Reinforcement Learning – Exercises Lectures 1-5

Maurice Frank
11650656
maurice.frank@posteo.de
Code: github

September 18, 2019

Lecture 0:

0.1 Linear algebra and multivariable derivatives

1.

$$AB = \begin{bmatrix} a_{11} & 0 \\ 0 & a_{22} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \tag{1}$$

$$= \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} \\ a_{22}b_{21} & a_{22}b_{22} \end{bmatrix} \tag{2}$$

$$AB^T = \begin{bmatrix} a_{11} & 0 \\ 0 & a_{22} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{21} \\ b_{12} & b_{22} \end{bmatrix} \tag{3}$$

$$= \begin{bmatrix} a_{11}b_{11} & a_{11}b_{21} \\ a_{22}b_{12} & a_{22}b_{22} \end{bmatrix} \tag{4}$$

$$d^T B d = \begin{bmatrix} d_1 & d_2 \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} \tag{5}$$

$$= d_1^2 b_{11} + d_1 d_2 b_{12} + d_1 d_2 b_{21} + d_2^2 b_{22} \tag{6}$$

2.

$$A^{-1} = \begin{bmatrix} a_{11}^{-1} & 0 \\ 0 & a_{22}^{-1} \end{bmatrix} \tag{7}$$

$$B^{-1} = \frac{1}{b_{11}b_{22} - b_{21}b_{12}} \begin{bmatrix} b_{22} & -b_{21} \\ -b_{12} & b_{11} \end{bmatrix} \tag{8}$$

3.

$$\frac{\partial c}{\partial x} = \begin{bmatrix} -2x \\ \frac{1}{yx} \end{bmatrix} \tag{9}$$

$$\frac{\partial c}{\partial e} = \begin{bmatrix} -2x & 1 \\ \frac{1}{yx} & -\ln(x)y^{-2} \end{bmatrix} \tag{10}$$

4.

$$f(\boldsymbol{x}) = \sum_{i}^{N} i x_i \tag{11}$$

$$\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} = [1, \ldots, N] \tag{12}$$

0.2 Probability theory

1.

$$\mathbb{E}[X + \alpha Y] = \mathbb{E}[X] + \alpha \mathbb{E}[Y] \tag{13}$$

$$= \mu + \alpha \nu \tag{14}$$

2.

$$\operatorname{Var}[X + \alpha Y] = \operatorname{Var}[X] + \alpha^2 \operatorname{Var}[Y] + 2\alpha \operatorname{cov}[X, Y] \tag{15}$$

3.

At last we have σ^2 which is just the variance of the noise in the measurements. This is model independent (we can not train it away). The bias term tells us how good our estimator estimates the sample data points. The estimator variance tells us how jumpy our estimator is. If the model has little parameters ('smooth') it will have high bias but low variance (if regularizes over the evident points but doesn't estimate the data that good anymore). If the model is a complex one with high capacity it will have low bias but high variance.

4.

This is called the bias-variance trade-off as in machine learning we are mostly interested in building a model which has high bias and high variance. The trade-off shows us that these two are opposite in their objective and that optimizing both is not easy.

0.3 OLS, linear projection and gradient descent

We have training set $\boldsymbol{X} \in \mathbb{R}^{n \times m}$ with targets $\boldsymbol{y} \in \mathbb{R}^n$. We have a linear model $f_{\boldsymbol{\beta}}(\boldsymbol{X}) = \boldsymbol{X} \cdot \boldsymbol{\beta}$.

1.

$$\boldsymbol{\beta} \in \mathbb{R}^m$$

2.

$$\begin{aligned}
\hat{\boldsymbol{\beta}} &= \arg\min_{\boldsymbol{\beta}} (\boldsymbol{y} - f_{\boldsymbol{\beta}}(\boldsymbol{X}))^2 \\
\frac{\partial}{\partial \hat{\boldsymbol{\beta}}} (\boldsymbol{y} - f_{\hat{\boldsymbol{\beta}}}(\boldsymbol{X}))^2 &= \frac{\partial}{\partial \hat{\boldsymbol{\beta}}} (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})^2 \\
&= \frac{\partial}{\partial \hat{\boldsymbol{\beta}}} \boldsymbol{y}^2 - 2\boldsymbol{y}\boldsymbol{X}\hat{\boldsymbol{\beta}} + (\boldsymbol{X}\hat{\boldsymbol{\beta}})^2 \\
&= 2\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\beta}} - 2\boldsymbol{y}\boldsymbol{X} \\
&\overset{\text{def}}{=} 0 \\
&\iff \\
\mathbf{X}^T\boldsymbol{X}\hat{\boldsymbol{\beta}} &= \boldsymbol{X}^T\boldsymbol{y} \\
&\iff \\
\hat{\boldsymbol{\beta}} &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}
\end{aligned}$$

3.

$$\epsilon_{\boldsymbol{\beta}} = \boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}$$

4.

