Lecture 1: Reinforcement Learning: What and Why?

Melih Kandemir

Semester: Fall 2021

1 Introduction

A main property that characterizes a machine learning problem is how the target task is conveyed to the algorithm. In supervised learning, labels associated with provided inputs determine the target task. Reinforcement Learning (RL) studies algorithms that train a computer to perform a target task by supervising it via rewards. In its simplest case, rewards could be only telling right or wrong. Usually, rewards provide weak feedback, hence make the learning problem harder. See Lecun‘s cake for an interesting analogy. Sometimes it is much easier to define tasks by rewards than labels. A simple example is recommender systems.

RL is the counterpart of teaching an animal by feeding in machine learning. RL has its roots in behavioral biology, which relates the self rewarding mechanism of organisms to the fundamental principle of life. The organisms incline to accumulate information about their environment, as a side effect of their resistance against entropy [1]. Reinforcement learning is a fundamental element of machine learning research and its prominence is likely to improve. There is an ongoing discussion on whether all machine learning tasks can be explained as special cases of reinforcement learning, hence would it be sufficient to concentrate research only on machine learning [2].

RL is an essential ingredient for dexterous robot control. Using RL, it is possible to train a robot to sort fish into boxes and play human-level table tennis. Training robots with such complex skills is essential for future technologies. We are in the process of transitioning from a society of industrial robots to personal robots. This process follows a similar pattern to the history of computers: Industrial computers $\rightarrow$ personal computers $\rightarrow$ mobile computers $\rightarrow$ micro computers.

2 Supervised learning as empirical risk minimization

Let us refresh our memories with the Supervised Learning (SL) setup and then compare it to RL to highlight what is new in this course. Consider a data set $\mathcal{D} = \{(s_1, a_1), (s_2, a_2), \ldots, (s_N, a_N)\}$ consisting of N input observations $s_* \in \mathbb{R}^D$ of D dimensions with corresponding labels $a_* \in \mathcal{A}$. We are interested in finding a function that maps observations to labels $f : \mathcal{S} \rightarrow \mathbb{R}$. It is supervised learning when we restrict our solution into a parameterized set of functions $f \triangleq f_\theta(s_*)$ and fit its parameters to data with respect to least squares:

$$\widehat{\theta} \triangleq \underset{\theta}{\text{minimize}} \ \frac{1}{N} \sum_{i=1}^{N} (a_i - f_\theta(s_i))^2. \tag{1}$$

It is Deep Learning (DL) when $f_\theta(s_*)$ is chosen to be a neural network with many hidden layers. The optimization problem in Eq. 1 makes intuitive sense. It seeks for a θ that minimizes the average error between the observed label a_i and the label the mapping, or the model, predicts from the input observations. This intuitive solution stems from the following modeling assumptions:

i) Input observations s and labels a are random variables following an unknown distribution $p(s, a)$.

ii) The data set $\mathcal{D}$ contains N independent and identically distributed (i.i.d) samples from this distribution, i.e. $s_i, a_i \sim p(s, a), \quad i = 1, \ldots, N$. We can view $\mathcal{D}$ also as a random variable that follows $p(\mathcal{D}) = \prod_{i=1}^{N} p(s_i, a_i)$.

iii) For a given predictor $\widehat{a}$, define risk as $\mathcal{L}(a, \widehat{a}) \triangleq (a - \widehat{a})^2$.

Then for $\widehat{a} \triangleq f_\theta(s)$, the expected value of the risk $\mathcal{L}$ with respect to the data generating distribution $p(s,a)$ is

$$R_\theta = \int\int (a - f_\theta(s))^2 p(s,a) ds da = \mathbb{E}_{p(s,a)} \mathcal{L}(a, f_\theta(s)). \tag{2}$$

We are interested in choosing θ that would minimize this risk, i.e. $\widehat{\theta} \triangleq \underset{\theta}{\text{minimize}} \; R_\theta$. Remember that we do not know $p(s,a)$, hence we cannot evaluate R_θ. We need to estimate our target quantity R_θ, called the *estimand*, by another quantity that is a function of observations obtained from $p(s,a)$, called the *estimator*. For the given sample set $\mathcal{D}$, the *sample mean estimator* of R_θ is defined as

$$\widehat{R}_\theta \triangleq \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(a_i, f_\theta(s_i)). \tag{3}$$

