iteration, self-play, and a single neural network, can be applied to retrosynthesis.

Popova et al. (2018) apply deep RL for computational de novo drug design, discovering molecules with desired properties. Jaques et al. (2017) as discussed below for music melody generation also study computational molecular generation.

20.6.2 MATHEMATICS

Deep learning has many applications in maths, e.g., neural ordinary differential equations (ODEs) (Chen et al., 2018), proofs (Irving et al., 2016; Loos et al., 2017; Rocktäschel and Riedel, 2017; Urban et al., 2018). In the following, we discuss a case using deep RL for partial differential equations (PDEs).

PDEs are mathematical tools for wide applications in science and engineering. It is desirable to design a PDE controller with minimal assumptions, without knowledge of the PDE, and being data-driven. Pan et al. (2018) study how to control dynamical systems described by PDEs using RL methods, with high-dimensional continuous action spaces, having spatial relationship among action dimensions. The authors propose action descriptors to encode such spatial regularities and to control such PDEs. The authors show sample efficiency of action descriptors theoretically, comparing with conventional RL methods not considering such regularities. The authors implement action descriptors with Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2016), and experiment with two high-dimensional PDE control problems.

20.6.3 MUSIC

Jaques et al. (2017) propose Sequence Tutor, combining maximum likelihood estimation (MLE) with RL, to consider both data and task-related goals, to improve the structure and quality of generated sequences, and to maintain sample diversity and information learned from data, by pre-training a Reward RNN using MLE, and training another RNN with off-policy RL, to generate sequences with high quality, considering domain-specific rewards, and penalizing divergence from the prior policy learned by Reward RNN. The authors investigate the connection between KL control and sequence generation, and relationship among G-learning (Fox et al., 2016), Ψ-learning (Rawlik et al., 2012), and Q-learning. The authors evaluate Sequence Tutor on musical melody generation. It is nontrivial to design a reward function to capture the aesthetic beauty of generated melodies, and a pure data-driven approach can not yield melodies with good structure. Sequence Tutor incorporates

rules from music theory into the model generating melodies, to produce more pleasant-sounding and subjectively pleasing melodies than alternative methods. The authors also conduct experiments with Sequence Tutor for computational molecular generation.

van den Oord et al. (2016) propose WaveNet for raw audio waveforms generation. See Briot et al. (2018) about deep learning techniques for music generation. See Zhu et al. (2018) for pop music generation. See also Dieleman et al. (2018). See the Magenta project, `https://magenta.tensorflow.org`, for investigation of deep learning and reinforcement learning for art and music creation. See the 2018 ICML, IJCAI/ECAI, and AAMAS Joint Workshop on Machine Learning for Music, `https://sites.google.com/site/faimmusic2018/`.

21 DISCUSSIONS

We present deep reinforcement learning in this manuscript in an overview style. We discuss the following questions: Why deep? What is state of the art? What are the issues, and potential solutions? Briefly, the powerful and flexible representation by deep learning helps deep RL make many achievements. We discuss state of the art, issues and potential solutions for deep RL in chapters on six core elements, six important mechanisms, and twelve applications. In the following, we present a brief summary, discuss challenges and opportunities, and close with an epilogue.

21.1 BRIEF SUMMARY

There are many concepts, algorithms, and issues in (deep) reinforcement learning (RL), as illustrated in Figure 5. We touch many of them in this manuscript.

Credit assignment, sparse reward, and sample efficiency are common issues for RL problems. We list several approaches proposed to address them in the following. In off-policy learning, both on-policy and off-policy data can be used for learning. Auxiliary reward and self-supervised learning are for learning from non-reward signals in the environment. Reward shaping is for providing denser rewards. Hierarchical RL is for temporal abstraction. General value function, in particular, Horde, universal function approximator, and hindsight experience replay, is for learning shared representation/knowledge among goals. Exploration techniques are for learning more from valuable actions. Model-based RL can generate more data to learn from. Learning to learn, e.g., one/zero/few-shot learning, transfer learning, and multi-task learning, learns from related tasks to achieve efficient learning. Incorporating inductive bias, structure, and knowledge can help achieve more intelligent representation and problem formulation. And so on and so forth.

Finn (2017) and Levine (2018) discuss that sample efficiency improves roughly by ten times in each step from one RL method to another in the following, from gradient-free methods, like CMA-ES, fully online methods like A3C, policy gradient methods, like TRPO, value estimation methods with reply buffer, like Q-learning, DQN, DDPG, and NAF, model-based deep RL methods, like guided policy search (GPS), all the way to model-based "shallow" RL methods, like PILCO.

Silver (2018) summarizes principles of deep RL: evaluation drives progress, scalability determines success, generality future-proofs algorithms, trust in the agent?s experience, state is subjective, control the stream, value functions model the world, planning: learn from imagined experience, empower the function approximator, and, learn to learn.

Some issues deserve more discussions, in particular, safety and interpretability. We discuss several aspects of AI safety, e.g., reward in Chapter 5, multi-agent RL in Chapter 12, and, adversarial examples in Section 20.5. There are recent work on RL safety, e.g., Chow et al. (2018), Huang et al. (2018), and Wen and Topcu (2018). See surveys for AI safety (Amodei et al., 2016; Garcìa and Fernàndez, 2015). See a course on safety and control for artificial general intelligence, at `http://inst.eecs.berkeley.edu/~cs294-149/`. See blogs about safety, e.g., at `https://medium.com/@deepmindsafetyresearch`. Doshi-Velez and Kim (2017), Lage et al. (2018), Lipton (2018), and Melis and Jaakkola (2018) discuss issues of interpretability. Zhang and Zhu (2018) survey visual interpretability for deep learning.

21.2 CHALLENGES AND OPPORTUNITIES

In the following, we discuss issues in deep RL, and propose research directions as both challenges and opportunities, which are challenging, or even widely open.

Rahimi and Recht (2017b) raise the concern that "machine learning has become alchemy". This alludes in particular to deep learning. See their blogs, Rahimi and Recht (2017c), Rahimi and Recht (2017a). In Sculley et al. (2018), Ali Rahimi and colleagues discuss productive changes for empirical rigor, and recommend standards for empirical evaluation: tuning methodology, sliced analysis, ablation studies, sanity checks and counterfactuals, and at least one negative result.

Lipton and Steinhardt (2018) discuss troubling trends in machine learning, including failure to distinguish between explanation and speculation, failure to identify the sources of empirical gains, mathiness, which obfuscates or impresses but not clarifies with mathematics, and, misuse of lan-

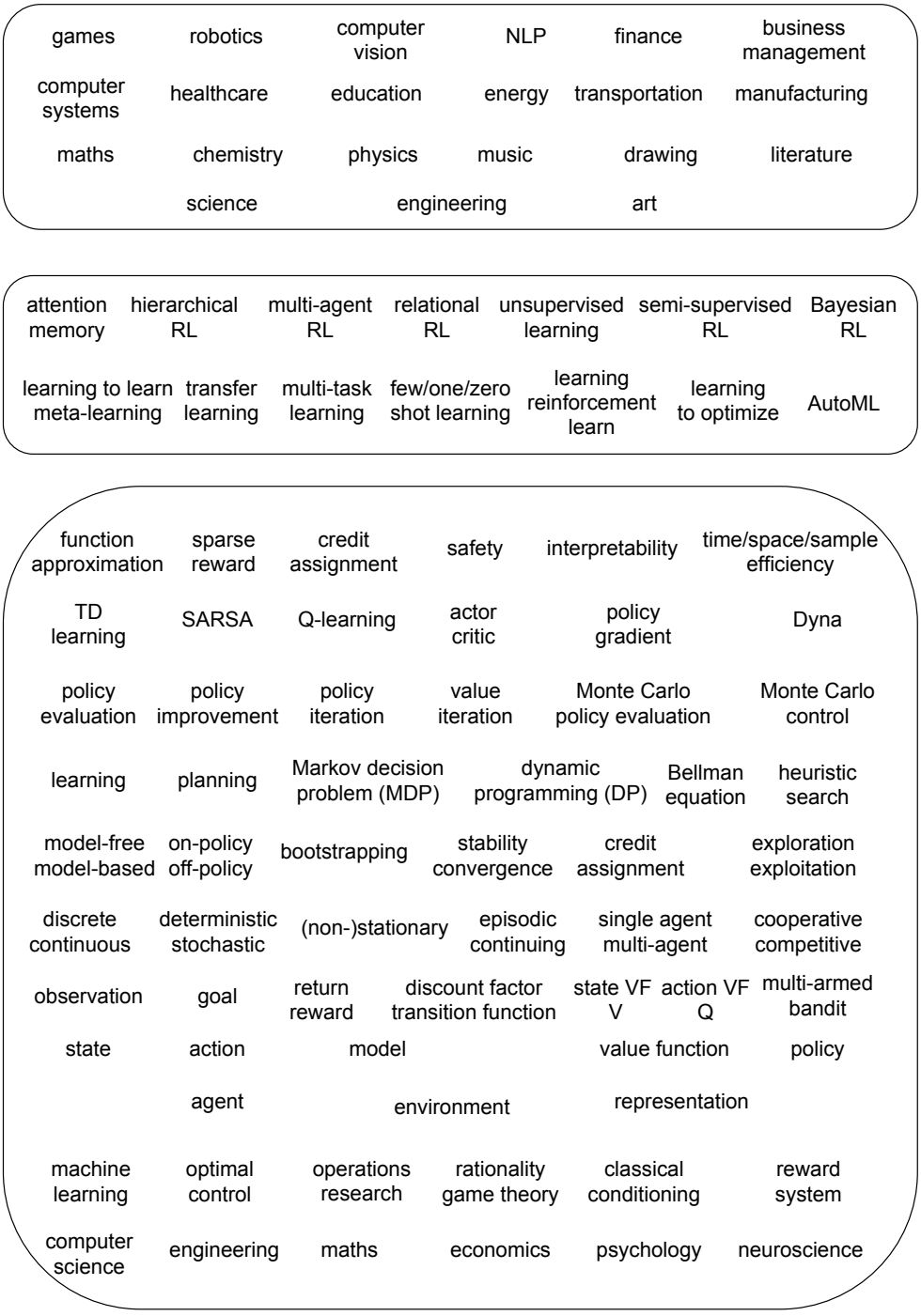

Figure 5: Concepts, Algorithms, and Issues in Reinforcement Learning

guage, with suggestive definitions, overloading technical terminology, or suitcase words for a variety of meanings. See NIPS 2018 Workshop on Critiquing and Correcting Trends in Machine Learning.

Henderson et al. (2018) investigate reproducibility, experimental techniques, and reporting procedures for deep RL. The authors show that experimental results are influenced by hyperparameters, including network architecture and reward scale, random seeds and trials, environments (like Hopper or HalfCheetah etc. in OpenAI Baseline), and codebases. This causes difficulties for reproducing deep RL research results. The authors analyze the following reporting evaluation metrics: online view vs. policy optimization (offline), confidence bounds (sample bootstrap), power analysis (about sample size), significance (like t-test). Henderson et al. (2018) recommend to report implementation details, all hyperparameter settings, experimental setup, and evaluation methods for reproducibility. The authors also recommend to find the working set of hyperparameters, match baseline algorithm implementation with the original codebase, run many trails with different random seeds then average results, and perform proper significance tests to validate better performance.

Khetarpal et al. (2018) discuss evaluation differences in RL and in supervised learning, and propose an evaluation pipeline.

Levine (2018) discusses challenges with deep RL, including stability, efficiency, scalability, hyperparameters tuning, sample complexity, model-based learning, generalization, reward specification, prior knowledge, etc.

These papers [10] discuss various issues with deep learning, machine learning, deep RL, and provide valuable insights. There are also benchmark papers like Duan et al. (2016). However, we still lack papers conducting systematic, comparative study of deep RL algorithms, so that we pick one or more benchmark problems, do a thorough study, report both successes and failures, summarize advices and lessons, and, give guidelines about how to use deep RL algorithms. Our deep RL community need such papers. As well, most RL + NLP/computer vision papers use REINFORCE. A natural question is: how about other (deep) RL algorithms? We can evaluate performance of many algorithms, like DQN, A3C, DDPG, TRPO, PPO, PCL, Trust-PCL, Retrace, Reactor, interpolated policy gradient, soft Q-learning, etc. As such, we propose the following research direction.

Research Direction 1: systematic, comparative study of deep RL algorithms

Bellemare et al. (2018) open source Dopamine, aiming for a flexible, stable, and reproducible Tensorflow-based RL framework, as an achievement in this direction.

We have seen exciting results in two-player and multi-agent games recently. AlphaGo (Silver et al., 2016a; 2017) has achieved super-human performance. DeepStack (Moravčík et al., 2017) defeated professional poker players. Jaderberg et al. (2018) achieve human level performance in the game of Capture the Flag (Chapter 12). OpenAI Five has beaten human players on 5v5 Dota 2, although with huge computation (`https://blog.openai.com/openai-five/`). Zambaldi et al. (2018) achieve decent results on StarCraft II mini-games (Chapter 13). Sun et al. (2018) and Pang et al. (2018) have beaten full-game built-in AI in StarCraft II.

However, multi-agent problems are still very challenging, with issues like non-stationarity and even theoretical infeasibility, as we discuss in Chapter 12. Even so, we can endeavour to achieve decent results for multi-agent problems, like approximation solutions with high quality, and/or super-human performance. Multi-agent systems are a great tool to model interactions among agents, with rich applications in human society; and their advancements can significantly push the frontier of AI. We thus propose the second research direction as below.

Research Direction 2: "solve" multi-agent problems

[10] There is a blog titled Deep Reinforcement Learning Doesn't Work Yet at `https://www.alexirpan.com/2018/02/14/rl-hard.html`. It summarizes issues with deep RL, including sample inefficiency, better results with non-RL methods, issues with reward function, local optimal hard to escape, overfitting, and, hard to reproduce due to instability. The blog contains informative discussions; however, the title is wrong. There is another blog titled Lessons Learned Reproducing a Deep Reinforcement Learning Paper at `http://amid.fish/reproducing-deep-rl`.

StarCraft and Texas Hold'em Poker are great testbeds for studying multi-agent problems. It is desirable to see extensions of DeepStack to multi-player settings, with many hands of playing, in tournament and cash game styles.

StarCraft features many possible actions, complex interactions between players, short term tactics and long term strategies, etc. Learning strategies for Starcraft following videos with commentary would be a feasible strategy. There are many videos about StarCraft with excellent commentaries. If we may be able to extract valuable information, like strategies, from the multi-modality signals, and apply these to the agent design, we may be able to achieve a human level AI StarCraft agent. Such a system would be an integration of RL, computer vision, and NLP. Aytar et al. (2018) achieve breakthrough results on three hard Atari games with self-supervision techniques by watching YouTube (Chapter 10). This may give us more motivation and encouragements. As a related work, Branavan et al. (2012) propose to learn strategy games by reading manuals. With achievements in Sun et al. (2018) and Pang et al. (2018), it is interesting to watch if hierarchical RL approaches in these papers can achieve super-human performance.

We now discuss end-to-end learning with raw inputs, a trendy paradigm recently, e.g., AlexNet (Krizhevsky et al., 2012) with raw pixels for image classification, Seq2Seq (Sutskever et al., 2014) with raw sentences for machine translation, DQN (Mnih et al., 2015) with raw pixels and score to play Atari games, AlphaGo Zero (Silver et al., 2017) with piece information and score to play computer Go, and, Jaderberg et al. (2018) with raw pixels and score to play Quake III Arena Capture the Flag.

One question is, is such paradigm of end-to-end learning with raw input good? Sample efficiency is usually an issue. For example, as shown in Hessel et al. (2018), Rainbow needs 44 million frames to exceed the performance of distributional DQN (Bellemare et al., 2017), which needs much less data than DQN (Mnih et al., 2015). Such huge amount of data require huge computation.

Another issue is adversarial examples, which may be more severe for critical applications. Szegedy et al. (2013) show that various images, like a truck, a building, or a dog, after being added imperceptible noises, are all classified by AlexNet as "ostrich, Struthio camelus". Eykholt et al. (2018) show that physical images, e.g., stop signs, left turn signs etc., after being perturbed by adding black or white stickers, are misclassified by state of the art deep neural networks as speed limit 45 signs.

Some papers propose to learn fully autonomously. AlphaGo Zero (Silver et al., 2017) and Jaderberg et al. (2018) have achieved such a goal to some extend. However, we have to admit that both computer Go and Quake III Arena Capture the Flag have perfect simulation models, and unlimited data can be generated relatively easily. Many practical applications, like robotics, healthcare, and, education, do not have such luxury. We may or may not ultimately achieve such a goal of fully autonomous learning in a general sense; and it is not clear for problems with practical concerns. Consider education. Most of us follow some curricula designed by experts, and learn from experts, rather than learning tabula rasa. Even we will achieve such a goal, as in most scientific discoveries, we may encounter spiral development, rather than going straightforwardly to the goal. We probably need some hybrid solution in the interim.

We expect that manual engineering reconciles with end-to-end learning, and symbolism reconciles with connectionism. We thus propose to add an "intelligence" component in the end-to-end processing pipeline, rather than treating the system as an entire blackbox, as most current deep neural networks do, as shown in Figure 6.

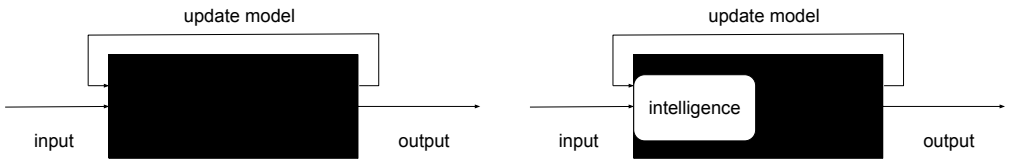

Figure 6: Add An Intelligence Component to End-to-End Pipeline

In the intelligence component, we may incorporate common knowledge like common sense, inductive bias, knowledge base, etc., common principles like Newton's laws, Bellman equation, etc., and, common algorithms like gradient descent, TD learning, policy gradient, etc.

The idea of adding an intelligent component is aligned with incorporating human intelligence as discussed in Lake et al. (2016). For example, when we study how to play billiard with a computer program, we probably want to incorporate Newton's law, rather than using deep learning to rediscover such laws with many video clips. Graves et al. (2013) follows end-to-end training for speech recognition, with a Fourier transformation of audio data. Self-supervised learning would be a promising approach for adding this intelligence component, e.g., Jaderberg et al. (2017) and Aytar et al. (2018).

Adding an intelligence component is abstract. Now we discuss something more concrete, esp. for tasks with perception, like with visual inputs. We then propose the following research direction.

Research Direction 3: learn from entities, but not just raw inputs

Our goal is to make the learning system more efficient w.r.t. sample, time, and space, to achieve interpretability and to avoid obvious mistakes like those in adversarial examples. At the same time, we still strive for end-to-end processing, and being fully differentiable. Our thesis is that, if we could process the raw input with some principle or knowledge, the resulting representation would be more convenient for the learning system to make further predictions or decisions.

Take the hard Atari game Montezuma's Revenge as an example. Suppose there were an intelligent system, which could identify entities in video frames, like agent, road, ladder, enemy, key, door, etc., and their attributes and relationships. Then a RL agent working on such representation would be much more efficient than working on pixels. A question is, if RL, unsupervised learning, or some machine learning/AI techniques can help identify entities, attributes, and their relationships. Successes in this direction would hinge on the maturity of computer vision, NLP, and AI.

There are recent progress in this direction. Goel et al. (2018) conduct unsupervised video object segmentation for deep RL. Eslami et al. (2018) present GQN for discovering representations with unsupervised learning. Chen et al. (2018a) propose a framework for iterative visual reasoning. For NLP, word2vec (Mikolov et al., 2013; Mikolov et al., 2017) is probably the most popular representation. van den Oord et al. (2018) propose to learn representations for multi-modality, including speech, images, text, and reinforcement learning. There are some recent papers about reasoning, e.g. (Santoro et al., 2017; Hudson and Manning, 2018; Battaglia et al., 2018), as we discuss in Chapter 13. We also discuss knowledge and reasoning in Section 8.3.

Malik (2018) discusses that there are great achievements in the fields of vision, motor control, and language semantic reasoning, and it is time to investigate them together. This supports our proposal.

Another fundamental, and related issue is about representation for RL problems. In deep learning, as well as in deep RL, neural network architecture is critical for the performance.

There are classical ways for function approximation, several popular neural network architectures, mechanisms for temporal abstraction, neural network architectures designed for deep RL and/or for reasoning, and discussions about causality and human intelligence. We discuss such representation issues in Chapter 8.

When we talk about computer vision with deep learning, CNNs appear in many people's minds. When we talk about RL algorithms, many people think about TD learning, Q-learning, and policy gradient. However, when we talk about representation or neural network architecture for (deep) RL, different people may come up with different ideas. It would be great to discover something for RL like CNNs for computer vision. We thus propose the following research direction.

