Pertemuan 1

Bagian Introduction to Reinforcement Learning

• Sejarah Reinforcement Learning (RL)

1. Cabang Machine Learning

Unsupervised Learning, Reinforcement Learning, dan Supervised Learning

1. Stanford Marshmallow Test

Pada tahun 1972 sebagai pemula RL. Pada percobaan tersebut terdapat Agent, Reward, dan Action.

1. Karakteristik Reward dan Action

• Reward: Sistem Reward dalam sebuah Lingkungan mungkin tidak langsung diberikan. (delayed reward). Reward bisa dilihat sebagai umpan balik terhadap Agent, karena beberapa reward mengalami penundaan maka umpan balik seringkali menjadi tertuda. Ada yang namanya “Delayed Gratification” dan seringkali memberikan total reward yang lebih besar. • Action: Pada contoh di atas Semua Tindakan yang mungkin telah diketahui.

1. RL secara formal dan adanya Trial & Erorr system 1- system 3

• 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.

• Agent berinteraksi terhadap environment dengan melakukan sebuah aksi/tindakan dari satu kondisi ke kondisi yang lain. • Agent akan menerima reward berdasarkan aksinya. • Berdasarkan reward tersebut, agent akan memahami apakah aksinya baik atau buruk. • Jika aksinya baik, agent akan menerima reward positif, sehingga agent tersebut akan cenderung lebih memilih melakukan aksi yang serupa dengan aksi tersebut (exploitation), atau melakukan aksi lain yang dapat menghasilkan reward positif.

• Elements & Environment dari RL

Di luar dari Agent dan Environment yang merupakan sebagai element utama, ada 4 sub elemen sebagai penyusun utama sistem Reinforcement Learning: Elemen utama: 1. Agent 2. Environment Sub Elemen utama: 1. Policy 2. Reward signal 3. Value function 4. Model environment (optional)

❑ 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.

• Aplikasi dari RL

Reinforcement Learning for AI of Game • Examples of Building AI for Snake Game using Reinforcement Learning Mendifinisikan state yang mungkin terjadi (11 states) Mendefinisikan action yang ada ( 3 actions : Straight, Turn right, Turn left)

Pertemuan 2

Markov Decision Process (MDP) dan Dynamic Programming

1. Markov Decision Proses

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

Aplikasi RL dalam Dunia Nyata

• Robotik: Memutuskan bergerak kemana. • Alokasi Sumber Daya: Memutuskan

apa yang diproduksi • Pertanian : Menentukan apa yang

ditanam, tapi tidak tahu cuaca dan hasil tanamContoh: Permainan Dadu

• Untuk setiap ronde: 𝑟 = 1, 2,… • Bisa memilih untuk main atau keluar • Jika keluar, mendapatkan 10 koin dan permainan berakhir • Jika main, mendapatkan 4 koin dan dadu dilempar • Jika dadu keluar angka 1 atau 2, permainan berakhir • Selain itu, lanjutkan ronde berikut

1. MDP dalam rumus RL secara formal

MDP untuk Permainan Dadu

• State (s) node • Action (a) • State : Action q(s, a) node • Probability P(s, a, s’) • Reward r(s, a, s’)

• States: himpunan dari states • 𝑆𝑠𝑡𝑎𝑟𝑡 ∈ 𝑆𝑡𝑎𝑡𝑒 : state awal • Actions(s): actions yang tersedia dari state s • 𝑃 𝑠, 𝑎, 𝑠 ′ : probabilitas ke s’ jika mengambil action a dari state s • Reward (s, a, s’) : idem dengan diatas • isEnd(s) : apakah state terakhir • 0 ≤ 𝛾 ≤ 1 faktor diskon

1. Menentukan Value sebuah Policy.

Contoh Transportasi

• Sebuah kota yang mempunyai nomor blok 1 sampai n • Berjalan dari s ke s+1 membutuhkan waktu 1 menit • Naik tram ajaib dari s ke 2s membutuhkan waktu 2 menit • Bagaimana berpindah dari 1 ke n dalam waktu paling sedikit? • Tram punya kemungkinan gagal 0.5 • Jika gagal, waktu habis 2 menit tapi tidak pindah blok

Solusi

• Policy adalah fungsi 𝜋 yang memetakan untuk setiap state 𝑠 ∈ 𝑆𝑡𝑎𝑡𝑒𝑠 ke action 𝑎 ∈ 𝐴𝑐𝑡𝑖𝑜𝑛(𝑠) • Contoh Robot pencari harta karun:

Evaluasi

• Mengikuti policy akan bisa menghasilkan jalur yang acak (mengapa?) • Return (utility) dari sebuah policy adalah Jumlah dari reward selama mengikuti jalur (nilai yang acak) • Contoh policy = “main” pada permainan dadu:

Pertemuan 3

Monte Carlo Prediction

• Definisi Monte Carlo Prediction

Konsep Monte Carlo (MC)

