Sztuczne sieci neuronowe z biblioteką PyTorch

Przemysław Olbratowski

30 stycznia 2021

Spis treści

1 Regresja liniowa przy pomocy równań normalnych (numpy)

1.1 Regresja liniowa

W ogólnym przypadku do punktów pomiarowych $(\vec{x}_i, \vec{y}_i)$ dopasowujemy funkcję

$$\vec{f}(\vec{x}) = \sum_k \vec{f}_k(\vec{x}) p_k$$

Ograniczymy się do jednowymiarowych punktów (x_i, y_i) i dopasowania wielomianu

$$f(x) = \sum_k x^k p_k$$

Robimy to metodą najmniejszych kwadratów minimalizując funkcję straty

$$L(\vec{p}) = \sum_i (f(x_i) - y_i)^2$$

ze względu na współczynniki p_k. Wartości $f(x_i)$ można zapisać jako

$$f(x_i) = \sum_k x_i^k p_k = \sum_k X_{ik} p_k = (\mathbf{X}\vec{p})_i$$

gdzie $\mathbf{X}$ nazywa się design matrix. Funkcję straty można zapisać jako

$$L(\vec{p}) = \sum_i (f(x_i) - y_i)^2 = \sum_i ((\mathbf{X}\vec{p})_i - y_i)^2 = \sum_i ((\mathbf{X}\vec{p} - \vec{y})_i)^2 = (\mathbf{X}\vec{p} - \vec{y})^2$$

1.2 Równania normalne

Aby zminimalizować funkcję straty należy przyrównać jej gradient do zera czyli rozwiązać układ równań

$$\vec{\nabla}(\mathbf{X}\vec{p} - \vec{y})^2 = 0$$

Po wykonaniu różniczkowania układ ten przyjmuje postać

$$\mathbf{X}^T\mathbf{X}\vec{p} = \mathbf{X}^T\vec{y}$$

Są to tak zwane równania normalne. Stanowią one układ równań liniowych, więc łatwo je rozwiązać.

2 Regresja liniowa przy pomocy minimalizacji metodą gradientu (numpy)

2.1 Minimalizacja metodą gradientu

Funkcję straty

$$L(\vec{p}) = (\mathbf{X}\vec{p} - \vec{y})^2$$

można też minimalizować metodą gradientu. Jej gradient wynosi

$$\vec{\nabla} L(\vec{p}) = \vec{\nabla}(\mathbf{X}\vec{p} - \vec{y})^2 = 2\mathbf{X}^T(\mathbf{X}\vec{p} - \vec{y})$$

Kolejne przybliżenie $\vec{p}$ dane jest przez poprzednie wzorem

$$\vec{p}_{k+1} = \vec{p}_k - \alpha\vec{\nabla} L(\vec{p}_k)$$

gdzie α to szybkość uczenia albo learning rate.

3 Zbiór danych do klasyfikacji irysów

4 Regresja logistyczna (numpy)

4.1 Prawdopodobieństwo przynależności do klas

I próbek po J cech układamy w macierz danych X_{ij}. Każda próbka należy do jednej z K klas. Przyjmujemy, że prawdopodobieństwo przynależności próbki i do klasy k jest dane funkcją softmax

$$P_{ik} = \frac{\exp(p_{0k} + X_{i0}p_{1k} + X_{i1}p_{2k} + ...)}{\sum_{k'} \exp(p_{0k'} + X_{i0}p_{1k'} + X_{i1}p_{2k'} + ...)}$$

gdzie p_{jk} to macierz współczynników. Dla poprawy zapisu do macierzy X_{ij} dodajemy po lewej kolumnę jedynek czyniąc z niej tak zwaną design matrix. Prawdopodobieństwo przyjmuje wtedy postać

$$P_{ik} = \frac{\exp(X_{i0}p_{0k} + X_{i1}p_{1k} + X_{i2}p_{2k} + ...)}{\sum_{k'} \exp(X_{i0}p_{0k'} + X_{i1}p_{1k'} + X_{i2}p_{2k'} + ...)}$$

Wprowadzamy macierz aktywacji

$$A_{ik} = X_{i0}p_{0k} + X_{i1}p_{1k} + X_{i2}p_{2k} + ... = \sum_j X_{ij}p_{jk}$$

