Machine Learning
Comparison of Supervised Learning and Reinforcement Learning

Liu Yihao
Wu Guangzheng
Jiang Yicheng

December 15, 2017

Abstract

Machine learning is the core of Artificial Intelligence. It enables a machine to learn from and make predictions on given data. The concept was first introduced by Arthur Samuel in 1959 and it has been developing about 50 years. Main approaches of Machine learning includes Supervised Learning, Unsupervised Learning, and Reinforcement Learning. In this article, we will mainly focused on superviced learning and reinforcement learning. We will use the example of AlphaGo to do some comparison of these two approaches.

Key Words: Machine Learning, Artificial Intelligence, Supervised Learning, Reinforcement Learning, AlphaGo

1 Introduction

Artificial intelligence (AI) is intelligence displayed by machines, in contrast with the natural intelligence displayed by humans and other animals. AI techniques have become an essential part of the technology industry, helping to solve many challenging problems in computer science. And machine learning plays the most important role in the process of a machine's finding the best solution to a certain problem. The main approaches of machine learning includes supervised learning, unsupervised learning, and reinforcement learning. We will use the example of AlphaGo and focus on supervised learning and reinforcement learning to see which approach has a better prospects for development.

2 Background

2.1 Supervised Learning

Supervised learning is an approach to build a function to match the input data and output result by learning a lot of given training data.

Give a simple example. When we are born, we did not have any concept of this world. We did not know what a bird is, what an air plane is or what a rocket is. And our parents or teacher or some other more experienced people told us and helped us build some concept of these objects. As time passing by, when we had been given enough samples, we would be able to distinguish them by ourselves. We gain this ability by learning from the experience of other people. And such a learning approach is called supervised learning.

The main process of supervised learning consists of:

1. Determine the type of training examples. (How to distinguish different kinds of object that can fly)

2. Gather a training set. (Given names and some corresponding examples for some kinds of objects)

3. Determine the input feature representation of the learned function. (How we are shown the names and corresponding examples)

4. Determine the structure of the learned function and corresponding learning algorithm. (We need a map between a given concrete objects to some abstract names)

5. Evaluate the accuracy of the learned function. (Whether our map is correct or not)

2.2 Reinforcement Learning

Whether we can learn how to distinguish obejcts that can fly if no one give us any examples? Of course we can. No one taught the first man who gave name "bird" to a bird. So this leads to another way of machine learning — reinforcement learning. We do not know the map from objects to their names and we even do not know the names. What we know is that we need to distinguish these objects according to their own feature and we want to minimize the overlapping. In formal words, reinforcement learning is a process for a machine to find what actions it ought to take in an environment so as to maximize some notion of cumulative reward.

The main process of reinforcement learning consists of:

1. Given a model of the environment without an analytic solution (We need to distinguish objects but we do not know the classification)
2. Given a simulation model of the environment (We are given those objects that we need to distinguish)
3. Collect information about how the environment interacts with it. (Whether the classification we find is perfect enough)

Backgrounds: [6]

1. Markov Decision Process:

Every problem in the world is based on some world view. Artificial Intelligence is also based on some world views - the Classical Physics. Some assumptions are made that we do not consider the effect of the Quatum Mechanics and the Theory of Relativity. Then the Markov Decision Process is raised to specify the world view of Artificial Intelligence.

According to the Markov Decision Process(MDP), we cut the world into small pieces, and put them in a set, like $\{s_0, a_0, r_0, s_1, a_1, r_1, \cdots, s_t, a_t, r_t\}$, where s represents the "state", which describes what our agent is currently, and a represents the action and r represents the reward. In a state s, if you give an action a, you will be sure to go to some other state $s^{'}$, not randomly, and receive some certain reward.

MDP here also made an assumption that the future can only be determined by the instant, and we do not need to care about the past. This is similar to the Wise-Person assumption made hundreds of years ago that if there is a person who knows exactly everything in the universe, he will know what these things will be like in the next second. In the MDP, it is similar. We also suppose that every state is fully observed, and we get the following equation.

$$\mathbf{P}(S_{t+1} \mid S_t) = \mathbf{P}(S_{t+1} \mid S_t, S_{t-1}, \cdots S_0)$$

Note that not only the MDP world view for an agent is that future can only be determined by instant, but also the world should be fully observed. If not, some things out of the observation may cause some effects on the observed world and make it not Markov.

2. Result:

The result of the MDP is then be defined. Since reward is representing the reward for some single time step, result is representing a sequence of reward, with the following equation.

$$G_t = R_{t+1} + \lambda R_{t+2} + \cdots = \sum_{t=0}^{\infty} \lambda^k R_{t+k+1}$$

Where the λ is called discount factor, mainly in the range of $[0, 1]$. It is similar to our experience in daily life. For example, we'd like to enjoy the love as Romeo and Juliet's, but we should pay more attention on the bread for tomorrow's breakfast. If we take

a larger discount factor, we are looking more for the future, while if we take a smaller discount, we are looking for the current situation. In addition, if we take the discount factor as zero, it is just the greedy algorithm.