Research Direction 4: design an optimal representation for RL

RL problems have their own structures and characteristics, e.g., value functions satisfy Bellman equation, so that they are probably different from those in deep learning, like image recognition and machine translation. Consequently, RL problems probably have their own optimal representation and neural network architecture. We conjecture that it is desirable to consider a holistic approach, i.e., considering perception and control together, rather than separately. Srinivas et al. (2018) and Tamar et al. (2018) are efforts in this direction.

To go one step further, we propose the next research direction to automate reinforcement learning, namely, AutoRL. A successful AutoRL tool would help us choose optimal state representation, function approximator, learning algorithms, hyperparameters, etc. There are efforts currently for machine learning tasks, namely, AutoML, as we discuss in Section 14.6.

Research Direction 5: AutoRL

We now talk about the last research direction. Reinforcement learning is very powerful and very important. Quoted from the authoritative AI textbook (Russell and Norvig, 2009), "reinforcement Learning might be considered to encompass all of AI: an agent is placed in an environment and must learn to behave successfully therein", and, "reinforcement learning can be viewed as a microcosm for the entire AI problem". Moreover, David Silver proposes a conjecture: artificial intelligence = reinforcement learning + deep learning (Silver, 2016). We attempt to justify this conjecture as follows. Hornik et al. (1989) establish that multilayer feedforward networks are universal approximators; that is, a feedback neural network with a single layer is sufficient to approximate any continuous function to an arbitrary precision. Hutter (2005) proves that tasks with computable descriptions in computer science can be formulated as RL problems. With deep learning providing mechanisms, and reinforcement learning defining the objective and achieving it, their integration can solve computable tasks, the aim of AI. Note the number of units in the hidden layer may be infeasibly large though, and, computability may be an issue, unless P = NP.

However, we see that deep learning is used much more widely, in supervised learning, unsupervised learning, and reinforcement learning. Furthermore, deep learning is also widely used in many applications, and is the core technique for many commercial products, like those with face recognition, speech recognition, etc. We enjoy the successes of deep RL, like those in Atari games and computer Go, which probably have limited commercial value. We see successes of a special RL family of techniques, namely, multi-armed bandits, for applications like news recommendation (Li et al., 2010). We also see achievements like those for neural architecture design (Zoph and Le, 2017). However, reinforcement learning still needs more efforts to become more practical, and we are still lacking of wide and practical applications of reinforcement learning that generate considerable commercial value. We thus propose the following research direction.

Research Direction 6: develop killer applications for (deep) RL

Successes of this research direction require the maturity of RL algorithms, for efficiency, stability, and robustness, etc. We see a positive feedback loop between algorithms and applications; they will help each other to make further improvements.

We now discuss a concrete recommendation for this direction: it is promising to invest more on applying AlphaGo techniques. AlphaGo techniques, in particular, deep learning, reinforcement learning, MCTS, and self play, are successful techniques, and have many potential applications. In particular, the elegant algorithm of AlphaGo Zero (Silver et al., 2017) would be straightforwardly applicable to a big family of problems.

We list potential applications of AlphaGo as suggested by the authors in their papers (Silver et al., 2016a; 2017), namely, general game-playing (in particular, video games) classical planning, partially observed planning, scheduling, constraint satisfaction, robotics, industrial control, online recommendation systems, protein folding, reducing energy consumption, and searching for revolutionary new materials. Although AlphaGo techniques have limitations, like requiring a good or even perfect simulator, we expect to see more and more application of AlphaGo techniques.

We list six research directions, as both challenges and opportunities of deep RL. Research direction 1, systematic, comparative study of deep RL algorithms, is about reproducibility, and under the surface, about stability and convergence properties of deep RL algorithms. Research direction 2, "solve" multi-agent problems, is usually about sample efficiency, sparse reward, stability, non-stationarity, and convergence in a large-scale, complex setting, a frontier in AI research. Research direction 3, learn from entities, but not just raw inputs, is about sample efficiency, sparse reward, and interpretability, by incorporating more knowledge, structure, and inductive bias. Research direction 4, design an optimal representation for RL, research direction 5, AutoRL, and, research

direction 6, develop killer applications for (deep) RL, are about the whole RL problem, about all issues in RL, like credit assignment, sparse reward, time/space/sample efficiency, accuracy, stability, convergence, interpretability, safety, scalability, robustness, simplicity, etc, from different angles of representation, automation, and application, respectively. We expect all these research directions to be open, except the first one, which is also challenging, and progress in these directions would deepen our understanding of (deep) RL, and push further frontiers of AI.

21.3 EPILOGUE

It is both the best and the worst of times for the field of deep RL, for the same reason: it has been growing so fast and so enormously. We have been witnessing breakthroughs, exciting new algorithms, architectures, and applications, and we expect to see much more and much faster. As a consequence, this manuscript is incomplete, in the sense of both depth and width. However, we attempt to summarize important achievements and discuss potential directions and applications in this amazing field.

Value function is central to reinforcement learning, e.g., in Deep Q-Network and its many extensions. Policy optimization approaches have been gaining traction, with many new algorithms, and in many, diverse applications, e.g., robotics, neural architecture design, spoken dialogue systems, machine translation, attention, and learning to learn, etc. This list is boundless. New learning mechanisms have emerged, e.g., using learning to learn, unsupervised learning, self-supervised learning, etc., to improve the quality and speed of learning, and more new mechanisms will be emerging. This is the renaissance of reinforcement learning (Krakovsky, 2016). In fact, deep learning and reinforcement learning have been making steady progress even during the last AI winter.

We have seen breakthroughs about deep RL, including DQN (Mnih et al., 2015), AlphaGo (Silver et al., 2016a; 2017), and DeepStack (Moravčík et al., 2017).

Exciting achievements abound: differentiable neural computer (Graves et al., 2016), unsupervised reinforcement and auxiliary learning (Jaderberg et al., 2017), asynchronous methods (Mnih et al., 2016), guided policy search (Levine et al., 2016), generative adversarial imitation learning (Ho and Ermon, 2016), and neural architecture design (Zoph and Le, 2017), etc. There are also many recent, novel applications of (deep) RL in many, diverse areas as discussed in previous chapters. Creativity would push the frontiers of deep RL further with respect to core elements, important mechanisms, and applications. In general, RL is probably helpful, if a problem can be regarded as or transformed to a sequential decision making problem, and states, actions, maybe rewards, can be constructed. Roughly speaking, if a task involves some manual designed "strategy", then there is a chance for reinforcement learning to help to automate and optimize the strategy.

Having a better understanding of how deep learning works is helpful for deep learning, machine learning, and AI. Poggio et al. (2017) review why and when deep- but not shallow-networks can avoid the curse of dimensionality. See Stanford STATS 385 course on Theories of Deep Learning at `https://stats385.github.io`. See Arora (2018) about theoretical understanding of deep learning. There are also papers for interpretability of deep learning, e.g. Doshi-Velez and Kim (2017), Lipton (2018), and Zhang and Zhu (2018).

It is important to investigate comments/criticisms for further progress. A popular criticism about deep learning is that it is a blackbox, or even an "alchemy" during the NIPS 2017 Test of Time Award speech (Rahimi and Recht, 2007). Lake et al. (2016) discuss incorporating machine intelligence with human intelligence for stronger AI; one commentary, Botvinick et al. (2017), discusses the importance of autonomy. Jordan (2018) discusses issues with AI. Darwiche (2018) discusses deep learning in the context of AI. See Peter Norvig's perspective (Press, 2016). Marcus (2018) criticizes deep learning, and Dietterich (2018) responds. Watch two debates, LeCun and Marcus (2017), LeCun and Manning (2018). See Stoica et al. (2017) for systems challenges for AI.

It is worthwhile to envision deep RL considering perspectives from the society, academia and industry on AI, e.g., Artificial Intelligence, Automation, and the Economy, Executive Office of the President, USA; Artificial Intelligence and Life in 2030 - One Hundred Year Study on Artificial Intelligence: Report of the 2015-2016 Study Panel, Stanford University (Stone et al., 2016); and AI, Machine Learning and Data Fuel the Future of Productivity by The Goldman Sachs Group, Inc., etc. Brynjolfsson and Mitchell (2017) and Mitchell and Brynjolfsson (2017) discuss implications

of AI and machine learning for workforce. There are also many articles, e.g., in Harvard Business Review, like Agrawal et al. (2017), Ng (2016a), and Ng (2016c). See recent books about the science and technology of AI and machine learning, and their implications for business and society, e.g., Agrawal et al. (2018), Domingos (2015), and Lee (2018).

Nature in May 2015 and Science in July 2015 featured survey papers on machine learning and AI. Science Robotics was launched in 2016. Science has a special issue on July 7, 2017 about AI on The Cyberscientist. Nature Machine Intelligence was launched in January 2019. These illustrate the apparent importance of AI. It is interesting to mention that NIPS 2018 main conference was sold out in less than 12 minutes after opening for registration; see (Li, 2018) .

Deep learning was among MIT Technology Review 10 Breakthrough Technologies in 2013. We have been witnessing the dramatic development of deep learning in both academia and industry in the last few years. Reinforcement learning was among MIT Technology Review 10 Breakthrough Technologies in 2017. Deep learning has made many achievements, has "conquered" speech recognition, computer vision, and now NLP, is more mature and well-accepted, and has been validated by products and market. In contrast, RL has lots of (potential yet promising) applications, yet not many wide-spread products so far. RL may still need better algorithms, and may still need products and market validation. Prediction is very difficult, especially about the future. However, it is probably the right time to nurture, educate and lead the market for reinforcement learning. We will see both deep learning and reinforcement learning prospering in the coming years and beyond.

Deep learning, in this third wave of AI, will have deeper influences, as we have already seen from its many achievements. Reinforcement learning, as a more general learning and decision making paradigm, will deeply influence deep learning, machine learning, and artificial intelligence in general. It is interesting to mention that when Professor Rich Sutton started working in the University of Alberta in 2003, he named his lab RLAI: Reinforcement Learning and Artificial Intelligence.

These abbreviations are used for frequently cited conferences and journals.

AAAI	the AAAI Conference on Artificial Intelligence
ACL	the Association for Computational Linguistics Annual Meeting
AAMAS	the International Conference on Autonomous Agents & Multiagent Systems
ArXiv	ArXiv e-prints
CCS	the ACM Conference on Computer and Communications Security
CVPR	the IEEE Conference on Computer Vision and Pattern Recognition
EMNLP	the Conference on Empirical Methods in Natural Language Processing
ICCV	the IEEE International Conference on Computer Vision
ICLR	the International Conference on Learning Representations
ICML	the International Conference on Machine Learning
ICRA	IEEE International Conference on Robotics and Automation
IJCAI	the International Joint Conference on Artificial Intelligence
IROS	International Conference on Intelligent Robots
JAIR	Journal of Artificial Intelligence Research
JMLR	the Journal of Machine Learning Research
KDD	the ACM International Conference on Knowledge Discovery and Data Mining
NAACL	the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies
NIPS	the Annual Conference on Neural Information Processing Systems
RSS	Robotics: Science and Systems
TNN	IEEE Transactions on Neural Networks
TPAMI	IEEE Transactions on Pattern Analysis and Machine Intelligence
UAI	the Conference on Uncertainty in Artificial Intelligence
WWW	the International World Wide Web Conference

REFERENCES

Abbeel, P. (2017a). Deep learning for robotics. http://goo.gl/oVonGS. NIPS 2017 Invited Talk.

Abbeel, P. (2017b). Reinforcement learning - policy optimization. https://mila.quebec/en/cours/deep-learning-summer-school-2017/. Deep Learning and Reinforcement Learning Summer School 2017.

Abbeel, P. and Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *ICML*.

Abdolmaleki, A., Lioutikov, R., Lau, N., Reis, L. P., Peters, J., and Neumann, G. (2015). Model-based relative entropy stochastic search. In *NIPS*.

Agarwal, D., Chen, B.-C., He, Q., Obukhov?, M., Yang, J., and Zhang, L. (2018). End to-end goal-oriented question answering systems. https://sites.google.com/view/goal-oriented-qa/. KDD 2018 Tutorial.

Agrawal, A., Gans, J., and Goldfarb, A. (2017). How AI will change the way we make decisions. https://hbr.org/2017/07/how-ai-will-change-the-way-we-make-decisions. Harvard Business Review.

Agrawal, A., Gans, J., and Goldfarb, A. (2018). *Prediction Machines: The Simple Economics of Artificial Intelligence*. Harvard Business Review Press.

Agrawal, P., Nair, A., Abbeel, P., Malik, J., and Levine, S. (2016). Learning to poke by poking: Experiential learning of intuitive physics. In *NIPS*.

Al-Shedivat, M., Bansal, T., Burda, Y., Sutskever, I., Mordatch, I., and Abbeel, P. (2018). Continuous adaptation via meta-learning in nonstationary and competitive environments. In *ICLR*.

Alberg, J. and Lipton, Z. C. (2017). Improving factor-based quantitative investing by forecasting company fundamentals. In *NIPS 2017 Time Series Workshop*.

Albrechta, S. V. and Stone, P. (2018). Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*.

Alsheikh, M. A., Lin, S., Niyato, D., and Tan, H.-P. (2014). Machine learning in wireless sensor networks: Algorithms, strategies, and applications. *IEEE Communications Surveys & Tutorials*, 16(4):1996–2018.

Amin, K., Jiang, N., and Singh, S. (2017). Repeated inverse reinforcement learning. In *NIPS*.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. (2016). Concrete Problems in AI Safety. *ArXiv*.

Amos, B., Jimenez, I., Sacks, J., Boots, B., and Kolter, J. Z. (2018). Differentiable MPC for end-to-end planning and control. In *NIPS*.

Anderson, H. S., Kharkar, A., Filar, B., Evans, D., and Roth, P. (2018). Learning to Evade Static PE Machine Learning Malware Models via Reinforcement Learning. *ArXiv*.

Anderson, P., Das, A., and Wu, Q. (2018). Connecting language and vision to actions. https://lvatutorial.github.io. ACL 2018 Tutorial.

Anderson, R. N., Boulanger, A., Powell, W. B., and Scott, W. (2011). Adaptive stochastic control for the smart grid. *Proceedings of the IEEE*, 99(6):1098–1115.

Andreas, J., Klein, D., and Levine, S. (2017). Modular multitask reinforcement learning with policy sketches. In *ICML*.

Andrychowicz, M., Denil, M., Colmenarejo, S. G., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and de Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. In *NIPS*.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. (2017). Hindsight experience replay. In *NIPS*.

Anschel, O., Baram, N., and Shimkin, N. (2017). Averaged-DQN: Variance reduction and stabilization for deep reinforcement learning. In *ICML*.

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. *ArXiv*.

Arora, S. (2018). Toward theoretical understanding of deep learning. `http://unsupervised.cs.princeton.edu/deeplearningtutorial.html`. ICML 2018 Tutorial.

Artetxe, M., Labaka, G., Agirre, E., and Cho, K. (2018). Unsupervised neural machine translation. In *ICLR*.

Asri, L. E., He, J., and Suleman, K. (2016). A sequence-to-sequence model for user simulation in spoken dialogue systems. In *Annual Meeting of the International Speech Communication Association (INTERSPEECH)*.

Athalye, A., Carlini, N., and Wagner, D. (2018). Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*.

Audiffren, J., Valko, M., Lazaric, A., and Ghavamzadeh, M. (2015). Maximum entropy semi-supervised inverse reinforcement learning. In *IJCAI*.

Auer, P. (2002). Using confidence bounds for exploitation-exploration trade-offs. *JMLR*, 3:397–422.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (2002). The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77.

Aytar, Y., Pfaff, T., Budden, D., Paine, T., Wang, Z., and de Freitas, N. (2018). Playing hard exploration games by watching YouTube. In *NIPS*.

Azar, M. G., Osband, I., and Munos, R. (2017). Minimax regret bounds for reinforcement learning. In *ICML*.

Ba, J., Hinton, G. E., Mnih, V., Leibo, J. Z., and Ionescu, C. (2016). Using fast weights to attend to the recent past. In *NIPS*.

Ba, J., Kiros, J. R., and Hinton, G. E. (2016). Layer Normalization. *ArXiv*.

Ba, J., Mnih, V., and Kavukcuoglu, K. (2014). Multiple object recognition with visual attention. In *ICLR*.

Babaeizadeh, M., Frosio, I., Tyree, S., Clemons, J., and Kautz, J. (2017). Reinforcement learning through asynchronous advantage actor-critic on a gpu. In *ICLR*.

Bacon, P.-L., Harb, J., and Precup, D. (2017). The option-critic architecture. In *AAAI*.

Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A., and Bengio, Y. (2017). An actor-critic algorithm for sequence prediction. In *ICLR*.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *ICLR*.

Bai, S., Zico Kolter, J., and Koltun, V. (2018). An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *ArXiv*.

Bailis, P., Olukoton, K., Re, C., and Zaharia, M. (2017). Infrastructure for Usable Machine Learning: The Stanford DAWN Project. *ArXiv*.

Baird, L. (1995). Residual algorithms: Reinforcement learning with function approximation. In *ICML*.

Baker, B., Gupta, O., Naik, N., and Raskar, R. (2017). Designing neural network architectures using reinforcement learning. In *ICLR*.

Balog, M., Gaunt, A. L., Brockschmidt, M., Nowozin, S., and Tarlow, D. (2017). Deepcoder: Learning to write programs. In *ICLR*.

Banino, A., Barry, C., Uria, B., Blundell, C., Lillicrap, T., Mirowski, P., Pritzel, A., Chadwick, M. J., Degris, T., Modayil, J., Wayne, G., Soyer, H., Viola, F., Zhang, B., Goroshin, R., Rabinowitz, N., Pascanu, R., Beattie, C., Petersen, S., Sadik, A., Gaffney, S., King, H., Kavukcuoglu, K., Hassabis, D., Hadsell, R., and Kumaran, D. (2018). Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557:429–433.

Bansal, T., Pachocki, J., Sidor, S., Sutskever, I., and Mordatch, I. (2018). Emergent complexity via multi-agent competition. In *ICLR*.

Bao, W., Yue, J., and Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long- short term memory. *PLoS ONE*, 12(7).

Barreto, A., Munos, R., Schaul, T., and Silver, D. (2017). Successor features for transfer in reinforcement learning. In *NIPS*.

Barth-Maron, G., Hoffman, M. W., Budden, D., Dabney, W., Horgan, D., TB, D., Muldal, A., Heess, N., and Lillicrap, T. (2018). Distributed distributional deterministic policy gradients. In *ICLR*.

Barto, A. (2013). Intrinsic motivation and reinforcement learning. In Baldassarre, G. and Mirolli, M., editors, *Intrinsically Motivated Learning in Natural and Artificial Systems*. Springer, Berlin, Heidelberg.

Barto, A. (2017). Intrinsically motivated reinforcement learning. `http://goo.gl/Q477g2`.

Barto, A. (2018). A history of reinforcement learning. `https://www.youtube.com/watch?v=ul6B2oFPNDM`. IJCAI-17 Award for Research Excellence.

Barto, A. G. and Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(4):341–379.

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(5):835–846.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, F., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., and Pascanu, R. (2018). Relational inductive biases, deep learning, and graph networks. *ArXiv*.

Battaglia, P. W., Pascanu, R., Lai, M., Rezende, D., and Kavukcuoglu, K. (2016). Interaction networks for learning about objects, relations and physics. In *NIPS*.

Bazzan, A. L. and Klügl, F. (2014). *Introduction to Intelligent Systems in Traffic and Transportation*. Morgan & Claypool.

Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., Schrittwieser, J., Anderson, K., York, S., Cant, M., Cain, A., Bolton, A., Gaffney, S., King, H., Hassabis, D., Legg, S., and Petersen, S. (2016). DeepMind Lab. *ArXiv*.

Bellemare, M. G., Castro, P. S., Gelada, C., Kumar, S., and Moitra, S. (2018). Dopamine. `https://github.com/google/dopamine`. A blog at `http://goo.gl/ZETkG6`.

Bellemare, M. G., Dabney, W., and Munos, R. (2017). A distributional perspective on reinforcement learning. In *ICML*.

Bellemare, M. G., Danihelka, I., Dabney, W., Mohamed, S., Lakshminarayanan, B., Hoyer, S., and Munos, R. (2017). The Cramer Distance as a Solution to Biased Wasserstein Gradients. *ArXiv*.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *JAIR*, 47:253–279.

Bellemare, M. G., Schaul, T., Srinivasan, S., Saxton, D., Ostrovski, G., and Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. In *NIPS*.

Belletti, F., Haziza, D., Gomes, G., and Bayen, A. M. (2018). Expert level control of ramp metering based on multi-task deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 19(4):1198–1207.

Bellman, R. (1978). *An Introduction to Artificial Intelligence*. Boyd & Fraser.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S. (2016). Neural Combinatorial Optimization with Reinforcement Learning. *ArXiv*.

Bello, I., Zoph, B., Vasudevan, V., and Le, Q. V. (2017). Neural optimizer search with reinforcement learning. In *ICML*.

Bengio, Y. (2018). From deep learning of disentangled representations to higher-level cognition. `https://www.youtube.com/watch?v=Yr1mOzC93xs`.