• MC tidak mengambil pengetahuan lengkap dari environment. • MC belajar dari experience, episode per episode, baik itu experience aktual, maupun simulasi. • MC belajar dari episode-episode secara utuh dan independent, tidak bootstrapping. • MC didefinisikan untuk jenis episodic environment. • Ide utama dari MC: Value didapatkan dari ratarata 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

• Mengetahui Monte Carlo Estimation & Control

MC Estimation

• Karena model tidak tersedia, perlu untuk mengestimasi action juga, selain hanya mengestimasi state-nya. • Tidak seperti DP, pada Model Free Algorithm, MC, state saja tidak cukup untuk menentukan policy. • Pada MC, action diperlukan dalam menentukan policy. Policy Evaluation Problem. • Policy evaluation problem mengestimasi 𝑞𝜋 𝑠, , yaitu expected return ketika mulai dari state 𝑠, melakukan action 𝑎, dan mengikuti policy 𝜋. • 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, 𝑞𝜋 𝑠, ? • Solusi: maintaining exploration

Maintaining Exploration dilakukan dengan: • Episode-episode memulai dengan sebuah pasangan state-action 𝑠, . Mekanisme pertama dari exploring starts. • Setiap pasangan memiliki probabilitas yang tidak sama dengan nol untuk dipilih sebagai permulaan episode. Mekanisme kedua dari exploring starts. • Semua pasangan state-action 𝑠, 𝑎 memiliki probabilitas tidak sama dengan nol untuk memilih semua action pada setiap state.

• Mengetahui On-Policy dan Off-Policy Monte Carlo

On-Policy MC: Konsep

• 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 𝜀 − 𝑔𝑟𝑒𝑒𝑑𝑦. • Dalam 𝜀 − 𝑔𝑟𝑒𝑒𝑑𝑦 policy, hampir semua action yang dipilih, mempunyai action value estimasi yang maksimal, tetapi dengan probabilitas 𝜀 dalam memilih action, tidak dengan random/acak.

Pertemuan 4

Temporal-Difference Learning

• Introduction TDL

Temporal Difference Learning Kasus 1: Pagi hari teman-teman mahasiswa ingin pergi kuliah. Pergi dengan mobil pribadi yang digunakan menuju kampus. Kemudian sesampai di jalan umum terjadi macet yang sangat parah, kemudian dengan menggunakan ponsel, anda melakukan update berkala mengenai informasi macet tersebut, anda dapat memperoleh informasi setiap 10 menit sekali terhadap bagaimana keadaan di lapangan saat itu. Sehingga anda mendapat informasi lebih banyak dan detail ketika sesampai dikampus tentang macet tersebut dan mempermudah anda untuk mencari alternatif jalan lainnya. Kasus 2: Pagi hari pak Budi ingin pergi ke kantornya di Jakarta Pusat, namun karena telat sedikit maka beliau terjebak macet. Pak budi tidak memperoleh informasi tentang kemacetan tersebut, dan beliau harus menunggu sesampai dikantor baru mendapatkan informasi kemacetan tersebut secara detail.

Dynamic Programming & Monte Carlo Method Dynamic Programming ❖Update per step, menggunakan bootstrapping ❖Membutuhkan model environment ❖Computation cost Monte Carlo Method ❖Update per episode ❖Model-free (tidak membutuhkan model environment) ❖Sulit untuk di aplikasikan pada continuing task

• TD (0)

Input: Policy 𝜋 yang akan di evalusi Algoritma parameter: step size 𝛼 ∈ 0,1 Inisialisasi 𝑉 𝑠 , untuk semua 𝑠 ∈ 𝒮 +, arbitrarily except that 𝑉 𝑡𝑒𝑟𝑚𝑖𝑛𝑎𝑙 = 0 Loop semua episode: Inisialisasi 𝑆 Loop setiap step episode: 𝐴 ← action diberikan oleh 𝜋 untuk 𝑆 Ambil action 𝐴, observe 𝑅, 𝑆 ′ 𝑉 𝑆 ← 𝑉 𝑆 + 𝛼 𝑅 + 𝛾𝑉 𝑆 ′ − 𝑉(𝑆) 𝑆 ← 𝑆 ′ hingga 𝑆 is terminal

Contoh kasus : Driving Home

Misalnya anda bekerja pada sebuah kantor di Jakarta, dan anda pergi bolak balik setiap hari dari rumah ke kantor dan sebaliknya. • Setiap hari bolak balik rumah kantor • Goal memprediksi berapa lama waktu pada waktu tertentu (misalnya saat pulang dari kantor ke rumah). • Pada saat mulai start dari kantor ada beberapa yang di catat, yaitu waktu, hari, dan lain-lain. Pertimbangkan policy evaluation atau prediction task

• TD Control - SARSA

SARSA (State-Action-Reward-State-Action)

• Terinspirasi dari policy iteration • Mengganti value function (𝑉𝜋) dengan action-value function • On Policy • Fakus pada state-action (S, A)

