NAMA : Iryanto Umpes

KELAS : VISIONER

Asal Universitas : Universitas Negeri Semarang

Remidial Reinforcement Learning/Robotick

Tugas remidialnya adalah,

1. Bikin resume materi RL dari pertemuan 1-6, masing2 per materi 1 halaman word, jadi total ada 6 halaman.

2. Buatlah satu contoh aplikasi RL dalam bidang apapun dan ceritakan bagaimana aplikasi itu di terapkan (tanpa codingan).

3. Buat tabel perbandingan dari metode Dynamic Programming, Monte Carlo, SARSA, Q-Learning, dan Deep Q Learning.

4. Buat dan cari salah satu contoh code dari aplikasi algortima Dynamic Programming, Monte Carlo, SARSA, Q-Learning, dan Deep Q Learning, pilih salah satu yang paling mudah, dan jelaskan cara kerja programnya.

Resume Materi RL

1.1 Pertemuan 1

a) Definisi dan sejarah RL

Reinforcement learning adalah bagian dari artificial intelligence yang melatih algoritma dengan sistem trial and error. RL berinteraksi dengan lingkungannya dan mengamati konsekuensi atas tindakannya sebagai tanggapan atas penghargaan maupun hukuman yang diterima. Informasi yang dihasilkan dari setiap interaksi dengan lingkungan, digunakan RL untuk memperbarui pengetahuannya

Karakteristik dari RL

• Tidak ada "kebenaran" pada proses pembelajaran (training) yang ada hanya hadiah (atau hukuman)

• Data yang diperoleh (State dan Reward) tergantung dari Action yang diberikan oleh Agent.

• Variabel waktu dan urut-urutan state yang dipilih oleh Agent melalui action memiliki pengaruh yang penting.

b) Algoritma RL

• Algoritme RL menghendaki pertukaran state-action-reward (st , at , rt) antara agent dan environment. Proses ini dinamakan sequential decision-making process, dan (st , at , rt) dinamakan experience.

• RL menghitung jumlah dari reward yang diterima oleh agent. Tujuan (objective) dari agent adalah memaksimalkan total reward.

• RL belajar (learn) dari interaksi agent dengan environment menggunakan proses trial & error dan reward yang diperoleh agent, untuk memperkuat (reinforce) aksi positif. A

c) Elements &Evironment

- Agent, yang dimaksud dengan agent adalah perangkat atau software (perangkat lunak) yang dapat belajar dari environment.
- Environment, sedangkan yang dimaksud dengan environment adalah segala sesuatu yang ada di luar agent yang dimana sebagai tempat agent untuk melakukan exploration dan exploitation.

d) Aplikkasi RL

- Atari Games
- OpenAI Five : Dota

1.2 Pertemuan 2

a) Markov decision process

Markov Decision Process merupakan sebuah tuple (S, A, P, R, γ).

Dimana:
•S merupakan state •A merupakan action
•P merupakan state transition probability function (transition probability) •R merupakan reward function
•γ merupakan discount factor (γ ϵ [0, 1])

Markov Decision Processes mendeskripsikan secara formal lingkungan untuk RL
• Secara spesifik biasanya dibuat saat environment fully observable
• Hampir semua RL problems dapat diformalisasi menggunakan MDP
• Optimal Control
• Partially Observable problems
• Bandits problem

b) Dynamic Programming

Value adalah harapan dari return (Expected return)

Value dari policy adalah harapan dari return (Expected return) ketika mengikuti suatu polic

Pemgunaan diskon γ

• Secara matematik memberikan konstanta untuk diskon itu mudah

• Menghindari return tak terhingga dalam proses Markov yang berbentuk siklus

• Masa depan yang tidak pasti yang mungkin tidak diwakili model

• Dalam hal finansial, reward langsung lebih menarik disbanding reward nanti

• Perilaku binatang / manusia biasanya lebih memilih reward langsung

• Tetap memungkinkan untuk menggunakan Markov reward yang tidak didiskon (γ = 1) (jika semua alur tidak berulang)