Then we have the value function, which is representing the expectation of the results.

$$v(s) = \mathbb{E}[G_t \mid S_t = s]$$

With the definition of value function, we have two ways to improve the current policy then

1. Improve $\pi(a \mid s)$ or $a = \pi(s)$ directly to have a higher reward.
2. Getting the improved policy by the estimated value function.

Frankly speaking, the value function is just formulating how human beings make decisions.

3. Bellman Function and Action-Value Function:

Since we get the value function, we'd like to deal with it. Another way to construct a value function is using the recursive, which we call the recursive function Bellman function.

$$v(s) = \mathbb{E}[G_t \mid S_t = s] = \mathbb{E}[R_{t+1} + \lambda G_{t+1} \mid S_t = s] = \mathbb{E}[R_{t+1} + \lambda v(S_{t+1}) \mid S_t = s]$$

From the function, we can conclude that the value of current state is determined by the current reward and the value of next state. But however, there's still some disadvantage of the value function. Bellman function cares nothing other than the states, but agents prefer knowing the value function of some actions instead of the states. Based on these requirements, the action-value function is then introduced as

$$Q^{\pi}(s, a) = \mathbb{E}[r + \lambda Q^{\pi}(s^{'}, a^{'}) \mid s, a]$$

And it is easy to see that our task is to fing the optimal value function:

$$Q^{*}(s, a) = max_{\pi} Q^{\pi}(s, a)$$

And with the equation mentioned above, we have

$$Q^{*}(s, a) = \mathbb{E}_{s'}[r + \lambda max_{a'} Q^{*}(s^{'}, a^{'}) \mid s, a]$$

Policy Iteration:

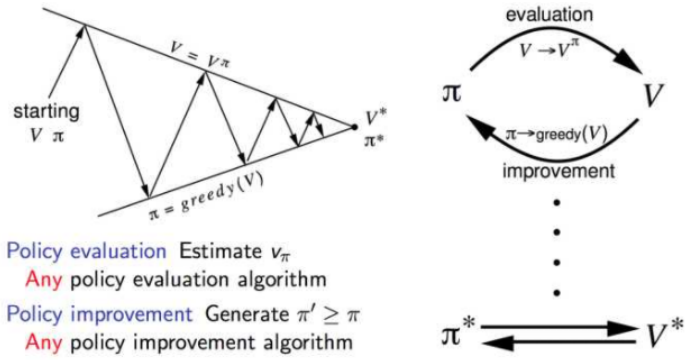
The idea of policy iteration is to deal with the value function recursively to let the policy converge to the optimal. We define the policy iteration equation with the Bellman function:

$$v_{k+1}(s) = \mathbb{E}[R_{t+1} + \lambda v(S_{t+1}) \mid S_t = s] = \sum_{a} \pi(a \mid s) \sum_{s^{'}, r} p(s^{'}, r \mid s, a)[r + \lambda v_k(s^{'})]$$

So there are two steps in value iteration:

1. Policy Evaluation: Estimate v_π.
2. Policy improvement: Generate $\pi^{'} \leq \pi$.

The following figure shows how these steps work.



And we have the following algorithms.

Algorithm 1: Policy Iteration - Initialization

Input : A sequence of states S and an initial policy π
Output: Initialization

Function Initialization(S, π)**:**
    $V(s) \in \mathbb{R}$ and $\pi(s) = \mathcal{A}(s)$ arbitrarily for all $s \in S$.
    PolicyEvaluation(S, π)
end

Algorithm 2: Policy Iteration - Policy Evaluation

Input : Initialized Policy Iteration
Output: Estimated v_π

Function PolicyEvaluation(S, π)**:**
    while $\Delta \leq \theta$ **do**
        $\Delta \leftarrow 0$
        for each $s \in S$ **do**
            $v \leftarrow V(s)$
            $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s))[r + \lambda V(s')]$
            $\Delta \leftarrow max(\Delta, |v - V(s)|)$
        end for
    end while
    PolicyImprovement(S, π)
end

Algorithm 3: Policy Iteration - Policy Improvement

Input : Estimated Policy
Output: The Optimal Policy

Function PolicyImprovement(S, π)**:**
  $policy - stable \leftarrow True$
  for each $s \in S$ **do**
    $a \leftarrow \pi(s)$
    $\pi(s) \leftarrow argmax_a \sum_{s',r} p(s', r \mid s, \pi(s))[r + \lambda V(s')]$
    if $a \neq \pi(s)$ **then**
      $policy - stable = False$
    end if
  end for
  if $policy - stable$ **then**
    return V and π
  end if
  else
    PolicyEvaluation(S, π)
  end if
end

However, the policy iteration is based on the model given before the initialization, so it cannot be applied directly. You can use it only based on some models. So we make some improvement on it.

Value Iteration:

Value Iteration is more direct, which focus directly on the Bellman Optimal function. So we have the following equation.

$$v_*(s) = max_a \sum_{s',r} p(s', r \mid s, a)[r + \lambda v_*(s')]$$
$$v_{k+1}(s) = max_a \sum_{s',r} p(s', r \mid s, a)[r + \lambda v_k(s')]$$

And we have the value iteration algorithm shown as follow.