Prawdopodobieństwo przyjmuje wtedy postać

$$P_{ik} = \frac{\exp A_{ik}}{\sum_{k'} \exp A_{ik'}}$$

Spełniony jest tu warunek normalizacji

$$\sum_k P_{ik} = 1$$

O każdej próbce i wiemy, że należy do klasy k_i. Chcemy tak dobrać współczynniki p_{jk} aby dla każdej próbki i uzyskać jak największe prawdopodobieństwo jej przynależności do klasy k_i. Uczynimy to metodą największej wiarygodności. Prawdopodobieństwo przynależności próbki i do klasy k_i wynosi

$$P_{ik_i} = \frac{\exp A_{ik_i}}{\sum_{k'} \exp A_{ik'}}$$

Prawdopodobieństwo przynależności wszystkich próbek do swoich klas wynosi

$$P = \prod_i P_{ik_i}$$

4.2 Metoda największej wiarygodności

Wielkość P rozumiana jako funkcja współczynników p_{jk} to funkcja wiarygodności. Postępując metodą największej wiarygodności maksymalizujemy P ze względu na p_{jk}. Wygodniej jest minimalizować funkcję straty równą

$$L = -\log P = -\log \prod_i P_{ik_i} = -\sum_i \log P_{ik_i}$$

Minimalizacji tej dokonujemy numerycznie metodą gradientu. Pochodne L po p_{jk} wynoszą

$$\frac{\partial L}{\partial p_{jk}} = -\sum_i \frac{1}{P_{ik_i}} \frac{\partial P_{ik_i}}{\partial p_{jk}}$$

Ze wzoru na pochodną funkcji złożonej

$$\frac{\partial P_{ik_i}}{\partial p_{jk}} = \sum_{k'} \frac{\partial P_{ik_i}}{\partial A_{ik'}} \frac{\partial A_{ik'}}{\partial p_{jk}}$$

Zajmijmy się pierwszym członem iloczynu

$$P_{ik_i} = \frac{\exp A_{ik_i}}{\sum_{k''} \exp A_{ik''}}$$

Ze wzoru na pochodną ilorazu

$$\frac{\partial P_{ik_i}}{\partial A_{ik'}} = \frac{\exp A_{ik_i}\delta_{k_ik'} \sum_{k''} \exp A_{ik''} - \exp A_{ik_i} \sum_{k''} \exp A_{ik''}\delta_{k''k'}}{(\sum_{k''} \exp A_{ik''})^2} =$$

$$= \frac{\exp A_{ik_i}\delta_{k_ik'} \sum_{k''} \exp A_{ik''} - \exp A_{ik_i} \exp A_{ik'}}{(\sum_{k''} \exp A_{ik''})^2} = \frac{\exp A_{ik_i}}{\sum_{k''} \exp A_{ik''}} \left(\delta_{k_ik'} - \frac{\exp A_{ik'}}{\sum_{k''} \exp A_{ik''}}\right) =$$

$$= P_{ik_i}(\delta_{k_ik'} - P_{ik'})$$

Zajmijmy się drugim członem iloczynu

$$A_{ik'} = \sum_{j'} X_{ij'} p_{j'k'}$$

Ze wzoru na pochodną sumy

$$\frac{\partial A_{ik'}}{\partial p_{jk}} = \sum_{j'} X_{ij'}\delta_{j'j}\delta_{k'k} = X_{ij}\delta_{k'k}$$

Ostatecznie pochodna prawdopodobieństwa wynosi

$$\frac{\partial P_{ik_i}}{\partial p_{jk}} = \sum_{k'} P_{ik_i}(\delta_{k_ik'} - P_{ik'})X_{ij}\delta_{k'k} = P_{ik_i}(\delta_{k_ik} - P_{ik})X_{ij}$$

Pochodna straty

$$\frac{\partial L}{\partial p_{jk}} = -\sum_i \frac{1}{P_{ik_i}} P_{ik_i}(\delta_{k_ik} - P_{ik})X_{ij} = -\sum_i (\delta_{k_ik} - P_{ik})X_{ij}$$

9 Regresja logistyczna

9.1 Dokładność obliczeń

W regresji logistycznej prawdopodobieństwo P_{ik} wyraża się przez aktywacje A_{ik} funkcją softmax

$$P_{ik} = \frac{\exp A_{ik}}{\sum_{k'} \exp A_{ik'}}$$

Funkcja ta jest niezmiennicza ze względu na zmiany aktywacji A_{ik} o niezależną od k stałą addytywną A_i

$$\frac{\exp(A_i + A_{ik})}{\sum_{k'} \exp(A_i + A_{ik'})} = \frac{\exp A_{ik}}{\sum_{k'} \exp A_{ik'}} = P_{ik}$$