Bengio, Y., Bengio, S., and Cloutier, J. (1991). Learning a synaptic learning rule. In *International Joint Conference on Neural Networks (IJCNN)*.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *TPAMI*, 35(8):1798–1828.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *ICML*.

Berkenkamp, F., Turchetta, M., Schoellig, A. P., and Krause, A. (2017). Safe model-based reinforcement learning with stability guarantees. In *NIPS*.

Berthelot, D., Schumm, T., and Metz, L. (2017). BEGAN: Boundary Equilibrium Generative Adversarial Networks. *ArXiv*.

Bertsekas, D. P. (2012). *Dynamic programming and optimal control (Vol. II, 4th Edition: Approximate Dynamic Programming)*. Athena Scientific, Massachusetts, USA.

Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific.

Besold, T. R., d'Avila Garcez, A., Bader, S., Bowman, H., Domingos, P., Hitzler, P., Kuehnberger, K.-U., Lamb, L. C., Lowd, D., Machado Vieira Lima, P., de Penning, L., Pinkas, G., Poon, H., and Zaverucha, G. (2017). Neural-Symbolic Learning and Reasoning: A Survey and Interpretation. *ArXiv*.

Bhatti, S., Desmaison, A., Miksik, O., Nardelli, N., Siddharth, N., and Torr, P. H. S. (2016). Playing Doom with SLAM-Augmented Deep Reinforcement Learning. *ArXiv*.

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., and Lloyd, S. (2017). Quantum machine learning. *Nature*, 549:195–202.

Bishop, C. (2011). *Pattern Recognition and Machine Learning*. Springer.

Blei, D. M. and Smyth, P. (2017). Science and data science. *PNAS*, 114(33):8689–8692.

Bohg, J., Hausman, K., Sankaran, B., Brock, O., Kragic, D., Schaal, S., and Sukhatme, G. S. (2017). Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291.

Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K. (2016). End to End Learning for Self-Driving Cars. *ArXiv*.

Bojarski, M., Yeres, P., Choromanska, A., Choromanski, K., Firner, B., Jackel, L., and Muller, U. (2017). Explaining How a Deep Neural Network Trained with End-to-End Learning Steers a Car. *ArXiv*.

Bordes, A., Boureau, Y.-L., and Weston, J. (2017). Learning end-to-end goal-oriented dialog. In *ICLR*.

Bottou, L. (2014). From machine learning to machine reasoning. *Machine Learning*, 94(2):133–149.

Bottou, L., Curtis, F. E., and Nocedal, J. (2018). Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311.

Botvinick, M., Barrett, D. G. T., Battaglia, P., de Freitas, N., Kumaran, D., Leibo, J. Z., Lillicrap, T., Modayil, J., Mohamed, S., Rabinowitz, N. C., Rezende, D. J., Santoro, A., Schaul, T., Summerfield, C., Wayne, G., Weber, T., Wierstra, D., Legg, S., and Hassabis, D. (2017). Building machines that learn and think for themselves. *Behavioral and Brain Sciences*, 40.

Bousmalis, K., Irpan, A., Wohlhart, P., Bai, Y., Kelcey, M., Kalakrishnan, M., Downs, L., Ibarz, J., Pastor, P., Konolige, K., Levine, S., and Vanhoucke, V. (2017). Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping. *ArXiv*.

Bowling, M., Burch, N., Johanson, M., and Tammelin, O. (2015). Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149.

Bowling, M. and Veloso, M. (2002). Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215–250.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Brachman, R. and Levesque, H. (2004). *Knowledge Representation and Reasoning*. Morgan Kaufmann.

Bradtke, S. J. and Barto, A. G. (1996). Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1-3):33–57.

Brafman, R. I. and Tennenholtz, M. (2002). R-MAX - a general polynomial time algorithm for near-optimal reinforcement learning. *JMLR*, 3:213–231.

Branavan, S. R. K., Silver, D., and Barzilay, R. (2012). Learning to win by reading manuals in a monte-carlo framework. *JAIR*, 43.

Brandt, M. W., Goyal, A., Santa-Clara, P., and Stroud, J. R. (2005). A simulation approach to dynamic portfolio choice with an application to learning about return predictability. *The Review of Financial Studies*, 18(3):831–873.

Brazdil, P., Carrier, C. G., Soares, C., and Vilalta, R. (2009). *Metalearning: Applications to Data Mining*. Springer.

Briot, J.-P., Hadjeres, G., and Pachet, F. (2018). *Deep Learning Techniques for Music Generation*. Springer International Publishing.

Britz, D. (2016). Attention and memory in deep learning and nlp. `http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/`.

Brock, A., Donahue, J., and Simonyan, K. (2018). Large Scale GAN Training for High Fidelity Natural Image Synthesis. *ArXiv e-prints*.

Brown, N. and Sandholm, T. (2017a). Safe and nested subgame solving for imperfect-information games. In *NIPS*.

Brown, N. and Sandholm, T. (2017b). Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*.

Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. (2012). A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43.

Brunner, G., Richter, O., Wang, Y., and Wattenhofer, R. (2018). Teaching a machine to read maps with deep reinforcement learning. In *AAAI*.

Brunskill, E. (2017). Reinforcement learning with people. `https://www.facebook.com/nipsfoundation/videos/1555771847847382/`. NIPS 2017 Tutorial.

Brynjolfsson, E. and Mitchell, T. (2017). What can machine learning do? workforce implications. *Science*, 358(6370):1530–1534.

Bucila, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model compression. In *KDD*.

Buckman, J., Hafner, D., Tucker, G., Brevdo, E., and Lee, H. (2018). Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *NIPS*.

Buro, M. (1999). How machines have learned to play othello. *IEEE Intelligent Systems Journal*, 14(6):12–14.

Busoniu, L., Babuska, R., and Schutter, B. D. (2008). A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 38(2).

Cai, H., Yang, J., Zhang, W., Han, S., and Yu, Y. (2018a). Path-level network transformation for efficient architecture search. In *ICML*.

Cai, Q., Filos-Ratsikas, A., Tang, P., and Zhang, Y. (2018b). Reinforcement mechanism design for fraudulent behaviour in e-commerce. In *AAAI*.

Caicedo, J. C. and Lazebnik, S. (2015). Active object localization with deep reinforcement learning. In *ICCV*.

Campbell, J. Y. and Viceira, L. M. (2002). *Strategic Asset Allocation Portfolio Choice for Long-Term Investors*. Oxford University Press.

Campbell, M., Hoane, A. J., and Hsu, F. (2002). Deep blue. *Artificial Intelligence*, 134:57–83.

Cao, Q., Lin, L., Shi, Y., Liang, X., and Li, G. (2017). Attention-aware face hallucination via deep reinforcement learning. In *CVPR*.

Carleo, G. and Troyer, M. (2017). Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606.

Caruana, R. (1997). Multitask learning. *Machine Learning*, 28(1):41–75.

Chakraborty, B. and Murphy, S. A. (2014). Dynamic treatment regimes. *Annual Review of Statistics and Its Application*, 1:447–464.

Chan, C., Ginosar, S., Zhou, T., and Efros, A. A. (2018). Everybody Dance Now. *ArXiv*.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection : A survey. *ACM Computing Surveys*, 41(3):1–72.

Charniak, E. and McDermott, D. (1985). *Introduction to Artificial Intelligence Programming*. Addison Wesley.

Chebotar, Y., Hausman, K., Zhang, M., Sukhatme, G., Schaal, S., and Levine, S. (2017). Combining model-based and model-free updates for trajectory-centric reinforcement learning. In *ICML*.

Chen, K. and Bowling, M. (2012). Tractable objectives for robust policy optimization. In *NIPS*.

Chen, L.-C., Collins, M., Zhu, Y., Papandreou, G., Zoph, B., Schroff, F., Adam, H., and Shlens, J. (2018a). Searching for efficient multi-scale architectures for dense image prediction. In *NIPS*.

Chen, T., Murali, A., and Gupta, A. (2018b). Hardware conditioned policies for multi-robot transfer learning. In *NIPS*.

Chen, T., Zheng, L., Yan, E., Jiang, Z., Moreau, T., Ceze, L., Guestrin, C., and Krishnamurthy, A. (2018c). Learning to optimize tensor programs. In *NIPS*.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. (2018). Neural Ordinary Differential Equations. *ArXiv*.

Chen, X. and Deng, X. (2006). Settling the complexity of two-player nash equilibrium. In *IEEE Symposium on Foundations of Computer Science (FOCS)*.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016a). InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*.

Chen, X., Li, L.-J., Fei-Fei, L., and Gupta, A. (2018a). Iterative visual reasoning beyond convolutions. In *CVPR*.

Chen, Y., Li, L., and Wang, M. (2018b). Scalable bilinear π learning using state and action features. In *ICML*.

Chen, Y.-N. V., Hakkani-Tür, D., Tur, G., Gao, J., and Deng, L. (2016b). End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In *Annual Meeting of the International Speech Communication Association (INTERSPEECH)*.

Chen, Z. and Liu, B. (2016). *Lifelong Machine Learning*. Morgan & Claypool.

Chen, Z. and Yi, D. (2017). The Game Imitation: Deep Supervised Convolutional Networks for Quick Video Game AI. *ArXiv*.

Cheng, Y., Xu, W., He, Z., He, W., Wu, H., Sun, M., and Liu, Y. (2016). Semi-supervised learning for neural machine translation. In *ACL*.

Cho, K. (2015). Natural Language Understanding with Distributed Representation. *ArXiv*.

Cho, K., van Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.

Choi, E., Hewlett, D., Polosukhin, I., Lacoste, A., Uszkoreit, J., and Berant, J. (2017). Coarse-to-fine question answering for long documents. In *ACL*.

Chopra, S., Auli, M., and Rush, A. M. (2016). Abstractive sentence summarization with attentive recurrent neural networks. In *NAACL*.

Chow, Y., Nachum, O., Ghavamzadeh, M., and Duenez-Guzman, E. (2018). A Lyapunov-based approach to safe reinforcement learning. In *NIPS*.

Christiano, P., Leike, J., Brown, T. B., Martic, M., Legg, S., and Amodei, D. (2017). Deep reinforcement learning from human preferences. In *NIPS*.

Chua, K., Calandra, R., McAllister, R., and Levine, S. (2018). Data-efficient model-based reinforcement learning with deep probabilistic dynamics models. In *NIPS*.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Deep Learning and Representation Learning Workshop*.

Cook, T. R. and Hall, A. S. (2017). Macroeconomic indicator forecasting with deep neural networks. *Federal Reserve Bank of Kansas City, Research Working Paper 17-11*.

Cranmer, K. (2016). Machine learning and likelihood-free inference in particle physics. `https://bit.ly/2nJGSuf`. NIPS 2016 Invited Talk.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. (2018). AutoAugment: Learning Augmentation Policies from Data. *ArXiv e-prints*.

Cueva, C. J. and Wei, X.-X. (2018). Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. In *ICLR*.

Dabney, W., Ostrovski, G., Silver, D., and Munos, R. (2018). Implicit quantile networks for distributional reinforcement learning. In *ICML*.

Dai, B., Shaw, A., He, N., Li, L., and Song, L. (2018a). Boosting the actor with dual critic. In *ICLR*.

Dai, B., Shaw, A., Li, L., Xiao, L., He, N., Liu, Z., Chen, J., and Song, L. (2018b). SBEED: Convergent reinforcement learning with nonlinear function approximation. In *ICML*.

Dai, H., Dai, B., and Song, L. (2016). Discriminative embeddings of latent variable models for structured data. In *ICML*.

Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B., and Song, L. (2017). Learning combinatorial optimization algorithms over graphs. In *NIPS*.

Dai, Z., Yang, Z., Yang, F., Cohen, W. W., and Salakhutdinov, R. (2017). Good semi-supervised learning that requires a bad gan. In *NIPS*.

Danihelka, I., Wayne, G., Uria, B., Kalchbrenner, N., and Graves, A. (2016). Associative long short-term memory. In *ICML*.

Darwiche, A. (2018). Human-level intelligence or animal-like abilities? *Communications of the ACM*, 61(10):56–67.

Das, A., Kottur, S., Moura, J. M. F., Lee, S., and Batra, D. (2017). Learning cooperative visual dialog agents with deep reinforcement learning. In *ICCV*.

Daumé, III, H., Langford, J., and Marcu, D. (2009). Search-based structured prediction. *Machine Learning*, 75(3):297–325.

Dayan, P. (1993). Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624.

Dayan, P. and Hinton, G. E. (1993). Feudal reinforcement learning. In *NIPS*.

De Asis, K., Hernandez-Garcia, J. F., Zacharias Holland, G., and Sutton, R. S. (2018). Multi-step reinforcement learning: A unifying algorithm. In *AAAI*.

de Avila Belbute-Peres, F., Smith, K., Allen, K., Tenenbaum, J., and Kolter, J. Z. (2018). End-to-end differentiable physics for learning and control. In *NIPS*.

Degris, T., White, M., and Sutton, R. S. (2012). Off-policy actor-critic. In *ICML*.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, Ł. (2018). Universal Transformers. *ArXiv*.

Deisenroth, M. P., Neumann, G., and Peters, J. (2013). A survey on policy search for robotics. *Foundations and Trend in Robotics*, 2:1–142.

Deisenroth, M. P. and Rasmussen, C. E. (2011). PILCO: A model-based and data-efficient approach to policy search. In *ICML*.

Deng, L. (2017). Three generations of spoken dialogue systems (bots), talk at AI Frontiers Conference. `https://www.slideshare.net/AIFrontiers/li-deng-three-generations-of-spoken-dialogue-systems-bots`.

Deng, L. and Li, X. (2013). Machine learning paradigms for speech recognition: An overview. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(5):1060–1089.

Deng, L. and Liu, Y., editors (2018). *Deep Learning in Natural Language Processing*. Springer.

Deng, Y., Bao, F., Kong, Y., Ren, Z., and Dai, Q. (2016). Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems*.

Denil, M., Agrawal, P., Kulkarni, T. D., Erez, T., Battaglia, P., and de Freitas, N. (2017). Learning to perform physics experiments via deep reinforcement learning. In *ICLR*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv e-prints*.

DeVries, P. M. R., Viégas, F., Wattenberg, M., and Meade, B. J. (2018). Deep learning of aftershock patterns following large earthquakes. *Nature*, 560:632–634.

Devrim Kaba, M., Gokhan Uzunbas, M., and Nam Lim, S. (2017). A reinforcement learning approach to the view planning problem. In *CVPR*.

Dhingra, B., Li, L., Li, X., Gao, J., Chen, Y.-N., Ahmed, F., and Deng, L. (2017). End-to-end reinforcement learning of dialogue agents for information access. In *ACL*.

Diederik P Kingma, M. W. (2014). Auto-encoding variational bayes. In *ICLR*.

Dieleman, S., van den Oord, A., and Simonyan, K. (2018). The challenge of realistic music generation: modelling raw audio at scale. In *NIPS*.

Dietterich, T. G. (2000). Hierarchical reinforcement learning with the MAXQ value function decomposition. *JAIR*, 13(1):227–303.

Dietterich, T. G. (2018). Reflections on innateness in machine learning. `https://medium.com/@tdietterich/reflections-on-innateness-in-machine-learning-4eebefa3e1af`.

Dimakopoulou, M., Osband, I., and Roy, B. V. (2018). Scalable coordinated exploration in concurrent reinforcement learning. In *NIPS*.

Diuk, C., Cohen, A., and Littman, M. L. (2008). An object-oriented representation for efficient reinforcement learning. In *ICML*.

Doan, T., Mazoure, B., and Lyle, C. (2018). GAN Q-learning. *ArXiv*.

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87.

Domingos, P. (2015). *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World*. Basic Books.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*.

Dong, D., Wu, H., He, W., Yu, D., and Wang, H. (2015). Multi-task learning for multiple language translation. In *ACL*.

Dong, S. and Roy, B. V. (2018). An information-theoretic analysis of Thompson sampling for large action spaces. In *NIPS*.

Doshi-Velez, F. and Kim, B. (2017). Towards A Rigorous Science of Interpretable Machine Learning. *ArXiv*.

Dosovitskiy, A. and Koltun, V. (2017). Learning to act by predicting the future. In *ICLR*.

Downey, C., Hefny, A., Li, B., Boots, B., and Gordon, G. (2017). Predictive state recurrent neural networks. In *NIPS*.

Du, S. S., Chen, J., Li, L., Xiao, L., and Zhou, D. (2017). Stochastic variance reduction methods for policy evaluation. In *ICML*.

Duan, R. (2017). *Meta Learning for Control*. University of California at Berkeley PhD Thesis.

Duan, Y., Andrychowicz, M., Stadie, B. C., Ho, J., Schneider, J., Sutskever, I., Abbeel, P., and Zaremba, W. (2017). One-shot imitation learning. In *NIPS*.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. (2016). Benchmarking deep reinforcement learning for continuous control. In *ICML*.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. (2016). RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning. *ArXiv*.

Džeroski, S., Raedt, L. D., and Driessens, K. (2001). Relational reinforcement learning. *Machine Learning*, 43((1/2)):7–52.

Efros, A. (2017). Self-supervised deep learning. `https://www.youtube.com/watch?v=YhYsvD6IfKE`.

El-Tantawy, S., Abdulhai, B., and Abdelgawad, H. (2013). Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): methodology and large-scale application on downtown toronto. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1140–1150.

Ellis, H. C. (1965). *The transfer of learning*. Macmillan.

Elsken, T., Hendrik Metzen, J., and Hutter, F. (2018). Neural Architecture Search: A Survey. *ArXiv*.

Ernst, D., Geurts, P., and Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. *JMLR*, 6:503–556.

Eslami, S. M. A., Heess, N., Weber, T., Tassa, Y., Szepesvári, D., Kavukcuoglu, K., and Hinton, G. E. (2016). Attend, infer, repeat: Fast scene understanding with generative models. In *NIPS*.

Eslami, S. M. A., Rezende, D. J., Besse, F., Viola, F., Morcos, A. S., Garnelo, M., Ruderman, A., Rusu, A. A., Danihelka, I., Gregor, K., Reichert, D. P., Buesing, L., Weber, T., Vinyals, O., Rosenbaum, D., Rabinowitz, N., King, H., Hillier, C., Botvinick, M., Wierstra, D., Kavukcuoglu, K., and Hassabis, D. (2018). Neural scene representation and rendering. *Science*, 360:1204–1210.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. (2018). IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. *ArXiv*.

Evans, R. and Grefenstette, E. (2018). Learning explanatory rules from noisy data. *JAIR*, 61(1):1–64.

Evtimov, I., Eykholt, K., Fernandes, E., Kohno, T., Li, B., Prakash, A., Rahmati, A., and Song, D. (2017). Robust Physical-World Attacks on Deep Learning Models. *ArXiv*.

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., and Song, D. (2018). Robust physical-world attacks on deep learning models. In *CVPR*.

Fang, M., Li, Y., and Cohn, T. (2017). Learning how to active learn: A deep reinforcement learning approach. In *EMNLP*.

Fang, X., Misra, S., Xue, G., and Yang, D. (2012). Smart grid - the new and improved power grid: A survey. *IEEE Communications Surveys Tutorials*, 14(4):944–980.

Farahmand, A.-m. (2018). Iterative value-aware model learning. In *NIPS*.

Farebrother, J., Machado, M. C., and Bowling, M. (2018). Generalization and Regularization in DQN. *ArXiv e-prints*.

Farquhar, G., Rocktäschel, T., Igl, M., and Whiteson, S. (2018). TreeQN and ATreeC: Differentiable tree-structured models for deep reinforcement learning. In *ICLR*.

Fatemi, M., Asri, L. E., Schulz, H., He, J., and Suleman, K. (2016). Policy networks with two-stage training for dialogue systems. In *the Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL)*.

Faust, A., Aimone, J. B., James, C. D., and Tapia, L. (2018). Resilient Computing with Reinforcement Learning on a Dynamical System: Case Study in Sorting. *ArXiv e-prints*.

Fauw, J. D., Ledsam, J. R., Romera-Paredes, B., Nikolov, S., Tomasev, N., Blackwell, S., Askham, H., Glorot, X., O'Donoghue, B., Visentin, D., van den Driessche, G., Lakshminarayanan, B., Meyer, C., Mackinder, F., Bouton, S., Ayoub, K., Chopra, R., King, D., Karthikesalingam, A., Hughes, C. O., Raine, R., Hughes, J., Sim, D. A., Egan, C., Tufail, A., Montgomery, H., Hassabis, D., Rees, G., Back, T., Khaw, P. T., Suleyman, M., Cornebise, J., Keane, P. A., and Ronneberger, O. (2018). Clinically applicable deep learning for diagnosis and referral in retinal disease. *Nature Medicine*.

Feng, J. and Zhou, Z.-H. (2017). AutoEncoder by Forest. *ArXiv*.

Finn, C. (2017). Model-based RL. `http://goo.gl/Fsd5n8`. Deep RL Bootcamp.

Finn, C., Abbeel, P., and Levine, S. (2017a). Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.

Finn, C., Christiano, P., Abbeel, P., and Levine, S. (2016a). A connection between GANs, inverse reinforcement learning, and energy-based models. In *NIPS 2016 Workshop on Adversarial Training*.

