Ruprecht-Karls-Universität Heidelberg

Fakultät für Mathematik und Informatik

Institut für Informatik

Masterarbeit

Bestärkendes Lernen für semantische Segmentierung von Bilddaten

Reinforcement Learning for semantic segmentation of image data

Name: Paul Hilt

Matrikelnummer: 3533310

Betreuer: Prof. Dr. rer. nat. Fred Hamprecht

Contents

List of Figures

0.1 Introduction

This is the template

Preliminaries

1.1 Reinforcement Learning

Please note that this is an aggressively shortened summary. For a deeper introduction please refer to [1]. The Reinforcement Learning problem originates from the idea of learning by interacting with an environment. The object that is learning is doing so by retrieving information from cause and effect. The causal model that is learned in such a way is updated with each change of the environment that can be related to some action. Therefore, the learned model fits the true model increasingly better with the number of induced causes and observed effects.
This type of learning problem can be modeled by "Finite Markov Decision Processes". Such processes usually need the following elements:

- **Environment** The environment is a dynamic model of some complex process.
- **State** The state S_t is generated by the environment. It changes over time according to the dynamics within the environment.
- **Action** An action A_t is a cause that might change the state of the environment. Actions are produced by the agent.
- **Reward** The reward R_t is a scalar value that is produced by the environment and received by the agent.
- **Agent** The agent is a instance which generates actions and observes the caused change of the state of the environment.
- **Policy** A policy $\pi(A_t|S_t)$ is a probability distribution over the set of possible actions at a time step. A agent is essentially defined by its policy as each action that is taken is sampled from that policy.

All signals in this model are time dependent. Such a model satisfies the Markov Property if next s' and reward r only depend on the current action-state tuple (s, a). If assuming finite state and action spaces together with the Markov Property gives a Finite Markov Decision Process. The environment dynamics can therefore be represented by the bivariate probability distribution

$$p(s', r|s, a) = Pr\{R_{t+1} = r, S_{t+1} = s'|S_t = s, A_t = a\} \tag{1.1}$$

Further, if A_t is sampled from $\pi(A_t|S_t)$ the Markov Property induces conditional independence of (S_{t-1}, R_{t-1}) and (S_{t+1}, R_{t+1}) given S_t.

The agents task is to predict a policy that maximizes the expected future rewards. This objective is given by

$$\arg\max_{\pi} \mathop{\mathbb{E}}_{p_\pi} \left[\sum_{t=0}^{T} R_t | S_0 \right] \tag{1.2}$$

Here T marks the time limit of the process and p_π represents the environment dynamics following a action history sampled from π.

1.1.1 Value functions

Most methods of solving eq. (1.2) use so estimations of so called value functions. This are functions that provide a quality measure for an agent evaluating a state-action tuple.
Commonly two value functions are used. The state-value function and the action-value function. They both depend on all future rewards

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}. \tag{1.3}$$

Here γ is referred to as the discount factor. Its value usually determines how prospective future rewards are weighted in the value function. E.g if $\gamma < 1$ rewards that are closer to t get a higher weight than those that are occuring at a later time. For $\gamma > 1$ the contrary holds.
The state value function is defined by

$$v_\pi(s) = \mathop{\mathbb{E}}_{p_\pi} [G_t | S_t = s] \tag{1.4}$$

and the action value by

$$q_\pi(s, a) = \mathop{\mathbb{E}}_{p_\pi} [G_t | S_t = s, A_t = a] \tag{1.5}$$

it follows

$$v_\pi(s) = \mathop{\mathbb{E}}_{a \sim \pi} [q_\pi(s, a)] \tag{1.6}$$

Usually the objective is to maximize either one or both of the value functions. Note that, $\max_\pi v_\pi(s)$ and $\max_\pi q_\pi(s, a)$ satisfy Bellman's principle of optimality. Hence they can be solved exactly by Dynamic Programming. This referred to as the tabular solution. However for most problems this is not feasible and the value functions are approximated by neural networks.

1.1.2 Q-learning

Q-learning is a method to find the action-value maximizing policy by using temporal differences. This often the backbone of policy gradient algorithms. Let

$$\pi(\cdot|s) = \underset{a}{\text{softmax}}\, q_\pi(s,a) \tag{1.7}$$

then the objective is to approximate q_π which is achieved by the temporal difference loss

$$\mathcal{L}_{TD} = \left[R_{t+1} + \gamma \max_a q_\pi(S_{t+1},a) - q_\pi(S_t,a_t)\right]^2 \tag{1.8}$$

this kind of approximation is usually referred to as one step TD method. The optimality follows directly and the convergence of the approximation under certain conditions has been proven in [2].
In contrast to one step TD methods there are Monte Carlo methods which collect the loss over whole episodes where one episode is defined by the history of temporal differences between a starting state and an end state. This methods are often solved by eligibility traces [3].
Optimizing eq (1.8) is referred to as on-policy policy optimization where the target policy π is the policy which is used when actions are sampled. This however is problematic as the target policy is defined by eq (1.7) depending on q_π which is not trustworthy as this is the function that is to be approximated. In order to have more control over the sampling of actions which is usually referred to as exploration a data collection policy $\mu(a|s)$ is used. Therefore in an off-policy setting eq (1.8) becomes