5.

6.

7.

Lecture 1: Introduction

1.1 Introduction

1.

The curse of dimensionality are mutliple sad observation called one will make when working with high-dimensional data. In general the problems arise from the fact that the number of value combinations rises exponentially, with the dimension in the exponential. E.g. in hyper-parameter optimization using grid-search the number of needed models to be tested rises exponentially with the number of hyper-parameters. Another example we often see in machine learning with high-dimensional data. With a limited number of trainings samples the distribution of those might be highly sparse in its space akin like a set of dirac functions. Trying to approximate that might be difficult.

2.

(a)

$$\begin{aligned} N_{\text{states}} &= N_{\text{predator states}} \cdot N_{\text{prey states}} \\ &= 5^2 \cdot 5^2 \\ &= 625 \end{aligned}$$

(b) As it is a toroid we just have to remember the differences of the two entities. So the state is just the offset in toroidial coordinates.

(c)

$$\begin{aligned} N'_{\text{states}} &= 5 \cdot 5 \\ &= 25 \end{aligned}$$

(d) The advantage of this approach is that we have fewer states and now multiple states that have to learn the same response to it. Thus we can assume faster training of our predator.

(e) For Tic-Tac-Toe we could reduce the state space by using the point symmetry of the game board. Of all starting states that are symmetric through the center only keep one.

3.

(a) The greedy agent will perform better. Tic-tac-toe is a solved games as such a trained agent can know the perfect move to any situation and no exploration is necessary.

4.

(a) We decrease the exploration probability ϵ each step with a discount factor η. We can write the exploration probability at step ϵ_t with:

$$\epsilon_t = \epsilon \cdot \eta^t$$

(b) No, that method would not work if the opponent changes strategy. It continously decreases exploration over time independent of the game dynamics. We can adapt our strategy by introducing the time step of the last strategy change of the opponent t_{change}. Then we restart from the beginning if the strategy changes:

$$\epsilon_t = \epsilon \cdot \eta^{t - t_{\text{change}}}$$

1.2 Exploration

1.

$$(1 - \epsilon) + \frac{\epsilon}{n}$$

2.

A_3 and A_4. The first one could be greedy as all states have the same average. Same for the second as 2 and 3 have the greedy average. The third action could be greedy as state 2 has the top average then. The next two are suboptimal thus have to be exploration.

3.

$R_0 = -1$ and $R_1 = +1$. By random we choose A_0 in the first step. See the development of the Q-values (bold is the chooses action with greedy policy):

step	Q_0^{pessi}	Q_1^{pessi}	Q_0^{opti}	Q_1^{opti}
0	-5	-5	5	5
1	**-1**	-5	**-1**	5
2	**-1**	-5	-1	**1**
3	**-1**	-5	-1	**1**

4.

The optimistic initalization leads to the higher return ($== 1$) than with the pessimistic initalization ($== -3$). If broken the tie the other way:

step	Q_0^{pessi}	Q_1^{pessi}	Q_0^{opti}	Q_1^{opti}
0	-5	-5	5	5
1	-5	**1**	5	**1**
2	-5	**1**	**-1**	1
3	-5	**1**	-1	**1**

In this case the pessimistic initalization lead to the higher return ($== 3$) than the optimistic initalization ($== 1$).

5.

The optimistic initalization leads to the better estimate of the Q-values.

6.

The optimistic initalization works better for exploration as its basic assumption is that any action could be the best until proven otherwise. As such it will lead to everything being tried using the high initalization values.

Lecture 2: MDPs and dynamic programming

2.1 Markov Decision Processes

1.