Finn, C. and Levine, S. (2015). Deep visual foresight for planning robot motion. In *ICRA*.

Finn, C. and Levine, S. (2017). Deep visual foresight for planning robot motion. In *ICRA*.

Finn, C. and Levine, S. (2018). Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. In *ICLR*.

Finn, C., Levine, S., and Abbeel, P. (2016b). Guided cost learning: Deep inverse optimal control via policy optimization. In *ICML*.

Finn, C., Xu, K., and Levine, S. (2018). Probabilistic model-agnostic meta-learning. In *NIPS*.

Finn, C., Yu, T., Fu, J., Abbeel, P., and Levine, S. (2017b). Generalizing skills with semi-supervised reinforcement learning. In *ICLR*.

Finn, C., Yu, T., Zhang, T., Abbeel, P., and Levine, S. (2017c). One-shot visual imitation learning via meta-learning. In *Conference on Robot Learning*.

Firoiu, V., Whitney, W. F., and Tenenbaum, J. B. (2017). Beating the World's Best at Super Smash Bros. with Deep Reinforcement Learning. *ArXiv*.

Florensa, C., Duan, Y., and Abbeel, P. (2017). Stochastic neural networks for hierarchical reinforcement learning. In *ICLR*.

Foerster, J., Assael, Y. M., de Freitas, N., and Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. In *NIPS*.

Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. (2018a). Counterfactual multi-agent policy gradients. In *AAAI*.

Foerster, J., Nardelli, N., Farquhar, G., Torr, P. H. S., Kohli, P., and Whiteson, S. (2017). Stabilising experience replay for deep multi-agent reinforcement learning. In *ICML*.

Foerster, J. N., Chen, R. Y., Al-Shedivat, M., Whiteson, S., Abbeel, P., and Mordatch, I. (2018b). Learning with opponent-learning awareness. In *AAMAS*.

Fortunato, M., Gheshlaghi Azar, M., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., and Legg, S. (2018). Noisy networks for exploration. In *ICLR*.

Fox, R., Pakman, A., and Tishby, N. (2016). Taming the noise in reinforcement learning via soft updates. In *the Conference on Uncertainty in Artificial Intelligence (UAI)*.

Fridman, L., Jenik, B., and Terwilliger, J. (2018). DeepTraffic: Driving Fast through Dense Traffic with Deep Reinforcement Learning. *ArXiv*.

Fu, J., Co-Reyes, J. D., and Levine, S. (2017). Ex2: Exploration with exemplar models for deep reinforcement learning. In *NIPS*.

Gao, C., Hayward, R. B., and Müller, M. (2017). Move prediction using deep convolutional neural networks in hex. *IEEE Transactions on Games*. `https://github.com/cgao3/benzene-vanilla-cmake`.

Gao, J., Galley, M., and Li, L. (2018a). Neural approaches to Conversational AI. *Foundations and Trends in Information Retrieval*. To appear.

Gao, J., Galley, M., and Li, L. (2018b). Neural approaches to Conversational AI. `https://www.microsoft.com/en-us/research/publication/neural-approaches-to-conversational-ai/`. ACL 2018 Tutorial.

Gao, Y., Chen, L., and Li, B. (2018c). Post: Device placement with cross-entropy minimization and proximal policy optimization. In *NIPS*.

Garcìa, J. and Fernàndez, F. (2015). A comprehensive survey on safe reinforcement learning. *JMLR*, 16:1437–1480.

Gavrilovska, L., Atanasovski, V., Macaluso, I., and DaSilva, L. A. (2013). Learning and reasoning in cognitive radio networks. *IEEE Communications Surveys Tutorials*, 15(4):1761–1777.

Geffner, H. (2018). Model-free, model-based, and general intelligence. In *IJCAI*.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional Sequence to Sequence Learning. *ArXiv*.

Gelly, S., Schoenauer, M., Sebag, M., Teytaud, O., Kocsis, L., Silver, D., and Szepesvári, C. (2012). The grand challenge of computer go: Monte carlo tree search and extensions. *Communications of the ACM*, 55(3):106–113.

Gelly, S. and Silver, D. (2007). Combining online and offline knowledge in UCT. In *ICML*.

George, D., Lehrach, W., Kansky, K., Lázaro-Gredilla, M., Laan, C., Marthi, B., Lou, X., Meng, Z., Liu, Y., Wang, H., Lavin, A., and Phoenix, D. S. (2017). A generative vision model that trains with high data efficiency and breaks text-based CAPTCHAs. *Science*.

Gershman, S. J. (2018). The successor representation: Its computational logic and neural substrates. *Journal of Neuroscience*, 38(33):7193–7200.

Getoor, L. and Taskar, B., editors (2007). *Introduction to Statistical Relational Learning*. MIT Press.

Ghavamzadeh, M., Engel, Y., and Valko, M. (2016). Bayesian policy gradient and actor-critic algorithms. *JMLR*, 17(66):1–53.

Ghavamzadeh, M., Mahadevan, S., and Makar, R. (2006). Hierarchical multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 13(2):197–229.

Ghavamzadeh, M., Mannor, S., Pineau, J., and Tamar, A. (2015). Bayesian reinforcement learning: a survey. *Foundations and Trends in Machine Learning*, 8(5-6):359–483.

Gheiratmand, M., Rish, I., Cecchi, G. A., Brown, M. R. G., Greiner, R., Polosecki, P. I., Bashivan, P., Greenshaw, A. J., Ramasubbu, R., and Dursun, S. M. (2017). Learning stable and predictive network-based patterns of schizophrenia and its clinical symptoms. *Nature Schizophrenia*, 3(22).

Ginsberg, M. L. (2001). GIB: Imperfect information in a computationally challenging game. *JAIR*, 14:303–358.

Girshick, R. (2015). Fast R-CNN. In *ICCV*.

Glasserman, P. (2004). *Monte Carlo Methods in Financial Engineering*. Springer-Verlag, New York.

Glavic, M., Fonteneau, R., and Ernst, D. (2017). Reinforcement learning for electric power system decision and control: Past considerations and perspectives. In *The 20th World Congress of the International Federation of Automatic Control*.

Goel, V., Weng, J., and Poupart, P. (2018). Unsupervised video object segmentation for deep reinforcement learning. In *NIPS*.

Goldberg, Y. (2017). *Neural Network Methods for Natural Language Processing*. Morgan & Claypool.

Goldberg, Y. and Kosorok, M. R. (2012). Q-learning with censored data. *Annals of Statistics*, 40(1):529–560.

Goodfellow, I. (2017). NIPS 2016 Tutorial: Generative Adversarial Networks. *ArXiv*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., , and Bengio, Y. (2014). Generative adversarial nets. In *NIPS*.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *ICLR*.

Grant, E., Finn, C., Levine, S., Darrell, T., and Griffiths, T. (2018). Recasting gradient-based meta-learning as hierarchical bayes. In *ICLR*.

Graves, A., rahman Mohamed, A., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural Turing Machines. *ArXiv*.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J., nech Badia, A. P., Hermann, K. M., Zwols, Y., Ostrovski, G., Cain, A., King, H., Summerfield, C., Blunsom, P., Kavukcuoglu, K., and Hassabis, D. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature*, 538:471–476.

Gregor, K., Danihelka, I., Graves, A., Rezende, D., and Wierstra, D. (2015). Draw: A recurrent neural network for image generation. In *ICML*.

Gruslys, A., Gheshlaghi Azar, M., Bellemare, M. G., and Munos, R. (2017). The Reactor: A Sample-Efficient Actor-Critic Architecture. *ArXiv*.

Gu, S., Holly, E., Lillicrap, T., and Levine, S. (2017a). Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *ICRA*.

Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., and Levine, S. (2017b). Q-Prop: Sample-efficient policy gradient with an off-policy critic. In *ICLR*.

Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., Schölkopf, B., and Levine, S. (2017). Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning. In *NIPS*.

Gu, S., Lillicrap, T., Sutskever, I., and Levine, S. (2016). Continuous deep Q-learning with model-based acceleration. In *ICML*.

Guestrin, C., Koller, D., Gearhart, C., and Kanodia, N. (2003). Generalizing plans to new environments in relational MDPs. In *IJCAI*.

Guez, A., Heess, N., Silver, D., and Dayan, P. (2014). Bayes-adaptive simulation-based search with value function approximation. In *NIPS*.

Guez, A., Weber, T., Antonoglou, I., Simonyan, K., Vinyals, O., Wierstra, D., Munos, R., and Silver, D. (2018). Learning to search with MCTSnets. In *ICML*.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. (2017). Improved training of wasserstein gans. In *NIPS*.

Guo, R., Cheng, L., Li, J., Hahn, P. R., and Liu, H. (2018). A Survey of Learning Causality with Data: Problems and Methods. *ArXiv e-prints*.

Guo, X., Singh, S., Lee, H., Lewis, R. L., and Wang, X. (2014). Deep learning for real-time atari game play using offline monte-carlo tree search planning. In *NIPS*.

Gupta, A. (2017). Supersizing self-supervision: Learning perception and action without human supervision. `https://simons.berkeley.edu/talks/abhinav-gupta-2017-3-28`.

Gupta, A., Devin, C., Liu, Y., Abbeel, P., and Levine, S. (2017a). Learning invariant feature spaces to transfer skills with reinforcement learning. In *ICLR*.

Gupta, A., Mendonca, R., Liu, Y., Abbeel, P., and Levine, S. (2018). Meta-reinforcement learning of structured exploration strategies. In *NIPS*.

Gupta, S., Davidson, J., Levine, S., Sukthankar, R., and Malik, J. (2017b). Cognitive mapping and planning for visual navigation. In *CVPR*.

Guu, K., Pasupat, P., Liu, E. Z., and Liang, P. (2017). From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *ACL*.

Ha, D. and Eck, D. (2018). A neural representation of sketch drawings. In *ICLR*.

Ha, D. and Schmidhuber, J. (2018). Recurrent world models facilitate policy evolution. In *NIPS*.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. (2017). Reinforcement learning with deep energy-based policies. In *ICML*.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*.

Haber, N., Mrowca, D., Wang, S., Fei-Fei, L., and Yamins, D. (2018). Learning to play with intrinsically-motivated, self-aware agents. In *NIPS*.

Hadfield-Menell, D., Dragan, A., Abbeel, P., and Russell, S. (2016). Cooperative inverse reinforcement learning. In *NIPS*.

Hadfield-Menell, D., Milli, S., Abbeel, P., Russell, S., and Dragan, A. (2017). Inverse reward design. In *NIPS*.

Han, J., Kamber, M., and Pei, J. (2011). *Data Mining: Concepts and Techniques (3rd edition)*. Morgan Kaufmann.

Hansen, N. (2016). The CMA Evolution Strategy: A Tutorial. *ArXiv*.

Hansen, N. and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195.

Hartford, J., Lewis, G., Leyton-Brown, K., and Taddy, M. (2017). Deep IV: A flexible approach for counterfactual prediction. In *ICML*.

Harutyunyan, A., Vrancx, P., Bacon, P.-L., Precup, D., and Nowe, A. (2018). Learning with options that terminate off-policy. In *AAAI*.

Hassabis, D., Kumaran, D., Summerfield, C., and Botvinick, M. (2017). Neuroscience-inspired artificial intelligence. *Neuron*, 95:245–258.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

Haugeland, J. (1989). *Artificial Intelligence: The Very Idea*. The MIT Press.

Hausknecht, M. and Stone, P. (2015). Deep recurrent Q-learning for partially observable MDPs. In *AAAI*.

Hausknecht, M. and Stone, P. (2016). Deep reinforcement learning in parameterized action space. In *ICLR*.

Havens, A., Jiang, Z., and Sarkar, S. (2018). Online robust policy learning in the presence of unknown adversaries. In *NIPS*.

Haykin, S. (2005). Cognitive radio: brain-empowered wireless communications. *IEEE Journal on Selected Areas in Communications*, 23(2):201–220.

He, D., Xia, Y., Qin, T., Wang, L., Yu, N., Liu, T.-Y., and Ma, W.-Y. (2016a). Dual learning for machine translation. In *NIPS*.

He, F. S., Liu, Y., Schwing, A. G., and Peng, J. (2017). Learning to play in a day: Faster deep reinforcement learning by optimality tightening. In *ICLR*.

He, J., Chen, J., He, X., Gao, J., Li, L., Deng, L., and Ostendorf, M. (2016b). Deep reinforcement learning with a natural language action space. In *ACL*.

He, J., Ostendorf, M., He, X., Chen, J., Gao, J., Li, L., and Deng, L. (2016c). Deep reinforcement learning with a combinatorial action space for predicting popular reddit threads. In *EMNLP*.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R-CNN. In *ICCV*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016d). Deep residual learning for image recognition. In *CVPR*.

He, L., Lee, K., Lewis, M., and Zettlemoyer, L. (2017). Deep semantic role labeling: What works and what's next. In *ACL*.

He, X. and Deng, L. (2013). Speech-centric information processing: An optimization-oriented approach. *Proceedings of the IEEE — Vol. 101, No. 5, May 2013*, 101(5):1116–1135.

He, Y., Lin, J., Liu, Z., Wang, H., Li, L.-J., and Han, S. (2018). Amc: Automl for model compression and acceleration on mobile devices. In *ECCV*.

Heaton, J. B., Polson, N. G., and Witte, J. H. (2016). Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*.

Heess, N., TB, D., Sriram, S., Lemmon, J., Merel, J., Wayne, G., Tassa, Y., Erez, T., Wang, Z., Eslami, A., Riedmiller, M., and Silver, D. (2017). Emergence of Locomotion Behaviours in Rich Environments. *ArXiv*.

Heess, N., Wayne, G., Silver, D., Lillicrap, T., Tassa, Y., and Erez, T. (2015). Learning continuous control policies by stochastic value gradients. In *NIPS*.

Heinrich, J. and Silver, D. (2016). Deep reinforcement learning from self-play in imperfect-information games. In *NIPS 2016 Deep Reinforcement Learning Workshop*.

Henaff, M., Whitney, W. F., and LeCun, Y. (2017). Model-Based Planning in Discrete Action Spaces. *ArXiv*.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2018). Deep reinforcement learning that matters. In *AAAI*.

Hernandez, D. and Greenwald, T. (2018). IBM has a Watson dilemma. `https://on.wsj.com/2nrlp7g`.

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. (2018). Rainbow: Combining Improvements in Deep Reinforcement Learning. In *AAAI*.

Hester, T. and Stone, P. (2017). Intrinsically motivated model learning for developing curious robots. *Artificial Intelligence*, 247:170–86.

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Dulac-Arnold, G., Osband, I., Agapiou, J., Leibo, J. Z., and Gruslys, A. (2018). Deep Q-learning from demonstrations. In *AAAI*.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). β-VAE: Learning basic visual concepts with a constrained variational framework. In *ICLR*.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., rahman Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., , and Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 82.

Hinton, G., Vinyals, O., and Dean, J. (2014). Distilling the knowledge in a neural network. In *NIPS 2014 Deep Learning Workshop*.

Hinton, G. E., Sabour, S., and Frosst, N. (2018). Matrix capsules with EM routing. In *ICLR*.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hirschberg, J. and Manning, C. D. (2015). Advances in natural language processing. *Science*, 349(6245):261–266.

Ho, J. and Ermon, S. (2016). Generative adversarial imitation learning. In *NIPS*.

Ho, J., Gupta, J. K., and Ermon, S. (2016). Model-free imitation learning with policy optimization. In *ICML*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9:1735–1780.

Hochreiter, S., Younger, A. S., and Conwell, P. R. (2001). Learning to learn using gradient descent. In *ICANN*.

Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., van Hasselt, H., and Silver, D. (2018). Distributed prioritized experience replay. In *ICLR*.

Horling, B. and Lesser, V. (2004). A survey of multi-agent organizational paradigms. *Knowledge Engineering Review*, 19(4):281–316.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Hoshen, Y. (2017). Vain: Attentional multi-agent predictive modeling. In *NIPS*.

Houthooft, R., Chen, X., Duan, Y., Schulman, J., Turck, F. D., and Abbeel, P. (2016). Vime: Variational information maximizing exploration. In *NIPS*.

Houthooft, R., Chen, Y., Isola, P., Stadie, B., Wolski, F., Ho, J., and Abbeel, P. (2018). Evolved policy gradients. In *NIPS*.

Hsu, K., Levine, S., and Finn, C. (2018). Unsupervised Learning via Meta-Learning. *ArXiv e-prints*.

Hu, J. and Wellman, M. P. (2003). Nash q-learning for general-sum stochastic games. *JMLR*, 4:1039–1069.

Hu, Y., Da, Q., Zeng, A., Yu, Y., and Xu, Y. (2018a). Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In *KDD*.

Hu, Y., Li, J., Li, X., Pan, G., and Xu, M. (2018b). Knowledge-guided agent-tactic-aware learning for starcraft micromanagement. In *IJCAI*.

Hu, Z., Liang, Y., Liu, Y., and Zhang, J. (2018c). Inference aided reinforcement learning for incentive mechanism design in crowdsourcing. In *NIPS*.

Hu, Z., Yang, Z., Salakhutdinov, R., Qin, L., Liang, X., Dong, H., and Xing, E. (2018d). Deep generative models with learnable knowledge constraints. In *NIPS*.

Hu, Z., Yang, Z., Salakhutdinov, R., and Xing, E. P. (2017). On Unifying Deep Generative Models. *ArXiv*.

Huang, J., Wu, F., Precup, D., and Cai, Y. (2018). Learning safe policies with expert guidance. In *NIPS*.

Huang, S., Papernot, N., Goodfellow, I., Duan, Y., and Abbeel, P. (2017). Adversarial attacks on neural network policies. In *ICLR Workshop Track*.

Huang, S.-C., Arneson, B., Hayward, R. B., and Müller, M. (2013). Mohex 2.0: A pattern-based mcts hex player. In *International Conference on Computers and Games*.

Hudson, D. A. and Manning, C. D. (2018). Compositional attention networks for machine reasoning. In *ICLR*.

Hughes, E., Leibo, J., Phillips, M., karl Tuyls, Dueñez-Guzman, E., Castañeda, A. G., Dunning, I., Zhu, T., McKee, K., Koster, R., Roff, H., and Graepel, T. (2018). Inequity aversion improves cooperation in intertemporal social dilemmas. In *NIPS*.

Hull, J. C. (2014). *Options, Futures and Other Derivatives (9th edition)*. Prentice Hall.

Hutson, M. (2018). Basic instincts. *Science*, 360(6391):845–847.

Hutter, M. (2005). *Universal Artificial Intelligence: Sequential Decisions Based on Algorithmic Probability*. Springer, Berlin.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*.

Irving, G., Szegedy, C., Alemi, A. A., Een, N., Chollet, F., and Urban, J. (2016). Deepmath - deep sequence models for premise selection. In *NIPS*.

Jaderberg, M., Czarnecki, W. M., Dunning, I., Marris, L., Lever, G., Garcia Castaneda, A., Beattie, C., Rabinowitz, N. C., Morcos, A. S., Ruderman, A., Sonnerat, N., Green, T., Deason, L., Leibo, J. Z., Silver, D., Hassabis, D., Kavukcuoglu, K., and Graepel, T. (2018). Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *ArXiv*.

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., Fernando, C., and Kavukcuoglu, K. (2017). Population Based Training of Neural Networks. *ArXiv*.

Jaderberg, M., Mnih, V., Czarnecki, W., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. (2017). Reinforcement learning with unsupervised auxiliary tasks. In *ICLR*.

Jaderberg, M., Simonyan, K., Zisserman, A., and Kavukcuoglu, K. (2015). Spatial transformer networks. In *NIPS*.

Jaksch, T., Ortner, R., and Auer, P. (2010). Near-optimal regret bounds for reinforcement learning. *JMLR*, 11:1563–1600.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R*. Springer.

Jan Peters, Katharina Mülling, Y. A. (2010). Relative entropy policy search. In *AAAI*.

Jaques, N., Gu, S., Bahdanau, D., Hernández-Lobato, J. M., Turner, R. E., and Eck, D. (2017). Sequence tutor: Conservative fine-tuning of sequence generation models with KL-control. In *ICML*.

Jayaraman, D. and Grauman, K. (2018). Learning to look around: Intelligently exploring unseen environments for unknown tasks. In *CVPR*.

Jiang, N., Krishnamurthy, A., Agarwal, A., Langford, J., and Schapire, R. E. (2017). Contextual decision processes with low bellman rank are pac-learnable. In *ICML*.

Jiang, N. and Li, L. (2016). Doubly robust off-policy value evaluation for reinforcement learning. In *ICML*.

Jie, Z., Liang, X., Feng, J., Jin, X., Lu, W. F., and Yan, S. (2016). Tree-structured reinforcement learning for sequential object localization. In *NIPS*.

Jin, C., Allen-Zhu, Z., Bubeck, S., and Jordan, M. (2018). Is Q-learning provably efficient? In *NIPS*.

Jin, H., Song, Q., and Hu, X. (2018). Efficient Neural Architecture Search with Network Morphism. *ArXiv*.

