Cambridge Small (2020), **37**, 1–8

CAMBRIDGE
UNIVERSITY PRESS

ARTICLE

Reinforcement Learning based Statistical Complexity Measuring for Deep Learning

Manda Kausthubh* and Harsh Modani*

Dept. Computer Science, IIIT Bangalore, Bangalore, India
*Corresponding author. Email: manda.kausthubh@iiitb.ac.in; harsh.modani@iiitb.ac.in

Abstract

Measuring Statistical Complexity in Deep Learning networks remains a difficult task. Methods and abstractions such as Vapnik-Chenikov Bounds and Rademacher bounds provide useful insights into the theory of learning and provide an excellent theoretical baseline for measuring statistical complexity, however, for Deep Learning, these methods fail to provide reasonable bounds with a great gap between theory and actual difference between empirical and expected risk. Reinforcement Learning provides a unique set of problems where learning play a unique roles as a function approximation. This submission looks at using RL to measure statistical complexity.

Keywords: Reinforcement Learning, Deep Learning and Statistical Complexity

1. Introduction

Statistical learning theory provides a framework for understanding how machine learning models generalize from training data to unseen data. Central to this framework is the notion of *statistical complexity*, which measures the capacity of a hypothesis class to fit data. Two key measures of complexity are the Vapnik-Chervonenkis (VC) dimension and Rademacher complexity, which provide insights into the trade-off between model expressiveness and generalization. This document explores these concepts and their implications.

1.1 VC Dimension

1.1.1 Definition

The Vapnik-Chervonenkis (VC) dimension is a measure of the capacity of a binary classification hypothesis class. Formally, let $\mathcal{H}$ be a class of functions mapping from an input space $\mathcal{X}$ to $\{0, 1\}$. The VC dimension of $\mathcal{H}$, denoted $\mathrm{VC}(\mathcal{H})$, is the size of the largest set of points $\{x_1, \ldots, x_d\} \subset \mathcal{X}$ that can be *shattered* by $\mathcal{H}$. A set is shattered if, for all 2^d possible labelings of the points, there exists a hypothesis $h \in \mathcal{H}$ that correctly assigns those labels.

$$\mathrm{VC}(\mathcal{H}) = \max\{d \mid \exists\{x_1, \ldots, x_d\} \text{ s.t. } \mathcal{H} \text{ shatters } \{x_1, \ldots, x_d\}\}. \tag{1}$$

1.1.2 Example

Consider the hypothesis class of linear classifiers in R^2. This class can shatter any set of 3 points in general position (i.e., not collinear), as a line can separate the points for all $2^3 = 8$ labelings. However, no set of 4 points can be shattered, as certain configurations (e.g., three points forming a triangle with the fourth inside) cannot be separated for all labelings. Thus, the VC dimension is 3.

1.1.3 Significance

The VC dimension quantifies the expressive power of $\mathcal{H}$. A larger VC dimension implies a more flexible hypothesis class, but it also increases the risk of overfitting. VC dimension is used to derive generalization bounds, which relate the training error to the expected test error. For a hypothesis class with VC dimension d, the generalization error can be bounded with high probability as:

$$R(h) \leq \hat{R}(h) + O\left(\sqrt{\frac{d\log(n/d) + \log(1/\delta)}{n}}\right), \tag{2}$$

where $R(h)$ is the true risk, $\hat{R}(h)$ is the empirical risk, n is the sample size, and δ is the confidence parameter.

1.2 Rademacher Complexity

1.2.1 Definition

Rademacher complexity measures the ability of a hypothesis class to fit random noise, providing a data-dependent measure of complexity. Let $\mathcal{H}$ be a hypothesis class, and let $S = \{x_1, \ldots, x_n\}$ be a sample of n points. The *empirical Rademacher complexity* of $\mathcal{H}$ on S is defined as:

$$\hat{\mathcal{R}}_S(\mathcal{H}) = E_\sigma\left[\sup_{h\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i h(x_i)\right], \tag{3}$$

where $\sigma_i \sim \text{Unif}(\{-1, 1\})$ are independent Rademacher random variables. The *Rademacher complexity* is the expectation over samples:

$$\mathcal{R}_n(\mathcal{H}) = E_S\left[\hat{\mathcal{R}}_S(\mathcal{H})\right]. \tag{4}$$

1.2.2 Interpretation

Rademacher complexity captures how well $\mathcal{H}$ can correlate with random labels. A high Rademacher complexity indicates that $\mathcal{H}$ is rich and can fit noise, which may lead to poor generalisation. Unlike VC dimension, Rademacher complexity is data-dependent and accounts for the distribution of the sample.

