5.5.3 Deep Q Learning

Tags

Deep Q Learning (DQN)

Q-learning adalah algoritme sederhana namun cukup kuat untuk membuat lembar contekan untuk agen kami. Ini membantu agen mengetahui dengan tepat tindakan mana yang harus dilakukan.

Tapi bayangkan lingkungan dengan 10.000 negara bagian dan 1.000 tindakan per negara bagian. Ini akan membuat tabel 10 juta sel. Segalanya akan cepat lepas kendali!

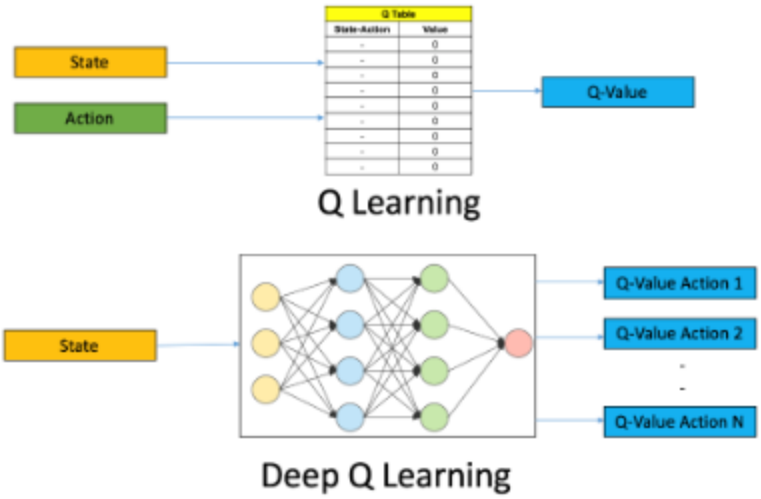
Cukup jelas bahwa kami tidak dapat menyimpulkan nilai Q dari status baru dari status yang sudah dieksplorasi. Ini menyajikan dua masalah:

- Pertama, jumlah memori yang diperlukan untuk menyimpan dan memperbarui tabel itu akan meningkat seiring dengan meningkatnya jumlah status
- Kedua, jumlah waktu yang diperlukan untuk menjelajahi setiap negara bagian untuk membuat tabel-Q yang diperlukan tidak realistis

Inilah pemikiran – bagaimana jika kita memperkirakan nilai-Q ini dengan model pembelajaran mesin seperti jaringan saraf? Nah, inilah ide di balik algoritma DeepMind yang membuatnya diakuisisi oleh Google seharga 500 juta dolar!

Deep Q-Network

Dalam pembelajaran Q yang mendalam, kami menggunakan jaringan saraf untuk memperkirakan fungsi nilai-Q. Status diberikan sebagai input dan nilai Q dari semua tindakan yang mungkin dihasilkan sebagai output. Perbandingan antara Q-learning & deep Q-learning diilustrasikan dengan indah di bawah ini:



Jadi, apa saja langkah-langkah yang terlibat dalam pembelajaran penguatan menggunakan jaringan Q-learning mendalam (DQNs)?

1. Semua pengalaman masa lalu disimpan oleh pengguna dalam memori
2. Tindakan selanjutnya ditentukan oleh output maksimum dari jaringan-Q
3. Fungsi kerugian di sini adalah kesalahan kuadrat rata-rata dari nilai-Q yang diprediksi dan nilai-Q target –Q*. Ini pada dasarnya adalah masalah regresi. Namun, kita tidak mengetahui target atau nilai aktual di sini karena kita sedang berhadapan dengan masalah pembelajaran penguatan. Kembali ke persamaan pembaruan nilai-Q yang diturunkan dari persamaan Bellman. kita punya:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big]$$

Bagian berwarna hijau mewakili target. Kita dapat berargumen bahwa ia memprediksi nilainya sendiri, tetapi karena R adalah hadiah sejati yang tidak bias, jaringan akan memperbarui gradiennya menggunakan propagasi balik untuk akhirnya bertemu.

What is Experience Replay?

Experience Replay adalah tindakan menyimpan dan memutar ulang status game (status, aksi, hadiah, status berikutnya) yang dapat dipelajari oleh algoritma RL. Experience Replay dapat digunakan dalam algoritma Off-Policy untuk belajar secara offline. Metode di luar kebijakan dapat memperbarui parameter algoritme menggunakan informasi yang disimpan dan disimpan dari tindakan yang diambil sebelumnya. Deep Q-

Learning menggunakan Experience Replay untuk belajar dalam batch kecil untuk menghindari penyimpangan distribusi set data dari berbagai status, tindakan, penghargaan, dan status berikutnya yang akan dilihat oleh jaringan saraf.