Johansson, F. D., Shalit, U., and Sontag, D. (2016). Learning representations for counterfactual inference. In *ICML*.

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F., Wattenberg, M., Corrado, G., Hughes, M., and Dean, J. (2017). Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Jordan, M. (2018). Artificial intelligence???the revolution hasn?t happened yet. `https://medium.com/@mijordan3/artificial-intelligence-the-revolution-hasnt-happened-yet-5e1d5812e1e7`.

Jordan, M. I. and Mitchell, T. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2017). Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Jun, K.-S., Li, L., Ma, Y., and Zhu, X. (2018). Adversarial attacks on stochastic bandits. In *NIPS*.

Jurafsky, D. and Martin, J. H. (2017). *Speech and Language Processing (3rd ed. draft)*. Prentice Hall.

Justesen, N., Bontrager, P., Togelius, J., and Risi, S. (2017). Deep Learning for Video Game Playing. *ArXiv*.

Kadlec, R., Schmid, M., Bajgar, O., and Kleindienst, J. (2016). Text understanding with the attention sum reader network. In *ACL*.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134.

Kahneman, D. (2011). *Thinking, Fast and Slow*. Farrar, Straus and Giroux.

Kaiser, L. and Bengio, S. (2016). Can active memory replace attention? In *NIPS*.

Kaiser, L., Gomez, A. N., Shazeer, N., Vaswani, A., Parmar, N., Jones, L., and Uszkoreit, J. (2017a). One Model To Learn Them All. *ArXiv*.

Kaiser, Ł., Nachum, O., Roy, A., and Bengio, S. (2017b). Learning to Remember Rare Events. In *ICLR*.

Kakade, S. (2002). A natural policy gradient. In *NIPS*.

Kakade, S. and Langford, J. (2002). Approximately optimal approximate reinforcement learning. In *ICML*.

Kalchbrenner, N. and Blunsom, P. (2013). Recurrent continuous translation models. In *EMNLP*.

Kallus, N. and Zhou, A. (2018). Confounding-robust policy improvement. In *NIPS*.

Kandasamy, K., Bachrach, Y., Tomioka, R., Tarlow, D., and Carter, D. (2017). Batch policy gradient methods for improving neural conversation models. In *ICLR*.

Kandasamy, K., Neiswanger, W., Schneider, J., Poczos, B., and Xing, E. (2018). Neural architecture search with Bayesian optimisation and optimal transport. In *NIPS*.

Kansky, K., Silver, T., Mély, D. A., Eldawy, M., Lázaro-Gredilla, M., Lou, X., Dorfman, N., Sidor, S., Phoenix, S., and George, D. (2017). Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. In *ICML*.

Kantarcioglu, M. and Xi, B. (2016). Adversarial data mining: Big data meets cyber security. `https://www.sigsac.org/ccs/CCS2016/tutorials/index.html`. ACM Conference on Computer and Communications Security (CCS 2016) Tutorial.

Kavosh and Littman, M. L. (2017). A new softmax operator for reinforcement learning. In *ICML*.

Kearns, M. and Singh, S. (2002). Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49:209–232.

Keramati, R., Whang, J., Cho, P., and Brunskill, E. (2018). Strategic Object Oriented Reinforcement Learning. *ArXiv*.

Khadka, S. and Tumer, K. (2018). Evolutionary reinforcement learning. In *NIPS*.

Khandani, A. E., Kim, A. J., and Lo, A. W. (2010). Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance*, 34:2767–2787.

Khetarpal, K., Ahmed, Z., Cianflone, A., Islam, R., and Pineau, J. (2018). Re-evaluate: Reproducibility in evaluating reinforcement learning algorithms. In *Reproducibility in Machine Learning Workshop at ICML*.

Killian, T., Daulton, S., Konidaris, G., and Doshi-Velez, F. (2017). Robust and efficient transfer learning with hidden-parameter markov decision processes. In *NIPS*.

Kim, B., Farahmand, A.-m., Pineau, J., and Precup, D. (2014). Learning from limited demonstrations. In *NIPS*.

Kingma, D. P., Rezende, D. J., Mohamed, S., and Welling, M. (2014). Semi-supervised learning with deep generative models. In *NIPS*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *PNAS*, 114(13):3521–3526.

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. M. (2017). OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv*.

Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*, 32(11):1238–1278.

Koch, G., Zemel, R., and Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. In *ICML*.

Kocsis, L. and Szepesvári, C. (2006). Bandit based monte-carlo planning. In *Proceedings of the European conference on Machine Learning (ECML)*.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Kolter, J. Z. and Ng, A. Y. (2009). Near-bayesian exploration in polynomial time. In *ICML*.

Kompella, V. R., Stollenga, M., Luciw, M., and Schmidhuber, J. (2017). Continual curiosity-driven skill acquisition from high-dimensional video inputs for humanoid robots. *Artificial Intelligence*, 247:313–335.

Konda, V. R. and Tsitsiklis, J. N. (2003). On actor-critic algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166.

Kong, X., Xin, B., Wang, Y., and Hua, G. (2017). Collaborative deep reinforcement learning for joint object search. In *CVPR*.

Kornblith, S., Shlens, J., and Le, Q. V. (2018). Do Better ImageNet Models Transfer Better? *ArXiv*.

Kosorok, M. R. and Moodie, E. E. M. (2015). *Adaptive Treatment Strategies in Practice: Planning Trials and Analyzing Data for Personalized Medicine*. ASA-SIAM Series on Statistics and Applied Probability.

Kottur, S., Moura, J. M., Lee, S., and Batra, D. (2017). Natural language does not emerge 'naturally' in multi-agent dialog. In *EMNLP*.

Krakovsky, M. (2016). Reinforcement renaissance. *Communications of the ACM*, 59(8):12–14.

Kraska, T., Beutel, A., Chi, E. H., Dean, J., and Polyzotis, N. (2018). The case for learned index structures. In *International Conference on Management of Data (ACM SIGMOD)*.

Kriegeskorte, N. and Douglas, P. K. (2018). Cognitive computational neuroscience. *Nature Neuroscience*.

Krishnan, S., Yang, Z., Goldberg, K., Hellerstein, J., and Stoica, I. (2018). Learning to Optimize Join Queries With Deep Reinforcement Learning. *ArXiv e-prints*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NIPS*.

Krull, A., Brachmann, E., Nowozin, S., Michel, F., Shotton, J., and Rother, C. (2017). Poseagent: Budget-constrained 6d object pose estimation via reinforcement learning. In *CVPR*.

Kuhn, M. and Johnson, K. (2013). *Applied Predictive Modeling*. Springer.

Kulkarni, T. D., Narasimhan, K. R., Saeedi, A., and Tenenbaum, J. B. (2016). Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *NIPS*.

Kulkarni, T. D., Saeedi, A., Gautam, S., and Gershman, S. J. (2016). Deep Successor Reinforcement Learning. *ArXiv*.

Kulkarni, T. D., Whitney, W., Kohli, P., and Tenenbaum, J. B. (2015). Deep convolutional inverse graphics network. In *NIPS*.

Kumaraswamy, R., Schlegel, M., White, A., and White, M. (2018). Context-dependent upper-confidence bounds for directed exploration. In *NIPS*.

Kurach, K., Lucic, M., Zhai, X., Michalski, M., and Gelly, S. (2018). The GAN Landscape: Losses, Architectures, Regularization, and Normalization. *ArXiv*.

Kurzweil, R. (1992). *The Age of Intelligent Machines*. The MIT Press.

Lage, I., Ross, A., Gershman, S. J., Kim, B., and Doshi-Velez, F. (2018). Human-in-the-loop interpretability prior. In *NIPS*.

Lagoudakis, M. G. and Parr, R. (2003). Least-squares policy iteration. *JMLR*, 4:1107 – 1149.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2016). Building machines that learn and think like people. *Behavioral and Brain Sciences*, 24:1–101.

Lample, G. and Chaplot, D. S. (2017). Playing FPS games with deep reinforcement learning. In *AAAI*.

Lanctot, M., Srinivasan, S., Zambaldi, V., Perolat, J., karl Tuyls, Munos, R., and Bowling, M. (2018). Actor-critic policy optimization in partially observable multiagent environments. In *NIPS*.

Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Tuyls, K., Perolat, J., Silver, D., and Graepel, T. (2017). A unified game-theoretic approach to multiagent reinforcement learning. In *NIPS*.

Langford, J. and Zhang, T. (2007). The epoch-greedy algorithm for multi-armed bandits with side information. In *NIPS*.

Lattimore, F., Lattimore, T., and Reid, M. D. (2016). Causal bandits: Learning good interventions via causal inference. In *NIPS*.

Lattimore, T., Kveton, B., Li, S., and Szepesvári, C. (2018). Toprank: A practical algorithm for online stochastic ranking. In *NIPS*.

Lattimore, T. and Szepesvári, C. (2018). *Bandit Algorithms*. Cambridge University Press.

Lazic, N., Boutilier, C., Lu, T., Wong, E., Roy, B., Ryu, M., and Imwalle, G. (2018). Data center cooling using model-predictive control. In *NIPS*.

Le, H. M., Jiang, N., Agarwal, A., Dudík, M., Yue, Y., and Daumé, III, H. (2018). Hierarchical imitation and reinforcement learning. In *ICML*.

Le, Q. V., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G. S., Dean, J., and Ng, A. Y. (2012). Building high-level features using large scale unsupervised learning. In *ICML*.

LeCun, Y. (2018). Learning world models: The next step towards AI. `https://www.youtube.com/watch?v=IK_svPHKA0U`. IJCAI-ECAI 2018 Invited Talk.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521:436–444.

LeCun, Y. and Manning, C. (2018). What innate priors should we build into the architecture of deep learning systems? `https://www.youtube.com/watch?v=fKk9KhGRBdI`.

LeCun, Y. and Marcus, G. (2017). Does AI need more innate machinery? `https://www.youtube.com/watch?v=aCCotxqxFsk`.

Lee, A. X., Levine, S., and Abbeel, P. (2017). Learning visual servoing with deep features and trust region fitted Q-iteration. In *ICLR*.

Lee, K.-F. (2018). *AI Superpowers: China, Silicon Valley, and the New World Order*. Houghton Mifflin Harcourt.

Lee, L., Parisotto, E., Chaplot, D. S., Xing, E., and Salakhutdinov, R. (2018). Gated path planning networks. In *ICML*.

Lehman, J., Chen, J., Clune, J., and Stanley, K. O. (2017). Safe Mutations for Deep and Recurrent Neural Networks through Output Gradients. *ArXiv*.

Leibo, J. Z., Zambaldi, V., Lanctot, M., Marecki, J., and Graepel, T. (2017). Multi-agent reinforcement learning in sequential social dilemmas. In *AAMAS*.

Leike, J., Ibarz, B., Amodei, D., Irving, G., and Legg, S. (2018). Reward learning from human preferences and demonstrations in Atari. In *NIPS*.

Lenz, I., Knepper, R., and Saxena, A. (2015). DeepMPC: Learning deep latent features for model predictive control. In *RSS*.

Leonetti, M., Iocchi, L., and Stone, P. (2016). A synthesis of automated planning and reinforcement learning for efficient, robust decision-making. *Artificial Intelligence*, 241:103–130.

Levine, S. (2018). CS 294: Deep reinforcement learning. `http://rail.eecs.berkeley.edu/deeprlcourse/`.

Levine, S. (2018). Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review. *ArXiv*.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *JMLR*, 17:1–40.

Levine, S. and Koltun, V. (2014). Learning complex neural network policies with trajectory optimization. In *ICML*.

Lewis, M., Yarats, D., Dauphin, Y. N., Parikh, D., and Batra, D. (2017). Deal or no deal? end-to-end learning for negotiation dialogues. In *EMNLP*.

Leyton-Brown, K. and Shoham, Y. (2008). *Essentials of Game Theory: A Concise, Multidisciplinary Introduction*. Morgan & Claypool.

Li, J., Miller, A. H., Chopra, S., Ranzato, M., and Weston, J. (2017a). Dialogue learning with human-in-the-loop. In *ICLR*.

Li, J., Miller, A. H., Chopra, S., Ranzato, M., and Weston, J. (2017b). Learning through dialogue interactions by asking questions. In *ICLR*.

Li, J., Monroe, W., Ritter, A., Galley, M., Gao, J., and Jurafsky, D. (2016). Deep reinforcement learning for dialogue generation. In *EMNLP*.

Li, K. and Malik, J. (2017). Learning to optimize. In *ICLR*.

Li, K. and Malik, J. (2017). Learning to Optimize Neural Nets. *ArXiv*.

Li, L. (2012). Sample complexity bounds of exploration. In Wiering, M. and van Otterlo, M., editors, *Reinforcement Learning: State-of-the-Art*, pages 175–204. Springer-Verlag Berlin Heidelberg.

Li, L., Chu, W., Langford, J., and Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *WWW*.

Li, L., Littman, M. L., Walsh, T. J., and Strehl, A. L. (2011). Knows what it knows: a framework for self-aware learning. *Machine Learning*, 82(3):399–443.

Li, M. and Vitányi, P. (2008). *An Introduction to Kolmogorov Complexity and Its Applications (3rd edition)*. Springer.

Li, S., Xiao, S., Zhu, S., Du, N., Xie, Y., and Song, L. (2018a). Learning temporal point processes via reinforcement learning. In *NIPS*.

Li, X., Chen, Y.-N., Li, L., and Gao, J. (2017). End-to-End Task-Completion Neural Dialogue Systems. *ArXiv*.

Li, Y. (2017). Deep Reinforcement Learning: An Overview. *ArXiv*.

Li, Y. (2018). About AI conferences. https://medium.com/@yuxili/about-ai-conferences-bbd8fbf84290.

Li, Y., Liang, X., Hu, Z., and Xing, E. (2018b). Hybrid retrieval-generation reinforced agent for medical image report generation. In *NIPS*.

Li, Y., Song, J., and Ermon, S. (2017). InfoGAIL: Interpretable imitation learning from visual demonstrations. In *NIPS*.

Li, Y., Szepesvári, C., and Schuurmans, D. (2009). Learning exercise policies for American options. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Liang, C., Berant, J., Le, Q., Forbus, K. D., and Lao, N. (2017a). Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *ACL*.

Liang, C., Berant, J., Le, Q., Forbus, K. D., and Lao, N. (2017b). Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *ACL*.

Liang, C., Norouzi, M., Berant, J., Le, Q. V., and Lao, N. (2018). Memory augmented policy optimization for program synthesis with generalization. In *NIPS*.

Liang, X., Lee, L., and Xing, E. P. (2017c). Deep variation-structured reinforcement learning for visual relationship and attribute detection. In *CVPR*.

Liang, Y., Machado, M. C., Talvitie, E., and Bowling, M. (2016). State of the art control of atari games using shallow reinforcement learning. In *AAMAS*.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. In *ICLR*.

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3):293–321.

Lin, Z., Gehring, J., Khalidov, V., and Synnaeve, G. (2017). Stardata: A starcraft ai research dataset. In *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*.

Ling, Y., Hasan, S. A., Datla, V., Qadir, A., Lee, K., Liu, J., and Farri, O. (2017). Diagnostic inferencing via improving clinical concept extraction with deep reinforcement learning: A preliminary study. In *Machine Learning for Healthcare*.

Lipton, Z., Li, X., Gao, J., Li, L., Ahmed, F., and Deng, L. (2018). BBQ-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. In *AAAI*.

Lipton, Z. C. (2018). The mythos of model interpretability. *ACM Queue*, 16(3).

Lipton, Z. C. and Steinhardt, J. (2018). Troubling trends in machine learning scholarship. In *ICML 2018 Machine Learning Debates Workshop*.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning? In *ICML*.

Littman, M. L. (2015). Reinforcement learning improves behaviour from evaluative feedback. *Nature*, 521:445–451.

Littman, M. L., Sutton, R. S., and Singh, S. (2001). Predictive representations of state. In *NIPS*.

Liu, B. (2012). *Sentiment Analysis and Opinion Mining*. Morgan & Claypool.

Liu, B., Gemp, I., Ghamvamzadeh, M., Liu, J., Mahadevan, S., and Petrik, M. (2018a). Proximal gradient temporal difference learning algorithms. *JAIR*.

Liu, C. and Tomizuka, M. (2016). Algorithmic safety measures for intelligent industrial co-robots. In *ICRA*.

Liu, C. and Tomizuka, M. (2017). Designing the robot behavior for safe human robot interactions. In Wang, Y. and Zhang, F., editors, *Trends in Control and Decision-Making for Human-Robot Collaboration Systems*. Springer.

Liu, C., Zoph, B., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J., and Murphy, K. (2017). Progressive Neural Architecture Search. *ArXiv*.

Liu, F., Li, S., Zhang, L., Zhou, C., Ye, R., Wang, Y., and Lu, J. (2017). 3DCNN-DQN-RNN: A deep reinforcement learning framework for semantic parsing of large-scale 3d point clouds. In *ICCV*.

Liu, H., Simonyan, K., Vinyals, O., Fernando, C., and Kavukcuoglu, K. (2017). Hierarchical Representations for Efficient Architecture Search. *ArXiv*.

Liu, N., Li, Z., Xu, Z., Xu, J., Lin, S., Qiu, Q., Tang, J., and Wang, Y. (2017a). A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning. In *37th IEEE International Conference on Distributed Computing (ICDCS 2017)*.

Liu, Q., Li, L., Tang, Z., and Zhou, D. (2018b). Breaking the curse of horizon: Infinite-horizon off-policy estimation. In *NIPS*.

Liu, S., Long, M., Wang, J., and Jordan, M. (2018c). Generalized zero-shot learning with deep calibration network. In *NIPS*.

Liu, S., Zhu, Z., Ye, N., Guadarrama, S., and Murphy, K. (2016). Improved Image Captioning via Policy Gradient optimization of SPIDEr. *ArXiv*.

Liu, Y., Chen, J., and Deng, L. (2017b). Unsupervised sequence classification using sequential output statistics. In *NIPS*.

Liu, Y., Gottesman, O., Raghu, A., Komorowski, M., Faisal, A. A., Doshi-Velez, F., and Brunskill, E. (2018d). Representation balancing MDPs for off-policy policy evaluation. In *NIPS*.

Liu, Y. and Sun, J. (2017). Deep learning for health care applications: Challenges and solutions. `https://sites.google.com/view/icml2017-deep-health-tutorial/home`. ICML 2017 Tutorial.

Liu, Y.-E., Mandel, T., Brunskill, E., and Popović, Z. (2014). Trading off scientific knowledge and user learning with multi-armed bandits. In *Educational Data Mining (EDM)*.

Lo, A. W. (2004). The Adaptive Markets Hypothesis: Market efficiency from an evolutionary perspective. *Journal of Portfolio Management*, 30:15–29.

Lo, A. W., Mamaysky, H., and Wang, J. (2000). Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. *Journal of Finance*, 55(4):1705–1765.

Long, M., Cao, Y., Wang, J., and Jordan, M. I. (2015). Learning transferable features with deep adaptation networks. In *ICML*.

Long, M., Cao, Z., Wang, J., and Yu, P. S. (2017). Learning multiple tasks with multilinear relationship networks. In *NIPS*.

Long, M., Zhu, H., Wang, J., and Jordan, M. I. (2016). Unsupervised domain adaptation with residual transfer networks. In *NIPS*.

Longstaff, F. A. and Schwartz, E. S. (2001). Valuing American options by simulation: a simple least-squares approach. *The Review of Financial Studies*, 14(1):113–147.

Loos, S., Irving, G., Szegedy, C., and Kaliszyk, C. (2017). Deep Network Guided Proof Search. *ArXiv*.

Lopez-Paz, D., Nishihara, R., Chintala, S., Schölkopf, B., and Bottou, L. (2017). Discovering causal signals in images. In *CVPR*.

Lopez-Paz, D. and Ranzato, M. (2017). Gradient Episodic Memory for Continuum Learning. *ArXiv*.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. In *NIPS*.

Lu, J., Xiong, C., Parikh, D., and Socher, R. (2016). Knowing When to Look: Adaptive Attention via A Visual Sentinel for Image Captioning. *ArXiv*.

Lu, T., Boutilier, C., and Schuurmans, D. (2018). Non-delusional Q-learning and value-iteration. In *NIPS*.

Lucic, M., Kurach, K., Michalski, M., Gelly, S., and Bousquet, O. (2018). Are GANs created equal? a large-scale study. In *NIPS*.

Luenberger, D. G. (1997). *Investment Science*. Oxford University Press.

Luo, R., Tian, F., Qin, T., Chen, E., and Liu, T. (2018). Neural architecture optimization. In *NIPS*.

Machado, M. C., Bellemare, M. G., and Bowling, M. (2017). A Laplacian framework for option discovery in reinforcement learning. In *ICML*.

Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M., and Bowling, M. (2017). Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents. *ArXiv*.

Madhavan, V., Such, F. P., Clune, J., Stanley, K., and Lehman, J. (2018). Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *NIPS*.

Mahadevan, S. (2018a). Imagination machines. `https://people.cs.umass.edu/~mahadeva/IJCAI_2018_Tutorial/`. IJCAI 2018 Tutorial.