1.2.3 Generalization Bounds

Rademacher complexity provides tighter generalisation bounds than VC dimension in many cases. For a hypothesis class $\mathcal{H}$ with bounded outputs (e.g., $|h(x)| \leq 1$), the generalisation error is bounded with probability at least $1 - \delta$ as:

$$R(h) \leq \hat{R}(h) + 2\mathcal{R}_n(\mathcal{H}) + O\left(\sqrt{\frac{\log(1/\delta)}{n}}\right). \tag{5}$$

This bound is particularly useful because it adapts to the data distribution and can be computed empirically for specific hypothesis classes.

1.2.4 Relationship Between VC Dimension and Rademacher Complexity

For hypothesis classes with finite VC dimension, the Rademacher complexity can be bounded in terms of the VC dimension. Specifically, for a hypothesis class $\mathcal{H}$ with $\text{VC}(\mathcal{H}) = d$, the Rademacher complexity scales as:

$$\mathcal{R}_n(\mathcal{H}) = O\left(\sqrt{\frac{d}{n}}\right). \tag{6}$$

This relationship shows that VC dimension provides a worst-case bound, while Rademacher complexity offers a more refined, data-dependent measure. For large n, Rademacher bounds are often tighter, making them valuable in practical settings.

1.3 Applications of Statistical Complexity

Both VC dimension and Rademacher complexity are widely used in:

- **Model selection**: Choosing hypothesis classes with appropriate complexity to balance bias and variance.
- **Algorithm design**: Developing learning algorithms with provable generalization guarantees.
- **Analysis of neural networks**: Estimating the complexity of deep learning models, where VC dimension may be large but Rademacher complexity provides practical bounds.

2. Reinforcement Learning and Deep Q-Learning

2.1 Overview

Deep Q-Networks (DQNs), introduced by Mnih et al. (2015), represent a significant advancement in reinforcement learning (RL) by combining Q-learning with deep neural networks. DQNs enable agents to learn optimal policies in high-dimensional state spaces, such as raw pixel inputs from Atari games, by approximating the Q-function with a neural network. This subsection outlines the core components, innovations, and significance of DQNs.

2.2 Q-Learning Foundation

Q-learning, a model-free RL algorithm, learns an action-value function $Q(s, a)$ representing the expected cumulative reward for taking action a in state s and following the optimal policy. The Q-function is updated using the Bellman equation:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right], \tag{7}$$

where r is the reward, s' is the next state, α is the learning rate, and $\gamma \in [0, 1)$ is the discount factor. However, Q-learning with tabular methods is infeasible for large state spaces, necessitating function approximation.

2.3 DQN Architecture

DQNs approximate the Q-function, $Q(s, a; \theta)$, using a deep neural network with parameters θ. The network takes state s as input and outputs Q-values for all possible actions a. The network is trained to minimize the loss:

$$L(\theta) = E \left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta) \right)^2 \right], \tag{8}$$

where θ^- denotes the parameters of a target network used to stabilize training.

2.4 Key Innovations

DQNs introduce two techniques to ensure stable learning:

2.4.1 Experience Replay

To reduce correlation in sequential data and improve sample efficiency, DQNs store transitions (s, a, r, s') in a replay buffer. Minibatches are sampled randomly from the buffer for training, smoothing the data distribution and reducing variance.

2.4.2 Target Network

A separate target network with parameters θ^- computes target Q-values. The target network is updated periodically by copying the online network's parameters ($\theta^- \leftarrow \theta$), mitigating instability from rapidly changing Q-value estimates.

2.5 Algorithm

The DQN algorithm involves:

1. Initializing the Q-network (θ), target network ($\theta^- = \theta$), and replay buffer.
2. For each episode:
   - Select action a in state s using an ϵ-greedy policy.
   - Execute a, observe r and s', and store (s, a, r, s') in the replay buffer.
   - Sample a minibatch, compute target $y = r + \gamma \max_{a'} Q(s', a'; \theta^-)$ (if non-terminal), and update θ by minimizing $(y - Q(s, a; \theta))^2$.
   - Periodically update $\theta^- \leftarrow \theta$.