(a) Description of the games defined in Section 1.2 in Sutton/Barton:

Game	State Space	Action Space	Reward Signal
Master chess	All possible chess game sequences	Set of single legal moves in chess	
Adaptive petroleum refinery controller	Current state of the refinery (e.g. yield, quality, fillness) + state of the marginal costs	Yield, cost and quality levers	RoI
Newborn gazelle	basically all possible worlds around the gazelle	any physical action possible as a newborn gazelle	health and joy
Trash robot	physical state, power level, history of movement trajectories	possible movements	How much trash taken + not loosing all charge

(b) Another example of an Reinforcement learning application:

Game	State space	Action space	Reward signal
Composing musician	all possible partly composed songs	adding new notes / instruments	beauty of the song

(c) An example for a game that is hard to represent with an MDP is Age of Empires with Fog of War set to on. As the fog of war hides the gambe boards current state to the actor we can not actually build a state space that describes the games state.

(d) One could think of a maze with parts that only can be overcome with enough stamina. In this case stamina would be a state variable.

(e) With these three actions we can reach any state the robot might get into as such they are sufficient actions for the game. The disadvantage though is that they are quite low level actions for the robot. It might be more usefull for the RL algorithm to focus on the high-level control decision. Actually driving the robot from A to B can be solved otherwise.

(f) One could separate the two problems, driving and decision making. Solve both with their own RL algorithm.

2.

(a)

$$G = \sum_{i=0}^{N} \gamma^i \cdot R_i$$

(b)

$$\sum_{k=0}^{\infty} \gamma^k = 1 + \gamma \cdot \sum_{k=0}^{\infty} \gamma^k$$

$$\iff$$

$$\sum_{k=0}^{\infty} \gamma^k - \gamma \cdot \sum_{k=0}^{\infty} \gamma^k = 1$$

$$\iff$$

$$\sum_{k=0}^{\infty} \gamma^k \cdot (1-\gamma) = 1$$

$$\iff$$

$$\sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma}$$

(c) Because we made the task episodic we assumed the run through the maze to be always the same length. Thus our robot does not focus on learning to solve the maze in shorter time lengths.

(d) A discount factor of $\gamma < 1$ would help with this because it would discount the return of solving the maze more the longer the robot needed to exit it. Thus the robot will learn to solve it faster.

(e) We could add a negative reward for each done step (moving is hard!). To avoid the negative reward the robot would learn to do fewer steps to solve the maze.

2.2 Homework: Dynamic Programming

1.

First the stochastic case:

$$\begin{aligned} v^{\pi}(s) &= \sum_{a} \pi(a|s) \sum_{s',r} p(s',r|s,a) \left[r + \gamma \cdot v_{\pi}(s')\right] \\ &= \sum_{a} \pi(a|s) \cdot q^{\pi}(s,a) \end{aligned}$$

and the deterministic policy:

$$\begin{aligned} v^{\pi}(s) &= \arg\max_{a} \pi(a|s) \sum_{s',r} p(s',r|s,a) \left[r + \gamma \cdot v_{\pi}(s')\right] \\ &= \arg\max_{a} \pi(a|s) \cdot q^{\pi}(s,a) \end{aligned}$$

2.

Algorithm 1 Policy Evaluation

```
repeat
    $\Delta \leftarrow 0$
    for all $s \in S$ do
        for all $a \in A(s)$ do
            $q \leftarrow Q(s,a)$
            $Q(s,a) \leftarrow \sum_{s',r} p(s',r|s,a)[r + \gamma \cdot Q(s', \pi(s'))]$
            $\Delta \leftarrow \max(\Delta, |q - Q(s,a)|)$
        end for
    end for
until $\Delta < \theta$
```

3.

Algorithm 2 Policy Improvement

```
policy-stable $\leftarrow$ $true$
for all $s \in S$ do
    old-action $\leftarrow \pi(s)$
    $\pi(s) \leftarrow \arg\max_a Q(s,a)$
    if old-action $\neq \pi(s)$ then
        policy-stable $\leftarrow$ $false$
    end if
end for
```

4.

$$q_{k+1}(s,a) = \sum_{s',r} p(s',r|s,a)\left[r + \gamma \cdot \max_{a'} q_k(s',a')\right]$$

Lecture 3: Monte Carlo methods

3.1 Homework: Monte carlo

1.