Minimizing this estimator yields the formula in Eq. 1. Note that as $\mathcal{D}$ is a random variable, so is $\widehat{R}_\theta$. Define the *bias* of an estimator as the expected value of its divergence from the estimand $B(\widehat{R}_\theta) \triangleq \mathbb{E}_{\widehat{R}_\theta}[\widehat{R}_\theta - R_\theta]$. Let us next see how much bias $\widehat{R}_\theta$ has

$$\begin{aligned}
B(\widehat{R}_\theta) &= \mathbb{E}_{\widehat{R}_\theta}[\widehat{R}_\theta - R_\theta] & (4)\\
&= \mathbb{E}_{\widehat{R}_\theta}[\widehat{R}_\theta] - R_\theta & (5)\\
&= \mathbb{E}_{p(s,a)}[\widehat{R}_\theta] - R_\theta & (6)\\
&= \mathbb{E}_{(a_i,s_i)\sim p(s,a)}\Big[\frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(a_i, f_\theta(s_i))\Big] - R_\theta & (7)\\
&= \frac{1}{N}\sum_{i=1}^{N} \underbrace{\mathbb{E}_{(a_i,s_i)\sim p(s,a)}[\mathcal{L}(a_i, f_\theta(s_i))]}_{R_\theta} - R_\theta & (8)\\
&= 0. & (9)
\end{aligned}$$

We have thus shown that $\widehat{R}_\theta$ is an *unbiased estimator* of R_θ. Hence we can safely use the estimator in place of the estimand without worrying about anything other than sampling error, i.e. the information we miss because we can have a data set with only finite number of observations. This problem setup is central to the commonplace use of machine learning and is referred to as *Empirical Risk Minimization (ERM)*.

3 Challenges of reinforcement learning

Consider the cart pole balancing problem, where a cart carrying an unactuated pole floats on a straight horizontal track. The cart is actuated by a torque applied either to the right or the left direction. See here for a real cart pole system and its simulation here on a simplified environment, which we will take as basis. How would the aforementioned ERM setup apply to this case?

Let us give a try. In the simple synthetic case, the cart pole lives in an environment where only the cart may move on the track and the pole may move on the cart. Hence let us define the input observation for a time step t to be $s_t = (c_t, \dot{c}_t, \alpha_t, \dot{\alpha}_t)$, where

- c_t is the position of the cart,
- c_t is the velocity of the cart ,
- α_t is the angle of the pole with respect to the cart normal in degrees,
- $\dot{\alpha}_t$ is the angular velocity of the pole.

These observations characterize the *state* of the environment. We can apply a fixed amount of torque on the cart either in the left direction, $a_t = -1$, or the right direction, $a_t = +1$. Our task is to learn a *control policy* that maps states to actions $a_t = \pi_\theta(s_t)$ in such a way that the pole is balanced upright on the cart. At $t = 0$, we have $s_0 = (0, 0, 180, 180)$. We aim to reach $s_{t^*} = (\cdot, 0 \pm 0.05, 0 \pm 0.05, 0 \pm 0.05)$ for $t^* < 200$ steps. Our initial data set is $\mathcal{D} = \{(s_0, ?)\}$. How do we proceed? We need to collect data from the environment to fit θ.

Try 1. We can generate random actions $a_0, \ldots, a_{200} \sim Bernoulli(0.5)$ and observe the corresponding states $s_1, \cdots, s_{200}$. This would give a data set on which one can fit θ, but this is not the control policy we want, as it would almost never bring us to the target state $s_{t^*} = (\cdot, 0 \pm 0.05, 0 \pm 0.05, 0 \pm 0.05)$. We still do not know how to keep the pole upright!