$$\mathcal{L}_{TD} = \left[R_{t+1} + \gamma \max_a q_\mu(S_{t+1},a) - q_\mu(S_t,a_t)\right]^2. \tag{1.9}$$

and during inference

$$\bar{\pi}(\cdot|s) = \underset{a}{\text{softmax}}\, q_\mu(s,a) \tag{1.10}$$

is used. There are many solutions to overcome the distribution mismatch between π and μ. Many use importance sampling or variance reduction techniques.[4]

1.1.3 Policy gradient methods

This class of methods optimizes a policy $\pi_\theta(a|s)$ directly with respect to its parameters θ. Let

$$\rho(\pi) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t | S_0, \pi\right] \tag{1.11}$$

be the expected, discounted future reward per step and let

$$d_\pi(s) = \sum_{t=0}^{\infty} \gamma^t Pr\left\{S_t = s | S_0, \pi\right\} \tag{1.12}$$

be the discounted stationary distribution of states under π. Then

$$\frac{\partial \rho_\pi}{\partial \theta} = \sum_s d_\pi(s) \sum_a \frac{\partial \pi(a|s)}{\partial \theta} \bar{q}_\pi(s,a) \tag{1.13}$$

Is the policy gradient with which gradient ascent on the policy can be performed in order to maximize ρ. A proof and a thorough discussion can be found in [5]. Note that the policy gradient is on-policy and that $\bar{q}_\pi$ is an approximation of q_π.
Since π is a probability distribution it follows $\frac{\partial \pi(a|s)}{\partial \theta} = 0, \forall s \in S$. Therefore

$$\frac{\partial \rho_\pi}{\partial \theta} = \sum_s d_\pi(s) \sum_a \frac{\partial \pi(a|s)}{\partial \theta} \left[\bar{q}_\pi(s,a) + b(s)\right], \qquad b : S \to \mathbb{R} \tag{1.14}$$

the function b is called a baseline and is often used to reduce variance and bias of the gradient ascent update step. Using $\frac{\nabla_\theta \pi(a|s)}{\pi(a|s)} = \nabla_\theta ln(\pi(a|s))$ and $\mathbb{E}_{x\sim p(x)}[f(x)] = \sum_x p(x) f(x)$, rewriting eq (1.13) yields

$$\frac{\partial \rho_\pi}{\partial \theta} = \mathop{\mathbb{E}}_{\substack{s\sim d_\pi(s) \\ a\sim\pi(a|s)}} \left[\nabla_\theta ln(\pi(a|s)) \left[\bar{q}_\pi(s,a) + b(s)\right]\right] \tag{1.15}$$

In practice where there are large state and action spaces the expectations w.r.t s and a become infeasible to obtain. Using $\mathbb{E}_{x\sim p(x)}[f(x)] = \frac{1}{n}\sum_n f(x), n \to \infty, x \sim p(x)$, ample based learning uses enough samples of s and a in order to obtain a good enough approximation of the expectations. Therefore eq. (1.15) becomes

$$\frac{\partial \rho_\pi}{\partial \theta} = \frac{1}{n} \sum_n \left[\nabla_\theta ln(\pi(a|s)) \left[\bar{q}_\pi(s,a) + b(s)\right]\right], \qquad s \sim d_\pi(s), \quad a \sim \pi(a|s), \quad n \to \infty \tag{1.16}$$

This leads to Actor Critic methods (A2C) where there are two instances that are updated in a turn based fashion. The critic is the value-function approximation and the actor is approximating the policy. Intuitively the critic evaluates the action taken by the actor who uses this evaluation to scale its gradient update (this is the role of $\bar{q}_\pi$ in eq (1.13)).

1.1.4 Soft Actor-Critic

This algorithm was introduced by [6].

1.1.5 Graph convolutional neural networks

1.1.6 Mutex watershed

1.1.7 Principal component analysis

1.1.8 Loss functions

Bibliography

[1] R. S. Sutton and A. G. Barto, “Reinforcement learning:an introduction,” 2015.

[2] ——, “Reinforcement learning:an introduction,” 2015, pp. 74–121.

[3] ——, “Reinforcement learning:an introduction,” 2015, pp. 157–194.

[4] Q. Liu, L. Li, Z. Tang, and D. Zhou, “Breaking the curse of horizon: Infinite-horizon off-policy estimation,” 2018.

[5] R. Sutton, D. Mcallester, S. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” *Adv. Neural Inf. Process. Syst*, vol. 12, 02 2000.

[6] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, “Soft actor-critic algorithms and applications,” 2018.