Misalkan kita mencoba untuk mendekati sebuah kompleks, fungsi nonlinier, Q(s, a), dengan Neural Network. Untuk melakukan ini, kita harus menghitung target menggunakan persamaan Bellman dan kemudian mempertimbangkan bahwa kita memiliki masalah pembelajaran yang diawasi. Namun, salah satu persyaratan mendasar untuk optimasi SGD adalah bahwa data pelatihan independen dan terdistribusi secara identik dan ketika Agen berinteraksi dengan Lingkungan, urutan tupel pengalaman dapat sangat berkorelasi. Algoritme pembelajaran Q naif yang belajar dari masing-masing tupel pengalaman ini secara berurutan berisiko terpengaruh oleh efek korelasi ini.

Kami dapat mencegah nilai tindakan berosilasi atau menyimpang secara dahsyat menggunakan buffer besar pengalaman kami dan sampel data pelatihan darinya, alih-alih menggunakan pengalaman terbaru kami. Teknik ini disebut replay buffer atau experience buffer. Buffer replay berisi kumpulan tupel pengalaman (S, A, R, S′). Tupel secara bertahap ditambahkan ke buffer saat kita berinteraksi dengan Lingkungan. Implementasi paling sederhana adalah buffer dengan ukuran tetap, dengan data baru ditambahkan ke akhir buffer sehingga mendorong pengalaman tertua keluar darinya.

Tindakan mengambil sampel sejumlah kecil tupel dari buffer replay untuk dipelajari dikenal sebagai Experience replay. Selain mematahkan korelasi yang berbahaya, replay pengalaman memungkinkan kita untuk belajar lebih banyak dari tupel individu beberapa kali, mengingat kejadian langka, dan secara umum memanfaatkan pengalaman kita dengan lebih baik.

Q-Learning using Experience Replay

This is the pseudo code for Experience replay

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
For episode = 1, M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1$,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $\left(\phi_t, a_t, r_t, \phi_{t+1}\right)$ in D
    Sample random minibatch of transitions $\left(\phi_j, a_j, r_j, \phi_{j+1}\right)$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^{-}\right) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

Mari kita pahami apa yang terjadi dalam kode ini

1. Inisialisasi memori replay dan dua pendekatan Q identik (DNN). adalah perkiraan target kami.
2. Mainkan m episode (permainan lengkap)
3. Mulai episode dari _1 (piksel di layar awal).
4. Praproses status (termasuk 4 frame terakhir, konversi RGB ke skala abu-abu, downsampling, cropping)
5. Untuk setiap langkah waktu selama episode

- Dengan probabilitas kecil, pilih tindakan acak (eksplorasi), jika tidak, pilih tindakan terbaik yang diketahui saat ini (eksploitasi).
- Jalankan tindakan yang dipilih dan simpan transisi yang diamati (diproses) dalam memori replay.
- Cicipi minibatch acak transisi dari memori replay dan lakukan langkah gradien yang layak pada Q (bukan pada Q)

- Sekali setiap beberapa langkah atur fungsi target, Q, menjadi sama dengan Q
- Pembelajaran online yang tertunda seperti itu membantu dalam praktik: ”Modifikasi ini membuat algoritma lebih stabil dibandingkan dengan Q-learning online standar, di mana pembaruan yang meningkatkan (s_t,a_t) sering juga meningkatkan ($s\_(t$+1),a) untuk semua dan karenanya juga meningkatkan target y_j, mungkin mengarah untuk osilasi atau divergensi kebijakan”

Double Deep Q Network (Double DQN)

Implementasi Double Q-Learning dengan Deep Neural Network disebut dengan Double Deep Q Network (Double DQN).

Double DQN diusulkan oleh H. van Hasselt, 2016. Terinspirasi oleh Double Q-Learning, Double DQN menggunakan dua Deep Neural Networks yang berbeda, Deep Q Network (DQN) dan Target Network.

Perhatikan bahwa tidak ada tingkat pembelajaranαsaat memperbarui nilai-Q karena akan digunakan dalam tahap pengoptimalan memperbarui parameter Jaringan Q Dalam.