(a) With first-visit MC:

$$\begin{aligned} v(s_0) &= \frac{1}{3}[\gamma^2 \cdot 5 + \gamma^4 \cdot 5 + \gamma^3 \cdot 5] \\ &= 3.6585 \end{aligned}$$

(b) With every-visit MC:

$$\begin{aligned} v(s_0) &= \frac{1}{3}[3 \cdot 5 \cdot (1 + \gamma + \gamma^2) + 2 \cdot 5 \cdot \gamma^3 + \gamma^4 \cdot 5] \\ &= 99.873 \end{aligned}$$

2.

The problem with *ordinary importance sampling* in off-policy Monte Carlo is that the variance of its estimation is unbounded. Thus if the variance of the observed return is high (to infinite) the variance of the value estimation also get extremely high. This can considerably slow down convergence of the value estimation.

3.

The problem with *weighted importance sampling* is that its value estimation is biased. **HEEEELLLP????**

Lecture 4: Temporal difference methods

4.1 Temporal difference learning (Application)

1.

(a) TD(0)

$$V(s_{t-1}) \leftarrow V(s_{t-1}) + \alpha \cdot (r_t + \gamma V(s_t) - V(s_{t-1}))$$

t	A	B
0	0	0
1	-0.3	0
2	-0.3	0.37
3	-0.7	0.37
4	-0.893	0.37
5	-0.893	0.3437

(b) 3-step TD **TODO**

t	A	B

(c) SARSA

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

t	A,1	A,2	B,1	B,2
0	0	0	0	0
1	-0.3	0	0	0
2	-0.3	0	0.4	0
3	-0.3	-0.43	0.4	0
4	-0.63	-0.43	0.4	0
5	-0.63	-0.43	0.4	0.1

(d) Q-Learning

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

t	A,1	A,2	B,1	B,2
0	0	0	0	0
1	-0.3	0	0	0
2	-0.3	0	0.4	0
3	-0.3	-0.4	0.4	0
4	-0.53	-0.4	0.4	0
5	-0.53	-0.4	0.4	0.1

2.

A better policy would be to take action 1 for B and action 2 for A. This is just using the max Q values from the Q-learning estimation.

3.

1. The value estimate of π_{random} would reflect the actual correct values while the π_{student} policy would show a skewed image. For B it always picks the better off terminal action while valueing action 1 with zero. For A it would get stuck in a loop of worsening scores.

2. The problem with a fixed random policy is that it does not learn to avoid obvious wrong decision (like 2 for A). The problem with the alternative policy is that it was pre-emptively set from a wrong approximation of the correct values. So it is stuck not discovering correct alternative strategies.

3. An ϵ-greedy would be beneficial in this case as it would allow our policy to make definitly good decisions (like 2 for B) but if set still allows to see alternative options. In this case the student policy would be able to see the value of taking action 1 for A.

4.2 Contraction mapping

1.

2.

4.3 Temporal difference learning (theory)

1.

2.

3.

4.4 Homework: Maximization bias

1.

Q-learning

S \ A	0	1	2	3
A	2	1.5	—	—
B	1	1	2	0

SARSA

S \ A	0	1	2	3
A	2	1.5	—	—
B	1	1	2	0

2.

Both Q-learning and SARSA suffer from Maximization Bias. This bias happens as both learning methods use both a maximum function inside their Q update. For SARSA this comes from the policy (e.g. ϵ-greedy) and in Q-learning the maximum is directly used. The effect is visible when rewards associated with a state have a outlier maximum, then the Q estimation will focus mainly on this maximum. In our case we see the effects of this on the left action (0) of state A. As the maximum of rewards after B is 2 in our observed data we will see this action being biased towards 2 in the beginning of training.

3.

The problem with our argmax methods is that we are using the maximum of our estimates to choose the maximizing action. As we are updating the Q values while using them to find the maximum Q value we introduce the bias that focusses on the maximum. The idea of Double Q-learning now is to have to set of Q values, one to to choose the maximizing action and one to actually determine the value of the maximum. So we have two set of Q values. When we update a value of one of them we use the argmax on those Q values but then retrieve the actual Q value from the second set of Q values.

(Somehow this reminds me of how a Vickroy auction works)

4.

S \ A	0	1	2	3
A	1	1.5	—	—
B	1	1	1	1