Mahadevan, S. (2018b). Imagination machines: A new challenge for artificial intelligence. In *AAAI*.

Mahadevan, S. and Maggioni, M. (2007). Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *JMLR*, 8:2169–2231.

Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., and van der Maaten, L. (2018). Exploring the Limits of Weakly Supervised Pretraining. *ArXiv*.

Mahmood, A. R., van Hasselt, H., and Sutton, R. S. (2014). Weighted importance sampling for off-policy learning with linear function approximation. In *NIPS*.

Mahmud, M., Kaiser, M. S., Hussain, A., and Vassanelli, S. (2018). Applications of deep learning and reinforcement learning to biological data. *IEEE Transactions on Neural Networks and Learning Systems*, 29(6):2063–2079.

Malik, J. (2018). IJCAI Research Excellence Award talk. `http://goo.gl/MqxYiU`.

Malinowski, M., Doersch, C., Santoro, A., and Battaglia, P. (2018). Learning Visual Question Answering by Bootstrapping Hard Attention. *ArXiv*.

Mandel, T., Liu, Y. E., Levine, S., Brunskill, E., and Popović, Z. (2014). Offline policy evaluation across representations with applications to educational games. In *AAMAS*.

Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., and Raedt, L. D. (2018). DeepProbLog: Neural probabilistic logic programming. In *NIPS*.

Manning, C. D. (2017). Last words: Computational linguistics and deep learning, a look at the importance of natural language processing. `https://bit.ly/2wtuemM`.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Mannion, P., Duggan, J., and Howley, E. (2016). An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In McCluskey, T., Kotsialos, A., Müller, J., Klügl, F., Rana, O., and R., S., editors, *Autonomic Road Transport Support Systems*, pages 47–66. Springer.

Mao, H., Alizadeh, M., Menache, I., and Kandula, S. (2016). Resource management with deep reinforcement learning. In *ACM Workshop on Hot Topics in Networks (HotNets)*.

Mao, X., Li, Q., Xie, H., Lau, R. Y. K., and Wang, Z. (2016). Least Squares Generative Adversarial Networks. *ArXiv*.

Marcus, G. (2018). Deep Learning: A Critical Appraisal. *ArXiv*.

Mathe, S., Pirinen, A., and Sminchisescu, C. (2016). Reinforcement learning for visual object detection. In *CVPR*.

Maurer, A., Pontil, M., and Romera-Paredes, B. (2016). The benefit of multitask representation learning. *JMLR*, 17(81):1–32.

McAleer, S., Agostinelli, F., Shmakov, A., and Baldi, P. (2018). Solving the Rubik's Cube Without Human Knowledge. *ArXiv*.

McAllister, R. and Rasmussen, C. E. (2017). Data-efficient reinforcement learning in continuous-state POMDPs. In *NIPS*.

McCann, B., Bradbury, J., Xiong, C., and Socher, R. (2017). Learned in Translation: Contextualized Word Vectors. *ArXiv*.

McCann, B., Shirish Keskar, N., Xiong, C., and Socher, R. (2018). The Natural Language Decathlon: Multitask Learning as Question Answering. *ArXiv*.

Melis, D. A. and Jaakkola, T. (2018). Towards robust interpretability with self-explaining neural networks. In *NIPS*.

Melis, G., Dyer, C., and Blunsom, P. (2018). On the state of the art of evaluation in neural language models. In *ICLR*.

Merel, J., Tassa, Y., TB, D., Srinivasan, S., Lemmon, J., Wang, Z., Wayne, G., and Heess, N. (2017). Learning human behaviors from motion capture by adversarial imitation. *ArXiv*.

Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., He, X., Heck, L., Tur, G., Hakkani-Tür, D., Yu, D., and Zweig, G. (2015). Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Mestres, A., Rodriguez-Natal, A., Carner, J., Barlet-Ros, P., Alarcón, E., Solé, M., Muntés, V., Meyer, D., Barkai, S., Hibbett, M. J., Estrada, G., Maŕuf, K., Coras, F., Ermagan, V., Latapie, H., Cassar, C., Evans, J., Maino, F., Walrand, J., and Cabellos, A. (2017). Knowledge-defined networking. *ACM SIGCOMM Computer Communication Review*, 47(3):2–10.

Mhamdi, E. M. E., Guerraoui, R., Hendrikx, H., and Maurer, A. (2017). Dynamic safe interruptibility for decentralized multi-agent reinforcement learning. In *NIPS*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. In *ICLR*.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., and Joulin, A. (2017). Advances in Pre-Training Distributed Word Representations. *ArXiv*.

Miller, J. and Hardt, M. (2018). When Recurrent Models Don't Need To Be Recurrent. *ArXiv*.

Miotto, R., Wang, F., Wang, S., Jiang, X., and Dudley, J. T. (2017). Deep learning for healthcare: review, opportunities and challenges. *Briefings in Bioinformatics*, pages 1–11.

Mirhoseini, A., Pham, H., Le, Q. V., Steiner, B., Larsen, R., Zhou, Y., Kumar, N., and Mohammad Norouzi, Samy Bengio, J. D. (2017). Device placement optimization with reinforcement learning. In *ICML*.

Mirowski, P., Grimes, M., Malinowski, M., Hermann, K. M., Anderson, K., Teplyashin, D., Simonyan, K., koray kavukcuoglu, Zisserman, A., and Hadsell, R. (2018). Learning to navigate in cities without a map. In *NIPS*.

Mirowski, P., Pascanu, R., Viola, F., Soyer, H., Ballard, A., Banino, A., Denil, M., Goroshin, R., Sifre, L., Kavukcuoglu, K., Kumaran, D., and Hadsell, R. (2017). Learning to navigate in complex environments. In *ICLR*.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. (2018). A simple neural attentive meta-learner. In *ICLR*.

Mitchell, T. (1997). *Machine Learning*. McGraw Hill.

Mitchell, T. and Brynjolfsson, E. (2017). Track how technology is transforming work. *Nature*, 544:290–292.

Mitra, B. and Craswell, N. (2017). Neural Models for Information Retrieval. *ArXiv*.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Harley, T., Lillicrap, T. P., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *ICML*.

Mnih, V., Heess, N., Graves, A., and Kavukcuoglu, K. (2014). Recurrent models of visual attention. In *NIPS*.

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. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Mo, K., Li, S., Zhang, Y., Li, J., and Yang, Q. (2018). Personalizing a dialogue system with transfer learning. In *AAAI*.

Monroe, D. (2017). Deep learning takes on translation. *Communications of the ACM*, 60(6):12–14.

Moody, J. and Saffell, M. (2001). Learning to trade via direct reinforcement. *TNN*, 12(4):875–889.

Moore, A. and Atkeson, C. (1993). Prioritized sweeping: Reinforcement learning with less data and less time. *MLJ*, 13(1):103–130.

Moravčík, M., Schmid, M., Burch, N., Lisý, V., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M., and Bowling, M. (2017). Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513.

Morimura, T., Sugiyama, M., Kashima, H., Hachiya, H., and Tanaka, T. (2010a). Nonparametric return distribution approximation for reinforcement learning. In *ICML*.

Morimura, T., Sugiyama, M., Kashima, H., Hachiya, H., and Tanaka, T. (2010b). Parametric return density estimation for reinforcement learning. In *the Conference on Uncertainty in Artificial Intelligence (UAI)*.

Mrowca, D., Zhuang, C., Wang, E., Haber, N., Fei-Fei, L., Tenenbaum, J., and Yamins, D. (2018). A flexible neural representation for physics prediction. In *NIPS*.

Mukwevho, M. A. and Celik, T. (2018). Toward a smart cloud: A review of fault-tolerance methods in cloud systems. *IEEE Transactions on Services Computing*.

Mullainathan, S. (2017). Machine learning and prediction in economics and finance. `https://www.youtube.com/watch?v=xl3yQBhI6vY`. American Finance Association (AFA) Lecture.

Müller, M. (2002). Computer go. *Artificial Intelligence*, 134(1-2):145–179.

Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. G. (2016). Safe and efficient off-policy reinforcement learning. In *NIPS*.

Murphy, J. J. (1999). *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications*. Prentice Hall Press.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Nachum, O., Gu, S., Lee, H., and Levine, S. (2018). Data-efficient hierarchical reinforcement learning. In *NIPS*.

Nachum, O., Norouzi, M., and Schuurmans, D. (2017). Improving policy gradient by exploring underappreciated rewards. In *ICLR*.

Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. (2017). Bridging the gap between value and policy based reinforcement learning. In *NIPS*.

Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. (2018). Bridging the gap between value and policy based reinforcement learning. In *ICLR*.

Nair, A., Pong, V., Dalal, M., Bahl, S., Lin, S., and Levine, S. (2018). Visual goal-conditioned reinforcement learning by representation learning. In *NIPS*.

Nair, A., Srinivasan, P., Blackwell, S., Alcicek, C., Fearon, R., De Maria, A., Panneershelvam, V., Suleyman, M., Beattie, C., Petersen, S., Legg, S., Mnih, V., Kavukcuoglu, K., and Silver, D. (2015). Massively parallel methods for deep reinforcement learning. In *ICML 2015 Deep Learning Workshop*.

Narasimhan, K., Kulkarni, T., and Barzilay, R. (2015). Language understanding for text-based games using deep reinforcement learning. In *EMNLP*.

Narasimhan, K., Yala, A., and Barzilay, R. (2016). Improving information extraction by acquiring external evidence with reinforcement learning. In *EMNLP*.

Nazari, M., Oroojlooy, A., Snyder, L., and Takáč, M. (2018). Reinforcement learning for solving the vehicle routing problem. In *NIPS*.

Nedić, A. and Bertsekas, D. P. (2003). Least squares policy evaluation algorithms with linear function approximation. *Discrete Event Dynamic Systems: Theory and Applications*, 13:79–110.

Negrinho, R., Gormley, M., and Gordon, G. (2018). Learning beam search policies via imitation learning. In *NIPS*.

Neubig, G. (2017). Neural Machine Translation and Sequence-to-sequence Models: A Tutorial. *ArXiv*.

Neuneier, R. (1997). Enhancing Q-learning for optimal asset allocation. In *NIPS*.

Ng, A. (2016a). Hiring your first chief ai officer. . Harvard Business Review.

Ng, A. (2016b). Nuts and bolts of building applications using deep learning. https://www.youtube.com/watch?v=F1ka6a13S9I.

Ng, A. (2016c). What artificial intelligence can and can't do right now. https://hbr.org/2016/11/what-artificial-intelligence-can-and-cant-do-right-now. Harvard Business Review.

Ng, A. (2018). *Machine Learning Yearning (draft)*. deeplearning.ai.

Ng, A. and Russell, S. (2000). Algorithms for inverse reinforcement learning. In *ICML*.

Ng, A. Y., Harada, D., and Russell, S. J. (2000). Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*.

Ng, A. Y., Kim, H. J., Jordan, M. I., and Sastry, S. (2004). Autonomous helicopter flight via reinforcement learning. In *NIPS*.

Nilsson, N. J. (1998). *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann.

Nogueira, R. and Cho, K. (2016). End-to-End Goal-Driven Web Navigation. *ArXiv*.

Nogueira, R. and Cho, K. (2017). Task-oriented query reformulation with reinforcement learning. In *EMNLP*.

Northcutt, C. G. (2017). Artificial intelligence in online education: With an emphasis on personalization. http://curtisnorthcutt.com/resources/pdf/northcutt_mit_2017_ai_in_online_education.pdf.

O'Donoghue, B., Munos, R., Kavukcuoglu, K., and Mnih, V. (2017). Combining policy gradient and Q-learning. In *ICLR*.

Oh, J., Chockalingam, V., Singh, S., and Lee, H. (2016). Control of memory, active perception, and action in minecraft. In *ICML*.

Oh, J., Guo, X., Lee, H., Lewis, R., and Singh, S. (2015). Action-conditional video prediction using deep networks in atari games. In *NIPS*.

Oh, J., Singh, S., and Lee, H. (2017). Value prediction network. In *NIPS*.

O'Kelly, M., Sinha, A., Namkoong, H., Tedrake, R., and Duchi, J. C. (2018). Scalable end-to-end autonomous vehicle testing via rare-event simulation. In *NIPS*.

Olah, C. and Carter, S. (2016). Attention and augmented recurrent neural networks. *Distill*. http://distill.pub/2016/augmented-rnns.

Olah, C., Mordvintsev, A., and Schubert, L. (2017). Feature visualization. *Distill*.

Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K., and Mordvintsev, A. (2018). The building blocks of interpretability. *Distill*. https://distill.pub/2018/building-blocks.

Oliehoek, F. A., Spaan, M. T. J., and Vlassis, N. (2008). Optimal and approximate q-value functions for decentralized pomdps. *JAIR*, 32(1):289–353.

Omidshafiei, S., Pazis, J., Amato, C., How, J. P., and Vian, J. (2017). Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *ICML*.

Ontañón, S., Synnaeve, G., Uriarte, A., Richoux, F., Churchill, D., and Preuss, M. (2013). A survey of real-time strategy game ai research and competition in starcraft. *IEEE Transactions on Computational Intelligence and AI in Games*, 5(4):293–311.

OpenAI (2018). Learning Dexterous In-Hand Manipulation. *ArXiv*.

Oquab, M., Bottou, L., Laptev, I., and Sivic, J. (2015). Is object localization for free? – weakly-supervised learning with convolutional neural networks. In *CVPR*.

Ormoneit, D. and Sen, Ś. (2002). Kernel-based reinforcement learning. *Machine Learning*, 49(2-3):161–178.

Osband, I., Aslanides, J. S., and Cassirer, A. (2018). Randomized prior functions for deep reinforcement learning. In *NIPS*.

Osband, I., Blundell, C., Pritzel, A., and Roy, B. V. (2016). Deep exploration via bootstrapped DQN. In *NIPS*.

Ostrovski, G., Bellemare, M. G., van den Oord, A., and Munos, R. (2017). Count-based exploration with neural density models. In *ICML*.

Oudeyer, P.-Y., Gottlieb, J., and Lopes, M. (2016). Intrinsic motivation, curiosity and learning: theory and applications in educational technologies. *Progress in brain research, Elsevier*, 229:257–284.

Oudeyer, P.-Y. and Kaplan, F. (2007). What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1(6).

Palm, R., Paquet, U., and Winther, O. (2018). Recurrent relational networks. In *NIPS*.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345 – 1359.

Pan, Y., Farahmand, A.-m., White, M., Nabi, S., Grover, P., and Nikovski, D. (2018). Reinforcement learning with function-valued action spaces for partial differential equation control. In *ICML*.

Pang, Z.-J., Liu, R.-Z., Meng, Z.-Y., Zhang, Y., Yu, Y., and Lu, T. (2018). On Reinforcement Learning for Full-length Game of StarCraft. *ArXiv e-prints*.

Papernot, N., Abadi, M., Erlingsson, Ú., Goodfellow, I., and Talwar, K. (2017). Semi-supervised knowledge transfer for deep learning from private training data. In *ICLR*.

Papernot, N., Faghri, F., Carlini, N., Goodfellow, I., Feinman, R., Kurakin, A., Xie, C., Sharma, Y., Brown, T., Roy, A., Matyasko, A., Behzadan, V., Hambardzumyan, K., Zhang, Z., Juang, Y.-L., Li, Z., Sheatsley, R., Garg, A., Uesato, J., Gierke, W., Dong, Y., Berthelot, D., Hendricks, P., Rauber, J., Long, R., and McDaniel, P. (2016). Technical Report on the CleverHans v2.1.0 Adversarial Examples Library. *ArXiv*.

Parisotto, E., Ba, J. L., and Salakhutdinov, R. (2016). Actor-mimic: Deep multitask and transfer reinforcement learning. In *ICLR*.

Parisotto, E., Mohamed, A.-r., Singh, R., Li, L., Zhou, D., and Kohli, P. (2017). Neuro-symbolic program synthesis. In *ICLR*.

Parkes, D. C. and Wellman, M. P. (2015). Economic reasoning and artificial intelligence. *Science*, 349(6245):267–272.

Parr, R. and Russell, S. (1998). Reinforcement learning with hierarchies of machines. In *NIPS*.

Pastor, L. and Stambaugh, R. F. (2009). Predictive systems: Living with imperfect predictors. *Journal of Finance*, (to appear).

Pasunuru, R. and Bansal, M. (2017). Reinforced video captioning with entailment rewards. In *EMNLP*.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *ICML*.

Paulus, R., Xiong, C., and Socher, R. (2017). A Deep Reinforced Model for Abstractive Summarization. *ArXiv*.

Pearl, J. (2009). *Causality*. Cambridge University Press.

Pearl, J. (2018). The seven pillars of causal reasoning with reflections on machine learning. *UCLA Technical Report R-481*.

Pearl, J., Glymour, M., and Jewell, N. P. (2016). *Causal Inference in Statistics: A Primer*. Wiley.

Pearl, J. and Mackenzie, D. (2018). *The Book of Why: The New Science of Cause and Effect*. Basic Books.

Peng, B., Li, X., Li, L., Gao, J., Celikyilmaz, A., Lee, S., and Wong, K.-F. (2017a). Composite task-completion dialogue system via hierarchical deep reinforcement learning. In *EMNLP*.

Peng, P., Wen, Y., Yang, Y., Yuan, Q., Tang, Z., Long, H., and Wang, J. (2017b). Multiagent Bidirectionally-Coordinated Nets: Emergence of Human-level Coordination in Learning to Play StarCraft Combat Games. *ArXiv*.

Peng, X. B., Abbeel, P., Levine, S., and van de Panne, M. (2018a). Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. In *SIGGRAPH*.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. (2017c). Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. *ArXiv*.

Peng, X. B., Kanazawa, A., Malik, J., Abbeel, P., and Levine, S. (2018b). Sfv: Reinforcement learning of physical skills from videos. *ACM Transaction on Graphics*, 37(6).

Peng, Y.-S., Tang, K., Lin, H.-T., and Chang, E. (2018c). Exploring sparse features in deep reinforcement learning towards fast disease diagnosis. In *NIPS*.

Pérez-D'Arpino, C. and Shah, J. A. (2017). C-learn: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy. In *ICRA*.

Perolat, J., Leibo, J. Z., Zambaldi, V., Beattie, C., Tuyls, K., and Graepel, T. (2017). A multi-agent reinforcement learning model of common-pool resource appropriation. In *NIPS*.

Peters, J., Janzing, D., and Schölkopf, B. (2017). *Elements of Causal Inference: Foundations and Learning Algorithms*. MIT Press.

Peters, J. and Neumann, G. (2015). Policy search: Methods and applications. `https://icml.cc/2015/tutorials/PolicySearch.pdf`. ICML 2015 Tutorial.

Peters, J. and Schaal, S. (2007). Reinforcement learning by reward-weighted regression for operational space control. In *ICML*.

Petroski Such, F., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O., and Clune, J. (2017). Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning. *ArXiv*.

Pfau, D. and Vinyals, O. (2016). Connecting Generative Adversarial Networks and Actor-Critic Methods. *ArXiv*.

Phua, C., Lee, V., Smith, K., and Gayler, R. (2010). A Comprehensive Survey of Data Mining-based Fraud Detection Research. *ArXiv*.

Platt, J. (2017). Powering the next 100 years. `http://goo.gl/aa7svZ`. NIPS 2017 Invited Talk.

Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B., and Liao, Q. (2017). Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. *International Journal of Automation and Computing*, 14(5):503–519.

Pong, V., Gu, S., Dalal, M., and Levine, S. (2018). Temporal difference models: Model-free deep rl for model-based control. In *ICLR*.

Poole, D., Mackworth, A., and Goebel, R. (1998). *Computational Intelligence: A Logical Approach*. Oxford University Press.

Popov, I., Heess, N., Lillicrap, T., Hafner, R., Barth-Maron, G., Vecerik, M., Lampe, T., Tassa, Y., Erez, T., and Riedmiller, M. (2017). Data-efficient Deep Reinforcement Learning for Dexterous Manipulation. *ArXiv*.

Popova, M., Isayev, O., and Tropsha, A. (2018). Deep reinforcement learning for de novo drug design. *Science Advances*, 4(7).

Powell, W. B. (2010). Merging AI and OR to solve high-dimensional stochastic optimization problems using approximate dynamic programming. *INFORMS Journal on Computing*, 22(1):2–17.

Powell, W. B. (2011). *Approximate Dynamic Programming: Solving the curses of dimensionality (2nd Edition)*. John Wiley and Sons.

Prashanth, L., Jie, C., Fu, M., Marcus, S., and Szepesári, C. (2016). Cumulative prospect theory meets reinforcement learning: Prediction and control. In *ICML*.

Precup, D. (2018). Temporal abstraction. `https://dlrlsummerschool.ca`. Deep Learning and Reinforcement Learning Summer School.

Precup, D., Sutton, R. S., and Dasgupta, S. (2001). Off-policy temporal difference learning with function approximation. In *ICML*.

Press, G. (2016). Artificial intelligence pioneers: Peter Norvig, Google. `http://goo.gl/Ly8gxQ`. Forbes.

Provost, F. and Fawcett, T. (2013). *Data Science for Business*. O'Reilly Media.