c) Kesimpulan

• Evaluasi Policy itu adalah MDP, $\pi \longrightarrow V\pi$

• Value Iteration itu adalah MDP $\longrightarrow Vopt, \pi opt$

1.3 Pertemuan 3

a) Monte carlo prediction

- MC didefinisikan untuk jenis episodic environment.
- Ide utama dari MC: Value didapatkan dari rata rata returns.
- Dengan semakin banyak returns, nilai rata-ratanya diharapkan konvergen pada expected value.
- Seluruh episode dipertimbangkan dalam MC
- Hanya satu pilihan setiap perpindahan state di MC, sedangkan DP mempertimbangkan semua probabilitas transisi pada setiap perpindahan state
- Estimasi-estimasi untuk semua state adalah independent di MC, tidak bootstrap
- Waktu yang dibutuhkan untuk mengestimasi suatu state tidak bergantung pada jumlah total

  state

b) Monte carlo estimation & control

First–visit MC mengestimasi value dari pasangan state-action s, a dengan merata-rata returns, saat pertama kali menemui (visit) state s dan mengambil action a dalam suatu episode. Every–visit MC mengestimasi value dari pasangan state-action s, a dengan merata-rata returns, setiap menemui (visit) state s dan mengambil action a dalam suatu episode. Implikasinya, tidak semua state-action s, a akan ditemui (visited). Apakah masih bisa mendekati expected return, $q\pi\ s, a$? Solusi: maintaining exploratio

c) On/off-policy monte carlo

On–policy MC adalah algoritme Monte Carlo yang melakukan evaluasi atau improvisasi dari policy yang digunakan untuk membuat keputusan-keputusan. • Pembahasan MC sebelumnya adalah on-policy MC. • Lalu disini akan dibahas on-policy MC dengan policy dengan nama $\varepsilon - greedy$. • Dalam $\varepsilon - greedy$ policy, hampir semua action yang dipilih, mempunyai action value estimasi yang maksimal, tetapi dengan probabilitas ε dalam memilih action, tidak dengan random/acak.

d) Pemrograman monte carlo

1. Monte Carlo bekerja dengan pengalaman dari sample, dan menggunakannya untuk belajar secara langsung tanpa model.
2. Pada first–visit MC, value function dihitung saat pertama kali mengunjungi state s pertama dalam setiap episode.
3. Pada every–visit MC, value function dihitung setiap mengunjungi state s dalam setiap episode.
4. Pada on-policy MC, agent berkomitmen untuk selalu eksplorasi dan mencoba mencari policy terbaik.
5. Pada off-policy MC, dilakukan evaluasi atau improvisasi dari policy yang berbeda dalam meng-generate data (behavior policy), sedangkan target policy-nya sama

1.4 Pertemuan 4

a) Recap dan intro TDL

TDL = Mengestimasi reward pada setiap Langkah (step)

❑ Agent tidak memiliki pengetahuan tentang environment yang sedang di eksplorasi

❑ Agent belajar dari environment melalui metode trial-and-error

❑ TD Learning merupakan kombinasi antara Monte Carlo dan Dynamic Programming (DP)

❑ Monte Carlo tidak membutuhkan pemodelan terhadap environment

❑ Dynamic Programming (DP)

❑ Model Free Learnin

b) TDL

TD menggunakan metode few steps, hal ini lah yang membedakan dengan metode Monte Carlo yang harus menyesaikan 1 episode.

TD menggunakan policy evaluation method (tanpa kontrol), yang digunakan untuk memprediksi nilai fixed policy.

c) TD control

- Terinspirasi dari policy iteration
- Mengganti value function ($V\pi$) dengan action-value function
- On Policy
- Fakus pada state-action (S, A)

➢ Idea dasar dari TDL adalah kombinasi DP dan MC
➢ TDL diaplikasin pada model free
➢ TDL melakukan update value state per step
➢ Bersifat bootstapping
➢ Menggunakan metode trial and error dalam explorasi lingkungan
➢ Cepat dalam mencapai konvergensi
➢ TD control menggunakan SARSA dan Q-Learning 8.
➢ SARSA bekerja berdasarkan perubahan State-Action (S,A)
➢ SARSA adalah algoritma on policy
➢ Pengambilan keputusan dari state-action SARSA berdasarkan epsilon-greedy policy.