Algorithm 4: Value Iteration

Input : A sequence of states S and an initial policy π
Output: The Optimal Policy

Function ValueImprovement(S, π)**:**
  Initialize array V arbitrarily. (e.g. $V(s) = 0$ for all $s \in S^+$)
  while $\Delta \geq \theta$ **do**
    $v \leftarrow V(s)$
    $V(s) \leftarrow max_a \sum_{s',r} p(s', r \mid s, a)[r + \lambda V(s')]$
    $\Delta \leftarrow max(\Delta, |v - V(s)|)$
  end while
  Output a deterministic policy, π, such that
  $\pi(s) = argmax_a \sum_{s',r} p(s', r \mid s, a)[r + \lambda V(s')]$
end

The difference of value iteration and policy iteration is that for the policy iteration, it estimates to renew the value and use the value to find the policy. But value iteration doesn't find some improved policy, but continue updating the value until it converges, and just find the optimal policy by the converged value. The same problem they share is that they rely heavily on the model and can be applied in some limited areas. So some improvements are shown below.

3 Case of AlphaGo

In October 2015, AlphaGo became the first computer Go program to beat a human professional Go player without handicaps on a full-sized 19×19 board. [1] In March 2016, it beat Lee Sedol in a five-game match, giving a final score of 4 games to 1, the first time a computer Go program has beaten a 9-dan professional without handicaps. [2] At the 2017 Future of Go Summit, AlphaGo beat Ke Jie, the world No.1 ranked player at the time, in a three-game match. This AlphaGo learned from all match in human's history and it can always choose the best position to put its go.

While such a great machine was beat by AlphaGo Zero (another machine trained by reinforcement learning), giving a final score of 0 to 100. AlphaGo Zero is trained by self-play reinforcement learning, starting from random play, without any supervision or use of human data. And it is able to beat the previous AlphaGo after three days self-learning.

4 Discussion

Learning from this case and the literature [3], we know that "AlphaGo Zero outperformed AlphaGo Lee after just 36h. In comparison, AlphaGo Lee was trained over several months." " AlphaGo Zero used a single machine with 4 tensor processing units (TPUs), whereas AlphaGo Lee was distributed over many machines and used 48 TPUs." So we can see that the machine trained with reinforcement learning works much better than that with supervised learning. And we think that this is because reinforcement learning is more creative.

Learning from experience is a good way to do improvement. And humans have been familiar with such kind of learning modes for a quite long time. In our life, we can gain

new knowledge from teacher or any more experienced people. Also, we can learn from books written by previous humans. We keep following this way since we can share our experience with others and we can find a better way to solve some problems.

But a machine is different from humans. We are not only able to gain knowledge and experience, but also find some new methods. A machine trained in such way cannot make any innovation. Only what is can do is summarize the experience and find the best way to solve some problems using those ways that already exist. But this may not be the best way in the world. And sometimes human's experience may be wrong and mislead a machine.

While for a machine trained with reinforcement learning, no human's experience can help it. It can only gain knowledge and summarize experience by itself. Due to its strong computing ability, it can improve itself very soon. Just take AlphaGo Zero for an example, "Humankind has accumulated Go knowledge from millions of games played over thousands of years, collectively distilled into patterns, proverbs and books. In the space of a few days, starting tabula rasa, AlphaGo Zero was able to rediscover much of this Go knowledge, as well as novel strategies that provide new insights into the oldest of games." [3] Some experience from human may not be effective for a machine to gain useful experience. These data will not only waste machine's time, but also influence the analysis result of it. In a word, human's experience sometimes can help a machine, but it will influence its efficiency and give it a limitation on final results. So it is not surprising that AlphaGo Zero can beat previous AlphaGo easily after a short time self-learning.

5 Conclusion

We humans have experienced the same cases, starting from zero, trying by ourselves and summarizing experience to make better choices. However, we humans are not always strong enough to be able to start from zero and reach further achievement. We have to learn from previous people and stand on the shoulders of giants to go further. And since we can think by ourselves, we can make creation with previous experience. While for a machine, it is strong enough to explore by itself start from zero. Without the experience from humans, it can also reach the same and even more achievement. Maybe for a machine, reinforcement learning is a more approriate way for itself to do explorement.

References

[1] "Research Blog: AlphaGo: Mastering the ancient game of Go with Machine Learning". Google Research Blog. 27 January 2016.

[2] "Match 1 – Google DeepMind Challenge Match: Lee Sedol vs AlphaGo". 8 March 2016.

[3] "Mastering the game of Go without human knowledge", David Silver, Julian Schrittwieser, Demis Hassabis. Nature 550, 354359.18 October 2017.

[4] Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).

[5] Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." Nature 518.7540 (2015): 529-533.

[6] David Silver's Lecture Slides and Lectures `http://www0.cs.ucl.ac.uk/staff/d.silver/web/Home.html`