Double Q Network

estimated/expected Q-value

$$Q_{qnet}(s_t, a_t) \leftarrow R_{t+1} + \gamma\, Q_{tnet}(s_{t+1}, a)$$

$$a = \max_a Q_{qnet}(s_{t+1}, a)$$

$$q_{estimated} = Q_{tnet}(s_{t+1}, a)$$

Deep Q Network —memilih tindakan terbaik dengan nilai Q maksimum dari status berikutnya.

$$a = \max_a Q_{qnet}(s_{t+1}, a)$$

Jaringan Target —menghitung perkiraan nilai Q dengan tindakan yang dipilih di atas.

$$q_{estimated} = Q_{tnet}(s_{t+1}, a)$$

Perbarui nilai Q Jaringan Deep Q berdasarkan perkiraan nilai Q dari Jaringan Target

estimated/expected Q-value

$$Q_{qnet}(s_t, a_t) \leftarrow R_{t+1} + \gamma\, Q_{tnet}(s_{t+1}, a)$$

Perbarui parameter Jaringan Target berdasarkan parameter Jaringan Q Dalam per beberapa iterasi.

$$Q_{tnet}(s, a) = Q_{qnet}(s, a)$$

Perbarui parameter Deep Q Network berdasarkan pengoptimal Adam.

Cross Entropy Method

Bagaimana kita memecahkan masalah optimasi kebijakan yang memaksimumkan total reward yang diberikan beberapa kebijakan parametrized?

Discounted future reward

Untuk memulainya, untuk sebuah episode hadiah total adalah jumlah dari semua hadiah. Jika lingkungan kita stokastik, kita tidak akan pernah bisa yakin apakah kita akan mendapatkan imbalan yang sama di waktu berikutnya, kita melakukan tindakan yang sama. Jadi, semakin banyak kita pergi ke masa depan, semakin banyak total imbalan di masa depan yang mungkin berbeda. Jadi, untuk alasan itu adalah umum untuk menggunakan diskon imbalan masa depan di mana parameter diskon disebut faktor diskon dan berada di antara 0 dan 1.

Strategi yang baik untuk agen adalah selalu memilih tindakan yang memaksimalkan imbalan (diskon) di masa depan. Dengan kata lain, kami ingin memaksimalkan imbalan yang diharapkan per episode.

Parametrized Policy

Kebijakan stokastik didefinisikan sebagai probabilitas bersyarat dari beberapa tindakan yang diberikan suatu keadaan. Keluarga kebijakan yang diindeks oleh vektor parameter disebut kebijakan parametris. Kebijakan ini didefinisikan analog dengan klasifikasi pembelajaran terawasi atau masalah regresi. Dalam kasus kebijakan diskrit kami menampilkan vektor probabilitas dari tindakan yang mungkin dan dalam kasus kebijakan berkelanjutan kami mengeluarkan kovarians rata-rata dan diagonal dari distribusi Gaussian dari mana kami kemudian dapat mengambil sampel tindakan berkelanjutan kami.

Cross entropy method (CEM)

Jadi bagaimana kita memecahkan masalah optimasi kebijakan memaksimalkan total (diskon) reward diberikan beberapa kebijakan parametrized? Pendekatan paling sederhana adalah optimasi bebas derivatif (DFO) yang melihat masalah ini sebagai kotak hitam sehubungan dengan parametertheta. Kami mencoba banyak yang berbeda dan menyimpan hadiah untuk setiap episode. Ide utamanya kemudian adalah bergerak menuju goodtheta.

Salah satu pendekatan DFO tertentu disebut CEM. Di sini, kapan saja, Anda mempertahankan distribusi di atas vektor parameter dan memindahkan distribusi ke parameter dengan imbalan yang lebih tinggi. Ini bekerja dengan sangat baik, bahkan jika itu tidak terlalu efektif ketika merupakan vektor berdimensi tinggi.

Summary

- Dalam pembelajaran Q yang mendalam, kami menggunakan jaringan saraf untuk memperkirakan fungsi nilai-Q. Status diberikan sebagai input dan nilai Q dari semua tindakan yang mungkin dihasilkan sebagai output.
- Double DQN menggunakan dua Deep Neural Network yang berbeda, Deep Q Network (DQN) dan Target Network.
- Tidak ada kecepatan pembelajaranαsaat memperbarui nilai-Q karena akan digunakan dalam tahap pengoptimalan memperbarui parameter Jaringan Q Dalam.
- Di CEM kapan saja, Anda mempertahankan distribusi di atas vektor parameter dan memindahkan distribusi ke parameter dengan imbalan yang lebih tinggi