Try 2. We can find an expert E to show us how to balance the cart pole in a round of 200 time steps, called an *episode*. We can record the states s_t^E of the cart pole and the corresponding expert action a_t^E at each time step t, store all (s_t^E, a_t^E) pairs into $\mathcal{D}_E = \{(s_1^E, a_1^E) \cdots, (s_{200}^E, a_{200}^E)\}$, and fit a model as in Eq. 3. This is called *behavioral cloning*, which is the simplest way of *imitation learning* or *learning from demonstration*. This could indeed work when collecting sufficient data from an expert on the target task is affordable. There is still a problem. In the supervised learning example in Eq. 1, the subscripts were only indexing the observations. Here, they indicate time, which has an influence on the observations. That is, s_t^E is expected to be more correlated with s_{t+1}^E than with s_{t+30}^E. The data set does not contain i.i.d. distributed observations, hence $\widehat{R}_\theta$ is no longer an unbiased estimator of R_θ.

Try 3. Let us still assume we are given $\mathcal{D}_E$ from an expert E and try to fix how the model processes data. Consider a predictor model as follows:

$$h_t = \tanh(W_h h_{t-1} + W_s s_t), \tag{10}$$

$$\widehat{a}_t = \tanh(W_a h_t), \tag{11}$$

where $\tanh(u) = (e^{2u} - 1)/(e^{2u} + 1)$, $\widehat{a}_t$ are the predicted outputs, and $\{W_h, W_s, W_a\}$ are free parameters to be fit by solving the optimization problem below

$$\underset{W_h, W_s, W_a}{\text{argmin}} \frac{1}{200} \sum_{t=1}^{200} (a_t - \widehat{a}_t)^2. \tag{12}$$

Assume we are given h_0 and denote $L_u = \frac{1}{200} \sum_{t=u}^{200} (a_t - \widehat{a}_t)^2$, let us spell out the loss function above

$$\begin{aligned} L_1 &= (a_1 - \tanh(W_a \tanh(W_h h_0 + W_s s_1)))^2 + L_2. \qquad (13) \\ &= (a_1 - \tanh(W_a \tanh(W_h h_0 + W_s s_1)))^2 + (a_2 - \tanh(W_a \tanh(W_h \tanh(W_h h_0 + W_s s_1)) + W_s s_2))^2 + L_3 \\ &\vdots \end{aligned}$$

This loss function requires a recursion of 200 levels, which is computationally equivalent to building a neural network with 200 hidden layers. This is called a *Recurrent Neural Network (RNN)*. RNNs are useful modeling tools for dynamics modeling problems including reinforcement learning. What could be the disadvantages of building such a deep computation stack? Are we fine if we have a strong enough computer?

What happens if we do not have an expert to demonstrate us how to balance the cart pole? Then the model needs learn a policy $\widehat{a}_t = \pi(s_t)$ from its own interactions with the environment. Then the model needs to get feedback to know what we want it to do. This feedback is a reward r_t. Now we have three main ingredients that make a reinforcement learning setup:

- **States** s_t that describe the situation of the environment the agent lives in,
- **Actions** a_t the agent can take to solve the task at hand, i.e. to balance the pole
- **Rewards** r_t that describe whether the agent is fulfilling the task expected from it.

Learning a policy $\widehat{a}_t = \pi(s_t)$ that can maximize the cumulative reward $r_1 + r_2 + \ldots$, called the *return*, from sequences of state-action-reward triples $(s_1, a_1, r_1), (s_2, a_2, r_2) \ldots$ is called reinforcement learning.

Take home. The RL setup differs from standard Supervised Learning (SL) in the following ways:

- The SL model makes predictions passively about fixed properties of an environment. The RL model interacts with the environment by taking *actions*, which change the future behavior of the environment.

- SL expresses a target task via labels of individual observations in a training set. RL expresses a target task by rewarding the desired environment state.
- The goal of SL is to minimize the risk of wrong predictions. The goal of RL is to maximize the *cumulative reward* throughout the course of environment interactions.
- RL data is non-i.i.d. Present decisions affect future observations. The order of the observations matters.
- Reward is usually a sparse signal. Informative feedback could be delayed. In car driving, keeping the lane is immediate reward. In board game playing, winning is delayed reward.
- Rewarding subtasks to solve the delayed reward problem is risky. The model might choose to collect only subtask rewards.

References

[1] K. Lorenz. *Behind the Mirror: A Search for a Natural History of Human Knowledge.* Helen and Kurt Wolff Books, 1978.

[2] D. Silver, S. Baveja, D. Precup, and R. Sutton. Reward is enough. *Artificial Intelligence*, 299, 2021.