Algoritma Sarsa (on-policy TD Control) Algoritma parameter: step size 𝛼 ∈ 0,1 , small 𝜀 > 0 Inisialisasi 𝑄 𝑠, 𝑎 , for all 𝑠 ∈ 𝒮 + , 𝑎 ∈ 𝒜(𝑠), arbitrarily except that 𝑄 𝑡𝑒𝑟𝑚𝑖𝑛𝑎𝑙 − 𝑠𝑡𝑎𝑡𝑒,· = 0 Loop untuk setiap episode: Inisialisasi 𝑆 Pilih 𝐴 from 𝑆 menggunakan policy derived from 𝑄 𝑒. 𝑔. , 𝜀 − 𝑔𝑟𝑒𝑒𝑑𝑦 Loop untuk setiap step episode: Ambil action 𝐴, observe 𝑅, 𝑆 ′ Pilih 𝐴 ′ dari 𝑆 ′menggunakan policy derived dari 𝑄 𝑒. 𝑔. , 𝜀 − 𝑔𝑟𝑒𝑒𝑑𝑦 𝑄 𝑆, 𝐴 ← 𝑄 𝑆, 𝐴 + 𝛼 𝑅 + 𝛾𝑄 𝑆 ′ , 𝐴 ′ − 𝑄 𝑆, 𝐴 𝑆 ← 𝑆 ′ ; 𝐴 ← 𝐴 ′ hingga 𝑆 is terminal

Pertemuan 5

Q Learning dan Deep Q Learning

1. Definisi Q Learning,

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 (off-policy TD control) for estimating 𝝅 ≈ 𝝅∗ Parameter algoritma: step size 𝛼 ∈ 0,1 , small 𝜀 > 0 Inisialisasi 𝑄 𝑠, 𝑎 , untuk semua 𝑠 = 𝒮 + , 𝑎 ∈ 𝒜(𝑠), arbitrary except that 𝑄 𝑡𝑒𝑟𝑚𝑖𝑛𝑎𝑙,· = 0 Loop untuk setiap episode: Inisialisasi 𝑆 Loop untuk setiap step dari episode: Pilih 𝐴 dari 𝑆 menggunakan policy derived from 𝑄(𝑒. 𝑔. , 𝜀 − 𝑔𝑟𝑒𝑒𝑑𝑦) Ambil action 𝐴, observe 𝑅, 𝑆 ′ 𝑄 𝑆, 𝐴 ← 𝑄 𝑆, 𝐴 + 𝛼 𝑅 + 𝛾 max 𝑎 𝑄 𝑆 ′ , 𝑎 − 𝑄 𝑆, 𝐴 𝑆 ← 𝑆 ′ hingga 𝑆 is terminal

1. Memahami bagaimana Q Learning bekerja

Objektif dari Q Learning adalah mencari policy yang optimal berdasarkan in the sense that the expected value of the total reward over all successive steps is the maximum achievable. Atau dengan kata lain, tujuan dari QLearning adalah mencari policy yang optimal dan mencari optimal QValues untuk setiap pasangan state-action.

Init reward and Q Table Choose Action Perform Action and get reward Update Q Table

1. Mengetahui Deep Q Learning

Pada Deep Q Learning kita akan menggantikan Q-Table menggunakan sebuah Neural Network yang biasa disebut dengan Deep Q Network atau DQN

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.

1. Memahami bagaimana Deep Q Learning Bekerja

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.

Init replay memory capacity Init NN Choose action for this state (exploration or exploitation Execute action and get reward and next state Store experience to replay memory Get sample random batch experience from replay memory pass the sample random batch to NN Calculate Loss Update the weight

Pertemuan 6

Robotics

• Dasar dari Robotika

➢ Robotics adalah suatu disiplin ilmu yang mempelajari tentang konsep suatu robot

➢ 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.

• Mengetahui Kegunaan Konsep Machine Learning untuk Robotika

Reinforcement Learning for Robotics

RL for Robotics Kinematics System Manipulator Robot Bipedal Robot

• Mengimplementasikan Reinforcement Learning pada Robotika

1. Definisi Reward dan Q Table (cont)
2. Choose Action

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. Tapi…. Hal tersebut akan terdengar aneh untuk Langkah pertama, bukan? Karena pada Langkah pertama semua Q-Value yg ada pada Q-Table bernilai 0, jadi gimana cara agent memilih action? Nah, untuk jawab pertanyaan ini kita akan coba mengingat Kembali tentang Eksplorasi dan Eksploitasi.

1. Choose Action : Eksplorasi vs Eksploitasi

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. Lalu, pasti teman teman semua berfikir kalau pemilihan action dengan metode eksploitasi adalah yg terbaik.

1. Perform Action and Get Reward
2. Update Q-Table
3. Deep Q Learning

Pada Deep Q Learning kita akan menggantikan Q-Table menggunakan sebuah Neural Network yang biasa disebut dengan Deep Q Network atau DQN.

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.

1. 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. Kita dapat merepresentasikan experience dari agent untuk setiap t sebagai et . Pada time t, agent experience et didefinisikan sebagai tuple berikut : Tuple ini menyimpang state (st), action (at) yang diambil, reward (rt+1) yang didapatkan oleh agent dan state selanjutnya (st+1).

1. How We Calculate Loss on DQN? : Calculate the Max Term