Rabinowitz, N. C., Perbet, F., Song, H. F., Zhang, C., Eslami, S. A., and Botvinick, M. (2018). Machine theory of mind. In *ICML*.

Raedt, L. D., Frasconi, P., Kersting, K., and Muggleton, S., editors (2008). *Probabilistic inductive logic programming: theory and applications*. Springer.

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *NIPS*.

Rahimi, A. and Recht, B. (2017a). An addendum to alchemy. `http://www.argmin.net/2017/12/11/alchemy-addendum/`.

Rahimi, A. and Recht, B. (2017b). Back when we were kids. `https://www.youtube.com/watch?v=Qi1Yry33TQE`. NIPS 2017 Test-of-time Award Talk.

Rahimi, A. and Recht, B. (2017c). Reflections on random kitchen sinks. `http://www.argmin.net/2017/12/05/kitchen-sinks/`.

Rajendran, J., Lakshminarayanan, A., Khapra, M. M., P, P., and Ravindran, B. (2017). Attend, adapt and transfer: Attentive deep architecture for adaptive transfer from multiple sources in the same domain. *ICLR*.

Rajkomar, A., Oren, E., Chen, K., Dai, A. M., Hajaj, N., Liu, P. J., Liu, X., Sun, M., Sundberg, P., Yee, H., Zhang, K., Duggan, G. E., Flores, G., Hardt, M., Irvine, J., Le, Q., Litsch, K., Marcus, J., Mossin, A., Tansuwan, J., Wang, D., Wexler, J., Wilson, J., Ludwig, D., Volchenboum, S. L., Chou, K., Pearson, M., Madabushi, S., Shah, N. H., Butte, A. J., Howell, M., Cui, C., Corrado, G., and Dean, J. (2018). Scalable and accurate deep learning for electronic health records. *Nature Digital Medicine*, 1(18).

Rajpurkar, P., Irvin, J., Bagul, A., Ding, D., Duan, T., Mehta, H., Yang, B., Zhu, K., Laird, D., Ball, R. L., Langlotz, C., Shpanskaya, K., Lungren, M. P., and Ng, A. Y. (2018). Mura: Large dataset for abnormality detection in musculoskeletal radiographs. In *Conference on Medical Imaging with Deep Learning*.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2016). Sequence level training with recurrent neural networks. In *ICLR*.

Rao, Y., Lu, J., and Zhou, J. (2017). Attention-aware deep reinforcement learning for video face recognition. In *ICCV*.

Rashid, T., Samvelyan, M., de Witt, C. S., Farquhar, G., Foerster, J., and Whiteson, S. (2018). QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *ICML*.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

Rawlik, K., Toussaint, M., and Vijayakumar, S. (2012). On stochastic optimal control and reinforcement learning by approximate inference. In *RSS*.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. (2018). Regularized Evolution for Image Classifier Architecture Search. *ArXiv*.

Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Tan, J., Le, Q., and Kurakin, A. (2017). Large-scale evolution of image classifiers. In *ICML*.

Reddy, G., Wong-Ng, J., Celani, A., Sejnowski, T. J., and Vergassola, M. (2018). Glider soaring via reinforcement learning in the field. *Nature*.

Reed, S. and de Freitas, N. (2016). Neural programmer-interpreters. In *ICLR*.

Reichstaller, A. and Knapp, A. (2017). Transferring context-dependent test inputs. In *IEEE International Conference on Software Quality, Reliability and Security (QRS)*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*.

Ren, Z., Wang, X., Zhang, N., Lv, X., and Li, L.-J. (2017). Deep reinforcement learning-based image captioning with embedding reward. In *CVPR*.

Rennie, S. J., Marcheret, E., Mroueh, Y., Ross, J., and Goel, V. (2017). Self-critical sequence training for image captioning. In *CVPR*.

Rhinehart, N., Kitani, K., and Vernaza, P. (2018). Inverse reinforcement learning for computer vision. `https://www.youtube.com/watch?v=JbNeLiNnvII`. CVPR 2018 Tutorial.

Rhinehart, N. and Kitani, K. M. (2017). First-person activity forecasting with online inverse reinforcement learning. In *ICCV*.

Rich, E. and Knight, K. (1991). *Artificial Intelligence*. McGraw-Hill.

Riedmiller, M. (2005). Neural fitted Q iteration - first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning (ECML)*.

Riemer, M., Liu, M., and Tesauro, G. (2018). Learning abstract options. In *NIPS*.

Rocktäschel, T. and Riedel, S. (2017). End-to-end Differentiable Proving. *ArXiv*.

Ross, S., Gordon, G. J., and Bagnell, J. A. (2010). A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Rossi, F., Beek, P. V., and Walsh, T., editors (2006). *Handbook of constraint programming*. Elsevier.

Rotmensch, M., Halpern, Y., Tlimat, A., Horng, S., and Sontag, D. (2017). Learning a health knowledge graph from electronic medical records. *Nature Scientific Reports*, 7(5994).

Rowland, M., Bellemare, M. G., Dabney, W., Munos, R., and Teh, Y. W. (2018). An analysis of categorical distributional reinforcement learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Ruder, S. (2017). An Overview of Multi-Task Learning in Deep Neural Networks. *ArXiv*.

Ruelens, F., Claessens, B. J., Vandael, S., Schutter, B. D., Babuška, R., and Belmans, R. (2016). Residential demand response of thermostatically controlled loads using batch reinforcement learning. *IEEE Transactions on Smart Grid*, PP(99):1–11.

Rummery, G. A. and Niranjan, M. (1994). *On-line Q-learning using connectionist sytems*. Technical Report CUED/F-INFENG-TR 166, Cambridge University, UK.

Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach (3rd edition)*. Pearson.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive neural networks. In *NIPS Deep Learning Symposium*.

Rusu, A. A., Vecerik, M., Rothörl, T., Heess, N., Pascanu, R., and Hadsell, R. (2017). Sim-to-real robot learning from pixels with progressive nets. In *Conference on Robot Learning (CoRL)*.

Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic routing between capsules. In *NIPS*.

Sadeghi, F., Toshev, A., Jang, E., and Levine, S. (2018). Sim2real view invariant visual servoing by recurrent control. In *CVPR*.

Salakhutdinov, R. (2016). Foundations of unsupervised deep learning, Deep Learning School, https://www.bayareadlschool.org. `https://www.youtube.com/watch?v=rK6bchqeaN8`.

Salimans, T., Ho, J., Chen, X., and Sutskever, I. (2017). Evolution Strategies as a Scalable Alternative to Reinforcement Learning. *ArXiv*.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. (2016). Meta-learning with memory-augmented neural networks. In *ICML*.

Santoro, A., Faulkner, R., Raposo, D., Rae, J., Chrzanowski, M., Weber, T., Wierstra, D., Vinyals, O., Pascanu, R., and Lillicrap, T. (2018). Relational recurrent neural networks. *ArXiv*.

Santoro, A., Raposo, D., Barrett, D. G. T., Malinowski, M., Pascanu, R., Battaglia, P., and Lillicrap, T. (2017). A simple neural network module for relational reasoning. In *NIPS*.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. (2018). How does batch normalization help optimization? (no, it is not about internal covariate shift). In *NIPS*.

Saon, G., Sercu, T., Rennie, S., and Kuo, H.-K. J. (2016). The IBM 2016 English Conversational Telephone Speech Recognition System. In *Annual Meeting of the International Speech Communication Association (INTERSPEECH)*.

Saria, S. (2014). A $3 trillion challenge to computational scientists: Transforming healthcare delivery. *IEEE Intelligent Systems*, 29(4):82–87.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Schaal, S. (2006). Dynamic movement primitives -a framework for motor control in humans and humanoid robotics. In Kimura, H., Tsuchiya, K., Ishiguro, A., and Witte, H., editors, *Adaptive Motion of Animals and Machines*. Springer, Tokyo.

Schaeffer, J. (1997). *One Jump Ahead*. Springer-Verlag, New York.

Schaeffer, J., Burch, N., Björnsson, Y., Kishimoto, A., Müller, M., Lake, R., Lu, P., and Sutphen, S. (2007). Checkers is solved. *Science*, 317:1518–1522.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. (2015). Universal value function approximators. In *ICML*.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2016). Prioritized experience replay. In *ICLR*.

Schmidhuber, J. (1987). *Evolutionary principles in self-referential learning*. Diploma thesis, Tech. Univ. Munich.

Schmidhuber, J. (1991). A possibility for implementing curiosity and boredom in model-building neural controllers. In *International Conference on Simulation of Adaptive Behavior on From Animals to Animats*, pages 222–227.

Schmidhuber, J. (2010). Formal theory of creativity, fun, and intrinsic motivation (1990-2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Schölkopf, B. and Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

Schulman, J. (2017). The nuts and bolts of deep reinforcement learning research. `https://www.youtube.com/watch?v=8EcdaCk9KaQ`.

Schulman, J., Abbeel, P., and Chen, X. (2017a). Equivalence Between Policy Gradients and Soft Q-Learning. *ArXiv*.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. (2015). Trust region policy optimization. In *ICML*.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2016). High-dimensional continuous control using generalized advantage estimation. In *ICLR*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017b). Proximal Policy Optimization Algorithms. *ArXiv*.

Sculley, D., Snoek, J., Wiltschko, A., and Rahimi, A. (2018). Winner's curse? on pace, progress, and empirical rigor. In *ICLR Workshop Track*.

Segler, M. H. S., Preuss, M., and Waller, M. P. (2018). Planning chemical syntheses with deep neural networks and symbolic AI. *Nature*, 555:604–610.

Sener, O., Sener, O., and Koltun, V. (2018). Multi-task learning as multi-objective optimization. In *NIPS*.

Serban, I. V., Lowe, R., Charlin, L., and Pineau, J. (2018). A survey of available corpora for building data-driven dialogue systems: The journal version. *Dialogue & Discourse*, 9(1):1–49.

Sharma, S., Lakshminarayanan, A. S., and Ravindran, B. (2017). Learning to repeat: Fine grained action repetition for deep reinforcement learning. In *ICLR*.

She, L. and Chai, J. (2017). Interactive learning for acquisition of grounded verb semantics towards human-robot communication. In *ACL*.

Shen, Y., Huang, P.-S., Gao, J., and Chen, W. (2017). Reasonet: Learning to stop reading in machine comprehension. In *KDD*.

Sherstan, C., Machado, M. C., and Pilarski, P. M. (2018). Accelerating learning in constructive predictive frameworks with the successor representation. In *IROS*.

Shi, J.-C., Yu, Y., Da, Q., Chen, S.-Y., and Zeng, A.-X. (2018). Virtual-Taobao: Virtualizing Real-world Online Retail Environment for Reinforcement Learning. *ArXiv*.

Shoham, Y. and Leyton-Brown, K. (2009). *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press.

Shoham, Y., Powers, R., and Grenager, T. (2007). If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, 171:365–377.

Shortreed, S. M., Laber, E., Lizotte, D. J., Stroup, T. S., Pineau, J., and Murphy, S. A. (2011). Informing sequential clinical decision-making through reinforcement learning: an empirical study. *MLJ*, 84:109–136.

Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., and Webb, R. (2017). Learning from simulated and unsupervised images through adversarial training. In *CVPR*.

Silver, D. (2015). UCL reinforcement learning course. `http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html`.

Silver, D. (2016). Deep reinforcement learning, a tutorial at ICML 2016. `http://icml.cc/2016/tutorials/deep_rl_tutorial.pdf`.

Silver, D. (2017). Deep reinforcement learning with subgoals. `https://goo.gl/h9Mz1a`. NIPS 2017 Hierarchical RL Workshop Invited Talk.

Silver, D. (2018). Principles of deep rl. `http://www.deeplearningindaba.com/uploads/1/0/2/6/102657286/principles_of_deep_rl.pdf`. Deep Learning Indaba 2018.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016a). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. (2017). Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *ArXiv*.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *ICML*.

Silver, D., Newnham, L., Barker, D., Weller, S., and McFall, J. (2013). Concurrent reinforcement learning from customer interactions. In *ICML*.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. (2017). Mastering the game of go without human knowledge. *Nature*, 550:354–359.

Silver, D., Sutton, R. S., and Müller, M. (2012). Temporal-difference search in computer Go. *Machine Learning*, 87:183–219.

Silver, D., van Hasselt, H., Hessel, M., Schaul, T., Guez, A., Harley, T., Dulac-Arnold, G., Reichert, D., Rabinowitz, N., Barreto, A., and Degris, T. (2016b). The predictron: End-to-end learning and planning. In *NIPS 2016 Deep Reinforcement Learning Workshop*.

Singh, S. (2017). Steps towards continual learning. `https://mila.quebec/en/cours/deep-learning-summer-school-2017/`. Deep Learning and Reinforcement Learning Summer School 2017.

Singh, S., Jaakkola, T., Littman, M. L., and Szepesvári, C. (2000). Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38(3):287–308.

Singh, S., James, M., and Rudary, M. (2004). Predictive state representations: A new theory for modeling dynamical systems. In *the Conference on Uncertainty in Artificial Intelligence (UAI)*.

Singh, S., Lewis, R., Barto, A., and Sorg, J. (2010). Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2).

Singh, S., Lewis, R. L., and Barto, A. G. (2009). Where do rewards come from? In *the Annual Conference of the Cognitive Science Society (CogSci)*.

Sirignano, J. (2016). Deep Learning for Limit Order Books. *ArXiv*.

Smith, V., Chiang, C.-K., Sanjabi, M., and Talwalkar, A. (2017). Federated multi-task learning. In *NIPS*.

Snell, J., Swersky, K., and Zemel, R. S. (2017). Prototypical Networks for Few-shot Learning. *ArXiv*.

Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., and Manning, C. D. (2011). Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP*.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment tree- bank. In *EMNLP*.

Sohn, S., Oh, J., and Lee, H. (2018). Multitask reinforcement learning for zero-shot generalization with subtask dependencies. In *NIPS*.

Sommer, R. and Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In *IEEE Symposium on Security and Privacy (SP)*.

Song, D. (2018). AI and security: Lessons, challenges and future directions. `https://www.youtube.com/watch?v=dcEo2r7_V44`. ICML 2018 Invited Talk.

Song, J., Ren, H., Sadigh, D., and Ermon, S. (2018). Multi-agent generative adversarial imitation learning. In *NIPS*.

Srinivas, A., Jabri, A., Abbeel, P., Levine, S., and Finn, C. (2018). Universal planning networks. In *ICML*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15:1929–1958.

Stadie, B., Yang, G., Abbeel, P., Wu, Y., Duan, Y., Chen, X., Houthooft, R., and Sutskever, I. (2018). The importance of sampling in meta-reinforcement learning. In *NIPS*.

Stadie, B. C., Abbeel, P., and Sutskever, I. (2017). Third person imitation learning. In *ICLR*.

Stoica, I., Song, D., Popa, R. A., Patterson, D. A., Mahoney, M. W., Katz, R. H., Joseph, A. D., Jordan, M., Hellerstein, J. M., Gonzalez, J., Goldberg, K., Ghodsi, A., Culler, D. E., and Abbeel, P. (2017). A berkeley view of systems challenges for AI. *Technical Report No. UCB/EECS-2017-159*.

Stone, P., Brooks, R., Brynjolfsson, E., Calo, R., Etzioni, O., Hager, G., Hirschberg, J., Kalyanakrishnan, S., Kamar, E., Kraus, S., Leyton-Brown, K., Parkes, D., Press, W., Saxenian, A., Shah, J., Tambe, M., and Teller, A. (2016). *Artificial Intelligence and Life in 2030 - One Hundred Year Study on Artificial Intelligence: Report of the 2015-2016 Study Panel*. Stanford University.

Stone, P. and Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383.

Strehl, A. L., Li, L., and Littman, M. L. (2009). Reinforcement learning in finite MDPs: PAC analysis. *JMLR*, 10:2413–2444.

Strehl, A. L. and Littman, M. L. (2008). An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74:1309–1331.

Strub, F., de Vries, H., Mary, J., Piot, B., Courville, A., and Pietquin, O. (2017). End-to-end optimization of goal-driven and visually grounded dialogue systems. In *IJCAI*.

Su, P.-H., Gasˇić, M., Mrksˇić, N., Rojas-Barahona, L., Ultes, S., Vandyke, D., Wen, T.-H., and Young, S. (2016). On-line active reward learning for policy optimisation in spoken dialogue systems. In *ACL*.

Sukhbaatar, S., Szlam, A., and Fergus, R. (2016). Learning multiagent communication with backpropagation. In *NIPS*.

Sukhbaatar, S., Weston, J., and Fergus, R. (2015). End-to-end memory networks. In *NIPS*.

Sun, P., Sun, X., Han, L., Xiong, J., Wang, Q., Li, B., Zheng, Y., Liu, J., Liu, Y., Liu, H., and Zhang, T. (2018). TStarBots: Defeating the Cheating Level Builtin AI in StarCraft II in the Full Game. *ArXiv e-prints*.

Sun, W., Gordon, G., Boots, B., and Bagnell, J. (2018). Dual policy iteration. In *NIPS*.

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., and Graepel, T. (2017). Value-decomposition networks for cooperative multi-agent learning. In *AAMAS*.

Supančič, III, J. and Ramanan, D. (2017). Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning. In *ICCV*.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *NIPS*.

Sutton, R. (2016). Reinforcement learning for artificial intelligence, course slides. http://www.incompleteideas.net/sutton/609%20dropbox/.

Sutton, R. (2018). The next big step in AI: Planning with a learned model. https://www.youtube.com/watch?v=6-Uiq8-wKrg. University of Alberta RLAI Tea Time Talk.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *ICML*.

Sutton, R. S. (1992). Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *AAAI*.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction (2nd Edition)*. MIT Press.

Sutton, R. S., Barto, A. G., and Williams, R. J. (1992). Reinforcement learning is direct adaptive optimal control. *IEEE Control Systems*, 12(2):19–22.

Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, C., and Wiewiora, E. (2009a). Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *ICML*.

Sutton, R. S., Mahmood, A. R., and White, M. (2016). An emphatic approach to the problem of off-policy temporal-difference learning. *JMLR*, 17:1–29.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *NIPS*.

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., and Precup, D. (2011). Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction, , proc. of 10th. In *AAMAS*.

Sutton, R. S., Precup, D., and Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211.

Sutton, R. S., Rafols, E. J., and Koop, A. (2005). Temporal abstraction in temporal-difference networks. In *NIPS*.

Sutton, R. S., Szepesvári, C., and Maei, H. R. (2009b). A convergent O(n) algorithm for off-policy temporal-difference learning with linear function approximation. In *NIPS*.

Sutton, R. S. and Tanner, B. (2004). Temporal-difference networks. In *NIPS*.

Syed, U., Bowling, M., and Schapire, R. E. (2008). Apprenticeship learning using linear programming. In *ICML*.

Syed, U. and Schapire, R. E. (2007). A game-theoretic approach to apprenticeship learning. In *NIPS*.

Syed, U. and Schapire, R. E. (2010). A reduction from apprenticeship learning to classification. In *NIPS*.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *ArXiv*.

Szepesvári, C. (2010). *Algorithms for Reinforcement Learning*. Morgan & Claypool.

Tadepalli, P., Givan, R., and Driessens, K. (2004). Relational reinforcement learning: An overview. In *ICML Workshop on Relational Reinforcement Learning*.

Tamar, A., Abbeel, P., Yang, G., Kurutach, T., and Russell, S. (2018). Learning plannable representations with causal InfoGAN. In *NIPS*.

Tamar, A., Wu, Y., Thomas, G., Levine, S., and Abbeel, P. (2016). Value iteration networks. In *NIPS*.

Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. In *ICML*.

Tang, D., Li, X., Gao, J., Wang, C., Li, L., and Jebara, T. (2018a). Subgoal discovery for hierarchical dialogue policy learning. In *EMNLP*.

Tang, G., Muller, M., Rios, A., and Sennrich, R. (2018b). Why self-attention? a targeted evaluation of neural machine translation architectures. In *EMNLP*.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, X., Duan, Y., Schulman, J., Turck, F. D., and Abbeel, P. (2017). Exploration: A study of count-based exploration for deep reinforcement learning. In *NIPS*.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., de Las Casas, D., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T., and Riedmiller, M. (2018). DeepMind Control Suite. *ArXiv*.

Taylor, M. E. and Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *JMLR*, 10:1633–1685.

Tenenbaum, J. (2018). Building machines that learn & think like people. `https://www.youtube.com/watch?v=RB78vRUO6X8`. ICML 2018 Invited Talk.

Tesauro, G. (1994). TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219.

Tessler, C., Givony, S., Zahavy, T., Mankowitz, D. J., and Mannor, S. (2017). A deep hierarchical approach to lifelong learning in minecraft. In *AAAI*.

Theocharous, G., Thomas, P. S., and Ghavamzadeh, M. (2015). Personalized ad recommendation systems for life-time value optimization with guarantees. In *IJCAI*.

Thrun, S. and Pratt, L., editors (1998). *Learning to Learn*. Springer.