d) Aplikasi TD dan SARSA

- Hands On TD(0)
- Hands On SARSA

1.5 Pertemuan 5

a) Introduction to robotics

Robot merupalan mesin yang beroperasikan secara otomatis yang menggantikan usaha manusia, meskipun mungkin tidak menyerupai manusia dalam penampilan atau melakukan fungsi dengan cara yang mirip manusia.

b) Static Goals Naving using RL
   - Q learning

Pada dasarnya dalam algoritma Q-Learning, agent akan memilih action berdasarkan Q-Table. Agent akan memilih action yang mempunyai Q-Value paling besar berdasarkan state saat ini. Eksplorasi adalah suatu metode pemilihan action dimana agent akan melakukan pemilihan action secara random dengan tujuan ia bisa mengetahui informasi tentang environment secara mendalam. Sedangkan Eksploitasi adalah sebuah metode pemilihan action dengan memilih action yang mempunyai return (dalam hal ini Q-Value) paling besar.

   - Epsilon-greedy exploration

epsilon-greedy exploration digunakan untuk menyeimbangkan antara ekplorasi dan eksploitasi menggunakan nilai epsilon (ε). ε biasa disebut dengan rate eksplorasi.

   - Update Q-Table

Setelah eksekusi action dan mendapatkan reward, saatnya kita mengupdate Q-Table dengan mengganti Q-Value yg lama dengan Q-Value yang baru, untuk mengupdate Q-Table dan menghitung Q-Value yg baru

c) Dynamic goals using RL
   - Deep Q Network

Deep Q Network adalah sebuah NN yang menerima states yg diberikan oleh environment sebagai input, lalu DQN akan menghasilkan output estimasi Q Values pada setiap actions yang dapat diambil pada state tersebut. Tujuan dari NN ini adalah untuk menghasilkan aproksimasi Q Function yg optimal.

   - Experience Replay and Replay Memory

Dalam proses training DQN kita akan menggunakan sebuah teknik yg dinamakan dengan experience replay. Dengan experience replay, kita menyimpan experience dari agent untuk setiap time step ke dalam sebuah wadah yang bernama replay memory.

1.6 Pertemuan 6

a) Q Learnin

- Q-Learning – Off Policy TD Control

Q-Learning merupakan pengembangan RL yang menggunakan Q-values (disebut juga action-values) untuk meningkatkan kemampuan agent belajar agent berulang-ulang. Konsep dasar Q-Learning: • Terinspirasi dari value iteration

• Sample an action

• Observe the reward and the next state

• Take the action with the highest Q (Max Q)

Action dari setiap step dapat dihitung untuk menemukan action terbaik (best action). Untuk keperluan ini digunakan Q-Table.

- Algoritma Q-Learning secara sederhana:

1. Tentukan current state = initial state.

2. Dari current state, cari dengan nilai Q terbesar.

3. Tentukan current state = next state.

4. Ulang Langkah (2) dan (3) hingga current state = goal state

b) Deep Q learning

Deep Q Network adalah sebuah NN yang menerima states yg diberikan oleh environment sebagai input, lalu DQN akan menghasilkan output estimasi Q Values pada setiap actions yang dapat diambil pada state tersebut. Tujuan dari NN ini adalah untuk menghasilkan aproksimasi Q Function yg optimal.

- Deep Q Network (cont)

Goal dari NN ini adalah untuk minimize loss, lalu setelah menghitung loss bobot pada network akan diupdate menggunakan stochastic gradient descent dan backpropagation seperti neural network pada umumnya

- Experience Replay and Replay Memory

Secara teori seluruh experience agent pada setiap time step akan disimpan pada replay memory. Sebenarnya dalam praktiknya, kita akan mendefinisikan besaran experience yang dapat ditampung oleh replay memory sebesar N, dan kita hanya akan menyimpan N terakhir experience dari agent

Contoh Aplikasi RL

Google search autosuggest

- Cara kerja Google search autosuggest

Search Engine atau mesin pencari memiliki 3 fungsi utama, yaitu crawling (untuk menemukan konten), indexing (untuk menyimpan dan melacak konten), dan retrieval (untuk mengambil konten relevan ketika seseorang bertanya pada search engine).