2.6 Significance and Applications

DQNs achieved human-level performance on Atari 2600 games, demonstrating the power of deep RL in complex environments. They have been applied in gaming, robotics (e.g., manipulation tasks), and resource management (e.g., energy optimization). However, DQNs face challenges like overestimation bias and are limited to discrete action spaces, addressed by extensions like Double DQN and Dueling DQN. DQNs combine Q-learning with deep neural networks, leveraging experience replay and target networks to enable stable learning in high-dimensional spaces. Their success has made them a cornerstone of modern RL, inspiring further innovations in the field.

3. Generalization in Deep Q-Networks

3.1 Generalization and Statistical Complexity

Generalization in Deep Q-Networks (DQNs) refers to the ability of the neural network $Q(s, a; \theta)$ to accurately predict Q-values for unseen state-action pairs, ensuring robust performance in the target environment. In Q-learning, the neural network approximates the optimal Q-function $Q^*(s, a)$, defined as:

$$Q^*(s, a) = E\left[r(s, a, s') + \gamma \max_{a'} Q^*(s', a')\right], \tag{9}$$

where $r(s, a, s')$ is the reward, $\gamma \in [0, 1)$ is the discount factor, and s' is the next state. Statistical complexity, measured via Vapnik-Chervonenkis (VC) dimension or Rademacher complexity, quantifies the capacity of the hypothesis class $\mathcal{H}$ (the set of Q-functions representable by the network) to fit diverse Q-value patterns, balancing expressiveness and overfitting risk.

3.2 Varying Reward Functions

In standard DQNs, the reward function $r(s, a, s')$ is fixed. Varying reward functions introduce different tasks, each with a distinct $Q_i^*(s, a)$. Generalization across these requires $\mathcal{H}$ to approximate multiple Q-functions, increasing its complexity. The target function remains $Q^*(s, a)$, but training targets are computed as:

$$y = r(s, a, s') + \gamma \max_{a'} Q(s', a'; \theta^-), \tag{10}$$

using a target network with parameters θ^-. Generalization entails accurate predictions within a single reward function (within-task) and adaptability to new reward functions (cross-task).

3.3 Generalization Error and Complexity

The generalization error is bounded by:

$$E\left[|Q(s, a; \theta) - Q^*(s, a)|\right] \leq \hat{R}(\theta) + 2\mathcal{R}_n(\mathcal{H}) + O\left(\sqrt{\frac{\log(1/\delta)}{n}}\right), \tag{11}$$

where $\hat{R}(\theta)$ is the empirical loss, $\mathcal{R}_n(\mathcal{H})$ is the Rademacher complexity, n is the sample size, and δ is the confidence parameter. For varying reward functions, $\mathcal{H}$'s complexity grows, widening the bound. The VC dimension of $\mathcal{H}$ scales with the network's size, while $\mathcal{R}_n(\mathcal{H})$ depends on the data distribution, including states and rewards.

3.4 Reformulating Generalization

Generalisation in DQNs is reformulated as minimising the expected Bellman error across state-action pairs and reward functions:

$$E_{s,a,r_i}\left[\left(Q(s,a;\theta)-\left(r_i(s,a,s')+\gamma\max_{a'}Q(s',a';\theta)\right)\right)^2\right]. \tag{12}$$

Techniques like experience replay expose the network to diverse transitions, enhancing within-task generalisation. Regularisation (e.g., dropout) and multi-task learning constrain $\mathcal{H}$, improving cross-task generalisation. Meta-learning can further enable rapid adaptation to new reward functions.

3.5 Conclusion

Generalisation in DQNs hinges on balancing the statistical complexity of the Q-function approximation with the diversity of state-action pairs and reward functions. By controlling VC dimension or Rademacher complexity, DQNs can achieve robust performance within and across tasks, leveraging innovations like experience replay and target networks to stabilise learning.

4. Generalization Bound for Q-Learning with Varying Reward Functions

4.1 Introduction

In reinforcement learning (RL), particularly Q-learning with Deep Q-Networks (DQNs), the goal is to approximate the optimal action-value function $Q^*(s,a)$, which depends on the environment's reward function. When reward functions vary across tasks, generalization becomes critical, ensuring the learned Q-function performs well on unseen tasks. This subsection derives a generalization bound for Q-learning with varying reward functions, leveraging multi-task RL and Universal Value Function Approximators (UVFAs) to address the challenge of adapting to new reward structures.