Thue, D., Bulitko, V., Spetch, M., and Wasylishen, E. (2007). Interactive storytelling: A player modelling approach. In *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*.

Tian, Y., Gong, Q., Shang, W., Wu, Y., and Zitnick, L. (2017). ELF: An extensive, lightweight and flexible research platform for real-time strategy games. In *NIPS*.

Tian, Y. and Zhu, Y. (2016). Better computer go player with neural network and long-term prediction. In *ICLR*.

Trischler, A., Ye, Z., Yuan, X., and Suleman, K. (2016). Natural language comprehension with the epireader. In *EMNLP*.

Tsitsiklis, J. N. and Van Roy, B. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690.

Tsitsiklis, J. N. and Van Roy, B. (2001). Regression methods for pricing complex American-style options. *IEEE Transactions on Neural Networks*, 12(4):694–703.

Upadhyay, U., De, A., and Rodriguez, M. G. (2018). Deep reinforcement learning of marked temporal point processes. In *NIPS*.

Urban, J., Kaliszyk, C., Michalewski, H., and Olšák, M. (2018). Reinforcement learning of theorem proving. In *NIPS*.

Usunier, N., Synnaeve, G., Lin, Z., and Chintala, S. (2017). Episodic exploration for deep deterministic policies: An application to StarCraft micromanagement tasks. In *ICLR*.

Valiant, L. (1984). A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). WaveNet: A Generative Model for Raw Audio. *ArXiv*.

van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. (2016). Conditional image generation with PixelCNN decoders. In *NIPS*.

van den Oord, A., Li, Y., and Vinyals, O. (2018). Representation Learning with Contrastive Predictive Coding. *ArXiv*.

van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-SNE. *JMLR*, pages 1–48.

van der Pol, E. and Oliehoek, F. A. (2017). Coordinated deep reinforcement learners for traffic light control. In *NIPS'16 Workshop on Learning, Inference and Control of Multi-Agent Systems*.

van Hasselt, H. (2010). Double Q-learning. In *NIPS*.

van Hasselt, H., Guez, A., , and Silver, D. (2016). Deep reinforcement learning with double Q-learning. In *AAAI*.

van Seijen, H., Fatemi, M., Romoff, J., Laroche, R., Barnes, T., and Tsang, J. (2017). Hybrid reward architecture for reinforcement learning. In *NIPS*.

van Seijen, H., van Hasselt, H., Whiteson, S., and Wiering, M. (2009). A theoretical and empirical analysis of expected sarsa. In *ADPRL*.

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *NIPS*.

Venkatraman, A., Rhinehart, N., Sun, W., Pinto, L., Hebert, M., Boots, B., Kitani, K. M., and Bagnell, J. A. (2017). Predictive-state decoders: Encoding the future into recurrent networks. In *NIPS*.

Večerík, M., Hester, T., Scholz, J., Wang, F., Pietquin, O., Piot, B., Heess, N., Rothörl, T., Lampe, T., and Riedmiller, M. (2017). Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. In *NIPS*.

Vezhnevets, A. S., Mnih, V., Agapiou, J., Osindero, S., Graves, A., Vinyals, O., and Kavukcuoglu, K. (2016). Strategic attentive writer for learning macro-actions. In *NIPS*.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., and Kavukcuoglu, K. (2017). Feudal networks for hierarchical reinforcement learning. In *ICML*.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. (2016). Matching networks for one shot learning. In *NIPS*.

Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Sasha Vezhnevets, A., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittwieser, J., Quan, J., Gaffney, S., Petersen, S., Simonyan, K., Schaul, T., van Hasselt, H., Silver, D., Lillicrap, T., Calderone, K., Keet, P., Brunasso, A., Lawrence, D., Ekermo, A., Repp, J., and Tsing, R. (2017). StarCraft II: A New Challenge for Reinforcement Learning. *ArXiv*.

Vinyals, O., Fortunato, M., and Jaitly, N. (2015). Pointer networks. In *NIPS*.

Wai, H.-T., Wang, P. Z., Yang, Z., and Hong, M. (2018). Multi-agent reinforcement learning via double averaging primal-dual optimization. In *NIPS*.

Wang, J., Yu, L., Zhang, W., Gong, Y., Xu, Y., Wang, B., Zhang, P., and Zhang, D. (2017a). IRGAN: A minimax game for unifying generative and discriminative information retrieval models. In *SIGIR*.

Wang, J. X., Kurth-Nelson, Z., Kumaran, D., Tirumala, D., Soyer, H., Leibo, J. Z., Hassabis, D., and Botvinick, M. (2018a). Prefrontal cortex as a meta-reinforcement learning system. *Nature Neuroscience*, 21:860–868.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. (2016). Learning to reinforcement learn. *ArXiv*.

Wang, S. I., Liang, P., and Manning, C. D. (2016a). Learning language games through interaction. In *ACL*.

Wang, T., Liao, R., Ba, J., and Fidler, S. (2018b). Nervenet: Learning structured policy with graph neural networks. In *ICLR*.

Wang, T., WU, Y., Moore, D., and Russell, S. (2018c). Meta-learning MCMC proposals. In *NIPS*.

Wang, W., Yang, N., Wei, F., Chang, B., and Zhou, M. (2017b). Gated self-matching networks for reading comprehension and question answering. In *ACL*.

Wang, W. Y., Li, J., and He, X. (2018d). Deep reinforcement learning for NLP. `https://www.cs.ucsb.edu/~william/papers/ACL2018DRL4NLP.pdf`. ACL 2018 Tutorial.

Wang, X., Chen, W., Wang, Y.-F., and Wang, W. Y. (2018e). No metrics are perfect: Adversarial reward learning for visual storytelling. In *ACL*.

Wang, X., Chen, W., Wu, J., Wang, Y.-F., and Wang, W. Y. (2018f). Video captioning via hierarchical reinforcement learning. In *CVPR*.

Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., and de Freitas, N. (2017c). Sample efficient actor-critic with experience replay. In *ICLR*.

Wang, Z., Merel, J., Reed, S., Wayne, G., de Freitas, N., and Heess, N. (2017). Robust Imitation of Diverse Behaviors. *ArXiv*.

Wang, Z. and O'Boyle, M. (2018). Machine learning in compiler optimization. *Proceedings of the IEEE*, PP(99):1–23.

Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., and de Freitas, N. (2016b). Dueling network architectures for deep reinforcement learning. In *ICML*.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8:279–292.

Watter, M., Springenberg, J. T., Boedecker, J., and Riedmiller, M. (2015). Embed to control: A locally linear latent dynamics model for control from raw images. In *NIPS*.

Watters, N., Tacchetti, A., Weber, T., Pascanu, R., Battaglia, P., and Zoran, D. (2017). Visual interaction networks: Learning a physics simulator from video. In *NIPS*.

Wayne, G., Hung, C.-C., Amos, D., Mirza, M., Ahuja, A., Grabska-Barwinska, A., Rae, J., Mirowski, P., Leibo, J. Z., Santoro, A., Gemici, M., Reynolds, M., Harley, T., Abramson, J., Mohamed, S., Rezende, D., Saxton, D., Cain, A., Hillier, C., Silver, D., Kavukcuoglu, K., Botvinick, M., Hassabis, D., and Lillicrap, T. (2018). Unsupervised Predictive Memory in a Goal-Directed Agent. *ArXiv*.

Weber, T., Racanière, S., Reichert, D. P., Buesing, L., Guez, A., Jimenez Rezende, D., Puigdomènech Badia, A., Vinyals, O., Heess, N., Li, Y., Pascanu, R., Battaglia, P., Silver, D., and Wierstra, D. (2017). Imagination-augmented agents for deep reinforcement learning. In *NIPS*.

Weiss, K., Khoshgoftaar, T. M., and Wang, D. (2016). A survey of transfer learning. *Journal of Big Data*, 3(9).

Welleck, S., Mao, J., Cho, K., and Zhang, Z. (2017). Saliency-based sequential image attention with multiset prediction. In *NIPS*.

Wen, M. and Topcu, U. (2018). Constrained cross-entropy method for safe reinforcement learning. In *NIPS*.

Wen, T.-H., Vandyke, D., Mrksic, N., Gasic, M., Rojas-Barahona, L. M., Su, P.-H., Ultes, S., and Young, S. (2017). A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Wen, Z., O'Neill, D., and Maei, H. (2015). Optimal demand response using device-based reinforcement learning. *IEEE Transactions on Smart Grid*, 6(5):2312–2324.

Weston, J., Chopra, S., and Bordes, A. (2015). Memory networks. In *ICLR*.

White, A. and White, M. (2016). Investigating practical linear temporal difference learning. In *AAMAS*.

Whye Teh, Y., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., and Pascanu, R. (2017). Distral: Robust multitask reinforcement learning. In *NIPS*.

Wichrowska, O., Maheswaranathan, N., Hoffman, M. W., Gomez Colmenarejo, S., Denil, M., de Freitas, N., and Sohl-Dickstein, J. (2017). Learned optimizers that scale and generalize. In *ICML*.

Williams, J. D., Asadi, K., and Zweig, G. (2017). Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *ACL*.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256.

Winston, P. H. (1992). *Artificial Intelligence*. Pearson.

Wong, C., Houlsby, N., Lu, Y., and Gesmundo, A. (2018). Transfer learning with neural AutoML. In *NIPS*.

Wu, J., Li, L., and Wang, W. Y. (2018). Reinforced co-training. In *NAACL*.

Wu, J., Lu, E., Kohli, P., Freeman, B., and Tenenbaum, J. (2017a). Learning to see physics via visual de-animation. In *NIPS*.

Wu, J., Tenenbaum, J. B., and Kohli, P. (2017b). Neural scene de-rendering. In *CVPR*.

Wu, J., Yildirim, I., Lim, J. J., Freeman, B., and Tenenbaum, J. (2015). Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *NIPS*.

Wu, L., Xia, Y., Zhao, L., Tian, F., Qin, T., Lai, J., and Liu, T.-Y. (2017c). Adversarial Neural Machine Translation. *ArXiv*.

Wu, Y., Mansimov, E., Liao, S., Grosse, R., and Ba, J. (2017). Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *NIPS*.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv*.

Wu, Y. and Tian, Y. (2017). Training agent for first-person shooter game with actor-critic curriculum learning. In *ICLR*.

Wymann, B., Espié, E., Guionneau, C., Dimitrakakis, C., and Rémi Coulom, A. S. (2014). TORCS, The Open Racing Car Simulator. `http://www.torcs.org`.

Xia, Y., Tan, X., Tian, F., Qin, T., Yu, N., and Liu, T.-Y. (2018). Model-level dual learning. In *ICML*.

Xiao, C., JinchengMei, and Müller, M. (2018). Memory-augmented monte carlo tree search. In *AAAI*.

Xie, N., Hachiya, H., and Sugiyama, M. (2012). Artist agent: A reinforcement learning approach to automatic stroke generation in oriental ink painting. In *ICML*.

Xiong, C., Zhong, V., and Socher, R. (2017a). Dynamic coattention networks for question answering. In *ICLR*.

Xiong, W., Droppo, J., Huang, X., Seide, F., Seltzer, M., Stolcke, A., Yu, D., and Zweig, G. (2017b). The microsoft 2016 conversational speech recognition system. In *The IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Xiong, W., Hoang, T., and Wang, W. Y. (2017c). Deeppath: A reinforcement learning method for knowledge graph reasoning. In *EMNLP*.

Xiong, W., Wu, L., Alleva, F., Droppo, J., Huang, X., and Stolcke, A. (2017). The Microsoft 2017 Conversational Speech Recognition System. *ArXiv*.

Xu, J. and Zhu, Z. (2018). Reinforced continual learning. In *NIPS*.

Xu, K., Ba, J. L., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.

Xu, L. D., He, W., and Li, S. (2014). Internet of things in industries: A survey. *IEEE Transactions on Industrial Informatics*, 10(4):2233–2243.

Xu, Z., van Hasselt, H., and Silver, D. (2018). Meta-Gradient Reinforcement Learning. *ArXiv*.

Yahya, A., Li, A., Kalakrishnan, M., Chebotar, Y., and Levine, S. (2017). Collective robot reinforcement learning with distributed asynchronous guided policy search. In *IROS*.

Yang, B. and Mitchell, T. (2017). Leveraging knowledge bases in LSTMs for improving machine reading. In *ACL*.

Yang, F., Lyu, D., Liu, B., and Gustafson, S. (2018). Peorl: Integrating symbolic planning and hierarchical reinforcement learning for robust decision-making. In *IJCAI*.

Yang, Z., He, X., Gao, J., Deng, L., and Smola, A. (2016). Stacked attention networks for image question answering. In *CVPR*.

Yang, Z., Hu, J., Salakhutdinov, R., and Cohen, W. W. (2017). Semi-supervised qa with generative domain-adaptive nets. In *ACL*.

Yannakakis, G. N. and Togelius, J. (2018). *Artificial Intelligence and Games*. Springer.

Yao, H., Szepesvari, C., Sutton, R. S., Modayil, J., and Bhatnagar, S. (2014). Universal option models. In *NIPS*.

Yi, K., Wu, J., Gan, C., Torralba, A., Kohli, P., and Tenenbaum, J. (2018). Neural-Symbolic VQA: Disentangling reasoning from vision and language understanding. In *NIPS*.

Yi, Z., Zhang, H., Tan, P., and Gong, M. (2017). Dualgan: Unsupervised dual learning for image-to-image translation. In *ICCV*.

Yogatama, D., Blunsom, P., Dyer, C., Grefenstette, E., and Ling, W. (2017). Learning to compose words into sentences with reinforcement learning. In *ICLR*.

Yoon, J., Kim, T., Dia, O., Kim, S., Bengio, Y., and Ahn, S. (2018). Bayesian model-agnostic meta-learning. In *NIPS*.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *NIPS*.

Young, S., Gašić, M., Thomson, B., and Williams, J. D. (2013). POMDP-based statistical spoken dialogue systems: a review. *Proceedings of IEEE*, 101(5):1160–1179.

Young, T., Hazarika, D., Poria, S., and Cambria, E. (2017). Recent Trends in Deep Learning Based Natural Language Processing. *ArXiv*.

Yu, F., Xian, W., Chen, Y., Liu, F., Liao, M., Madhavan, V., and Darrell, T. (2018). BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling. *ArXiv*.

Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017). SeqGAN: Sequence generative adversarial nets with policy gradient. In *AAAI*.

Yu, T., Finn, C., Xie, A., Dasari, S., Zhang, T., Abbeel, P., and Levine, S. (2018). One-shot imitation from observing humans via domain-adaptive meta-learning. In *RSS*.

Yu, Y.-L., Li, Y., Szepesvári, C., and Schuurmans, D. (2009). A general projection property for distribution families. In *NIPS*.

Yuan, X., He, P., Zhu, Q., and Li, X. (2017). Adversarial Examples: Attacks and Defenses for Deep Learning. *ArXiv*.

Yue, Y. and Le, H. M. (2018). Imitation learning. `https://sites.google.com/view/icml2018-imitation-learning/`. ICML 2018 Tutorial.

Yun, S., Choi, J., Yoo, Y., Yun, K., and Young Choi, J. (2017). Action-decision networks for visual tracking with deep reinforcement learning. In *CVPR*.

Zagoruyko, S. and Komodakis, N. (2017). Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*.

Zahavy, T., Harush, M., Merlis, N., Mankowitz, D. J., and Mannor, S. (2018). Learn what not to learn: Action elimination with deep reinforcement learning. In *NIPS*.

Zambaldi, V., Raposo, D., Santoro, A., Bapst, V., Li, Y., Babuschkin, I., Tuyls, K., Reichert, D., Lillicrap, T., Lockhart, E., Shanahan, M., Langston, V., Pascanu, R., Botvinick, M., Vinyals, O., and Battaglia, P. (2018). Relational Deep Reinforcement Learning. *ArXiv*.

Zaremba, W. and Sutskever, I. (2015). Reinforcement Learning Neural Turing Machines - Revised. *ArXiv*.

Zhang, C., Patras, P., and Haddadi, H. (2018). Deep Learning in Mobile and Wireless Networking: A Survey. *ArXiv*.

Zhang, J., Ding, Y., Shen, S., Cheng, Y., Sun, M., Luan, H., and Liu, Y. (2017). THUMT: An Open Source Toolkit for Neural Machine Translation. *ArXiv*.

Zhang, J., Springenberg, J. T., Boedecker, J., and Burgard, W. (2017). Deep reinforcement learning with successor features for navigation across similar environments. In *IROS*.

Zhang, L., Rosenblatt, G., Fetaya, E., Liao, R., Byrd, W., Might, M., Urtasun, R., and Zemel, R. (2018). Neural guided constraint logic programming for program synthesis. In *NIPS*.

Zhang, L., Wang, S., and Liu, B. (2018). Deep Learning for Sentiment Analysis : A Survey. *ArXiv*.

Zhang, Q. and Zhu, S.-C. (2018). Visual interpretability for deep learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 19(1):27–39.

Zhang, S., Yao, L., Sun, A., and Tay, Y. (2017). Deep Learning based Recommender System: A Survey and New Perspectives. *ArXiv e-prints*.

Zhang, X. and Lapata, M. (2017). Sentence simplification with deep reinforcement learning. In *EMNLP*.

Zhang, Y., , and Yang, Q. (2018a). An overview of multi-task learning. *National Science Review*, 5:30–43.

Zhang, Y., Mustafizur Rahman, M., Braylan, A., Dang, B., Chang, H.-L., Kim, H., McNamara, Q., Angert, A., Banner, E., Khetan, V., McDonnell, T., Thanh Nguyen, A., Xu, D., Wallace, B. C., and Lease, M. (2016a). Neural Information Retrieval: A Literature Review. *ArXiv*.

Zhang, Y., Pezeshki, M., Brakel, P., Zhang, S., Yoshua Bengio, C. L., and Courville, A. (2016b). Towards end-to-end speech recognition with deep convolutional neural networks. In *Interspeech*.

Zhang, Y., Wei, Y., and Yang, Q. (2018b). Learning to multitask. In *NIPS*.

Zhao, H., Zhang, S., Wu, G., Moura, J. M. F., Costeira, J. P., and Gordon, G. (2018a). Adversarial multiple source domain adaptation. In *NIPS*.

Zhao, T. and Eskenazi, M. (2016). Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *the Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL)*.

Zhao, X., Xia, L., Zhang, L., Ding, Z., Yin, D., and Tang, J. (2018b). Deep reinforcement learning for page-wise recommendations. In *the ACM Conference on Recommender Systems (ACM RecSys)*.

Zhao, X., Zhang, L., Ding, Z., Xia, L., Tang, J., and Yin, D. (2018c). Recommendations with negative feedback via pairwise deep reinforcement learning. In *KDD*.

Zheng, G., Zhang, F., Zheng, Z., Xiang, Y., Yuan, N. J., Xie, X., and Li, Z. (2018a). DRN: A deep reinforcement learning framework for news recommendation. In *WWW*.

Zheng, Z., Oh, J., and Singh, S. (2018b). On learning intrinsic rewards for policy gradient methods. In *NIPS*.

Zhong, Z., Yan, J., and Liu, C.-L. (2017). Practical Network Blocks Design with Q-Learning. *ArXiv*.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. (2015). Object detectors emerge in deep scene CNNs. In *ICLR*.

Zhou, H., Huang, M., Zhang, T., Zhu, X., and Liu, B. (2018). Emotional chatting machine: Emotional conversation generation with internal and external memory. In *AAAI*.

Zhou, Y. and Tuzel, O. (2018). Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*.

Zhou, Z., Mertikopoulos, P., Athey, S., Bambos, N., Glynn, P. W., and Ye, Y. (2018). Multi-agent online learning with asynchronous feedback loss. In *NIPS*.

Zhou, Z.-H. (2016). *Machine Learning (in Chinese)*. Tsinghua University Press, Beijing, China.

Zhou, Z.-H. and Feng, J. (2017). Deep forest: Towards an alternative to deep neural networks. In *IJCAI*.

Zhu, H., Liu, Q., Yuan, N. J., Qin, C., Li, J., Zhang, K., Zhou, G., Wei, F., Xu, Y., and Chen, E. (2018). XiaoIce Band: A melody and arrangement generation framework for pop music. In *KDD*.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017a). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*.

Zhu, X. and Goldberg, A. B. (2009). *Introduction to semi-supervised learning*. Morgan & Claypool.

Zhu, Y., Mottaghi, R., Kolve, E., Lim, J. J., Gupta, A., Li, F.-F., and Farhadi, A. (2017b). Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*.

Ziebart, B. D., Bagnell, J. A., and Dey, A. K. (2010). Modeling interaction via the principle of maximum causal entropy. In *ICML*.

Ziebart, B. D., Maas, A., Bagnell, J., and Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. In *AAAI*.

Zinkevich, M. (2017). *Rules of Machine Learning: Best Practices for ML Engineering*. `http://martin.zinkevich.org/rules_of_ml/rules_of_ml.pdf`.

Zoph, B. and Le, Q. V. (2017). Neural architecture search with reinforcement learning. In *ICLR*.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. (2017). Learning Transferable Architectures for Scalable Image Recognition. *ArXiv*.