Crawling

Crawling adalah tahap pertama dari cara kerja search engine; ketika mesin pencari mengakuisisi data dari sebuah website. Tahap ini meliputi scanning dan mengumpulkan detail dari setiap halaman website, seperti : judul, gambar, kata kunci, internal link, dan sebagainya. Setiap crawler (bot atau "spider") mengumpulkan data yang berbeda. Crawler ini juga akan mengunjungi ulang halaman pada website untuk melihat perubahan yang dilakukan pada website. Sebagian halaman dapat ditandai dengan "noindex", yaitu seperti memberi sinyal pada bot agar tidak mengindekskan halaman tersebut

Indexing

Indexing adalah tahap dimana data yang sudah di-crawl kemudian diproses dan diletakkan di database. Bayangkan jika Anda mempunyai banyak buku, dan Anda mencatat semua data dari buku yang Anda punya; dari banyaknya halaman, pengarang, genre, tahun terbit, dan lainnya. Crawling adalah proses ketika Anda membaca seluruh data tersebut, sedangkan Indexing adalah ketika Anda membuat catatan tentang seluruh data buku tersebut.

Retrieval & Ranking

Retrieval adalah proses ketika search engine seperti Google memproses permintaan dari pertanyaan yang biasa Anda ketik di Google search, dan memberikan halaman dari website yang paling relevan bagi permintaan Anda. Setiap search engine atau mesin pencari memiliki algoritma atau caranya tersendiri dalam memilih halaman mana yang paling relevan dengan permintaan Anda. Itulah mengapa Google, Bing, Yahoo, dan search engine lainnya memberikan hasil yang berbeda untuk setiap permintaan. Setiap perusahaan search engine merahasiakan algoritma dan cara mereka memberikan ranking/hasil pada halaman website. Semakin relevan hasil yang diberikan, maka semakin baik pula algoritma yang dimiliki

Tabel perbandingan dari metode Dynamic Programming, Monte Carlo, SARSA, Q-Learning, dan Deep Q Learning.

Perbandingan Metode				
Dynamic Programming	**Monte Carlo**	**SARSA**	**Q-Learning**	**Deep Q Learning**
Pemrograman dinamis membutuhkan pengetahuan yang lengkap tentang lingkungan atau semua transisi yang mungkin	MC merupakan penalirikan kesimpul terminal. Dari Satu fakta penting tentang atau perkiraan untuk setiap keadaan adalah independen, yang berarti perkiraan untuk satu keadaan tidak didasarkan pada perkiraan keadaan lain	SARSA cenderung menghindari jalur optimal berbahaya dan hanya perlahan-lahan belajar menggunakannya ketika parameter eksplorasi berkurang	Q-Learning memiliki varians per sampel yang sangat tinggi, dan mungkin menderita masalah konvergen sebagai hasilnya. Ini muncul sebagai masalah ketika melatih jaringan saraf melalui Q-learning	*Deep Q-learning* berusaha mengestimasi *Q-value* dari *action* yang diambil untuk setiap *state* yang ada

Contoh code dari aplikasi

Perhitungan fibonaci

```
def fibonacci(n):
    if n <= 2:
        hasil = 1
    else:
        hasil = fibonacci(n - 1) + fibonacci(n - 2)

    return hasil
```

Algoritma fibonacci sederhana seperti di atas dapat dikatakan sebagai algoritma D&C, karena kita membagikan perhitungan fibonacci ke dua fungsi fibonacci, sampai didapatkan nilai hasil terkecilnya, algoritma ini sering di gunakan dalam pembuatan aplikasi.

Sehingga Pendekatan DP menghindari kalkulasi fungsi yang berulang kali seperti ini dengan melakukan *memoization*, yaitu menyimpan hasil kalkulasi fungsi tersebut dan menggunakan nilai yang disimpan ketika perhitungan yang sama dibutuhkan kembali. Dengan menyimpan hasil kalkulasi seperti ini, tentunya jumlah total langkah perhitungan yang harus dilakukan menjadi berkurang.