4.2 Problem Setup

Consider a set of tasks, each defined by a reward function $R_i(s,a,s')$, drawn from a distribution $\mathcal{P}$. For each task i, the optimal Q-function is:

$$Q_i^*(s,a)=E\left[R_i(s,a,s')+\gamma\max_{a'}Q_i^*(s',a')\right], \tag{13}$$

where $\gamma\in[0,1)$ is the discount factor. In DQNs, a neural network $Q(s,a;\theta)$ approximates Q_i^*, but with varying rewards, a single Q-function may struggle to generalize. Instead, we use a UVFA, $Q(s,a,g;\theta)$, where g is a task identifier (e.g., a goal or reward specification), allowing the Q-function to condition on the task **schaul2015**.

We train $Q(s, a, g; \theta)$ on m tasks $\{g_1, \ldots, g_m\}$, each with n transitions (s, a, r_i, s'), totaling $N = m \cdot n$ samples. The objective is to minimize the average Bellman error:

$$L(\theta) = \frac{1}{N} \sum_{j=1}^{N} \left(Q(s_j, a_j, g_j; \theta) - \left(r_j + \gamma \max_{a'} Q(s'_j, a'; \theta^-) \right) \right)^2, \tag{14}$$

where θ^- are target network parameters. The generalization error for a new task $g \sim \mathcal{P}$ is:

$$E_{s,a,g} \left[\left(Q(s, a, g; \theta) - Q^*_g(s, a) \right)^2 \right]. \tag{15}$$

4.3 Generalization Bound

The hypothesis class $\mathcal{H}$ consists of Q-functions $Q(s, a, g; \theta)$. Due to the sequential nature of RL, standard supervised learning bounds do not directly apply, as targets depend on Q. However, we can adapt information-theoretic bounds from meta-RL, which account for task variability **chen2021**.

Let $\hat{L}(\theta)$ be the empirical loss on training tasks, and let $I(\theta; D)$ be the mutual information between parameters θ and training data D. With probability at least $1 - \delta$, the generalization error is bounded as:

$$E_{s,a,g} \left[\left(Q(s, a, g; \theta) - Q^*_g(s, a) \right)^2 \right] \leq \hat{L}(\theta) + \sqrt{\frac{I(\theta; D) + \log(1/\delta)}{m}} + \epsilon, \tag{16}$$

where ϵ accounts for task similarity, measured via a metric (e.g., Wasserstein distance between reward functions). The term $I(\theta; D)$ reflects the complexity of $\mathcal{H}$, and the bound tightens with more tasks (m) and samples (N).

If tasks are similar (small ϵ), the bound is tighter. For UVFAs, the Rademacher complexity $\mathcal{R}_N(\mathcal{H})$ can replace the mutual information term, yielding:

$$E_{s,a,g} \left[\left(Q(s, a, g; \theta) - Q^*_g(s, a) \right)^2 \right] \leq \hat{L}(\theta) + 2\mathcal{R}_N(\mathcal{H}) + O\left(\sqrt{\frac{\log(1/\delta)}{N}} \right). \tag{17}$$

This bound leverages the total sample size N, improving efficiency over task-count-based bounds **zhang2020**.

4.4 Task Similarity

Task similarity can be quantified using the difference in reward functions:

$$\epsilon = \sup_{s,a,s'} E_{g,g' \sim \mathcal{P}} \left[|R_g(s, a, s') - R_{g'}(s, a, s')| \right]. \tag{18}$$

Smaller ϵ implies closer Q^*_g and $Q^*_{g'}$, enhancing generalization. In Hidden-Parameter Block MDPs, similarity is modeled via latent parameters, further tightening bounds **zhang2020**.

4.5 Discussion

This bound combines empirical performance ($\hat{L}(\theta)$), model complexity ($I(\theta; D)$ or $\mathcal{R}_N(\mathcal{H})$), and task similarity ($\epsilon$). It is data-dependent, non-vacuous for deep networks, and scales with total samples, aligning with findings in meta-RL **wang2024**. Techniques like experience replay and regularization can reduce $\hat{L}(\theta)$ and $\mathcal{R}_N(\mathcal{H})$, improving generalization.

References

[1] Schaul, T., et al. (2015). Universal Value Function Approximators. *ICML*.

[2] Chen, Q., et al. (2021). Generalization Bounds For Meta-Learning: An Information-Theoretic Analysis. *NeurIPS*.

[3] Zhang, A., et al. (2020). Learning Robust State Abstractions for Hidden-Parameter Block MDPs. *arXiv:2007.07206*.

[4] Wang, C., et al. (2024). Theoretical Analysis of Meta Reinforcement Learning: Generalization Bounds and Convergence Guarantees. *arXiv:2405.13290*.

[5] Mnih, V., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533.

[6] Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). *Foundations of Machine Learning*. MIT Press.

Appendix 1. Example Appendix Section