Jednak obliczone numerycznie wartości P_{ik} zależą od A_i tak bardzo, że dla dużych A_i mogą być całkowicie błędne. Na szczęście strata

$$L = -\sum_i \log P_{ik_i}$$

Zależy tylko od logarytmów P_{ik}, które można obliczyć numerycznie w specjalny sposób eliminujący przedstawione niedokładności. Taki specjalny sposób jest zaimplementowany w funkcjach biblioteki pytorch, które liczą od razu logarytm funkcji softmax. Dlatego należy używać wyłącznie tych funkcji i nigdy nie liczyć straty ręcznie, obliczając najpierw wartość funkcji softmax a potem jej logarytm. Jeżeli potrzebne są prawdopodobieństwa P_{ik}, czyli wartości funkcji softmax, to nie należy ich liczyć ręcznie obliczając eksponensy. Zamiast tego należy korzystać z bibliotecznej wersji funkcji softmax, która zmniejsza przedstawione niedokładności, choć nie usuwa ich zupełnie. Na szczęście w zagadnieniu klasyfikacji nie musimy znać dokładnych wartości P_{ik} lecz jedynie wiedzieć, która z nich jest największa dla ustalonego i. P_{ik} jest oczywiście największe dla tego k, dla którego największe jest A_{ik}. Dlatego klasyfikacji możemy dokonać na podstawie samych aktywacji, w ogóle nie obliczając prawdopodobieństw. Eliminuje to również konieczność obliczania funkcji softmax, czyli przyspiesza obliczenia.

9.2 Minimalizacja

W regresji logistycznej aktywacje A_{ik} są liniowymi kombinacjami parametrów p_{jk}

$$A_{ik} = \sum_j X_{ij} p_{jk}$$

Jeżeli wszystkie parametry zmienimy o niezależną od k stałą addytywną p_j to aktywacje A_{ik} zmienią się o niezależną od k stałą addytywną A_i

$$\sum_j X_{ij}(p_j + p_{jk}) = \sum_j X_{ij} p_j + \sum_j X_{ij} p_{jk} = A_i + A_{ik}$$

Nie zmienią się zatem prawdopodobieństwa P_{ik} ani strata L. Strata L jest więc niezmiennicza ze względu na zmiany parametrów p_{jk} o niezależne od k stałe addytywne p_j. Innymi słowy w przestrzeni parametrów, względem których minimalizujemy funkcję, istnieją kierunki, czyli długie doliny, wzdłuż których ta funkcja się nie zmienia. W klasycznym podejściu do minimalizacji taką sytuację uważa się za niepożądaną.

9.3 Entropia krzyżowa

Dana jest zmienna losowa k oraz znormalizowane rozkłady prawdopodobieństwa p_k i q_k. Entropia krzyżowa

$$H = -\mathbb{E}_q[\log p] = -\sum_k q_k \log p_k$$

jest tym większa im bardziej rozkład p_k odbiega od rozkładu q_k. Jeżeli dla jakiegoś k_i rozkład $q_k = \delta_{kk_i}$, to

$$H = -\sum_k \delta_{kk_i} \log p_k = -\log p_{k_i}$$

W regresji logistycznej funkcja straty wynosi

$$L = -\sum_i \log P_{ik_i} = \sum_i \left(-\sum_k \delta_{kk_i} \log P_{ik} \right) = \sum_i \left(-\sum_k Q_{ik} \log P_{ik} \right) = \sum_i H_i$$

P_{ik} oraz Q_{ik} są odpowiednio wyliczonym z modelu oraz rzeczywistym prawdopodobieństwem przynależności próbki i do klasy k. Macierz Q_{ik} nazywa się też reprezentacją one hot. H_i jest więc entropią krzyżową wyliczonego rozkładu prawdopodobieństwa P_{ik} względem rzeczywistego rozkładu Q_{ik} dla pojedynczej próbki i zaś strata L jest sumą tych entropii po wszystkich próbkach.

10 Dokładność klasyfikacji

11 Podział danych na porcje

12 Zbiór uczący i testowy

13 Sieć wielowarstwowa w pełni połączona MLP

13.1 Powierzchnia klasyfikacji w regresji logistycznej

W regresji logistycznej prawdopodobieństwo przynależności próbki i do klasy k wynosi

$$P_{ik} = \frac{\exp A_{ik}}{\sum_{k'} \exp A_{ik'}} \quad A_{ik} = \sum_j X_{ij} p_{jk}$$

Dla uproszczenia mówiąc o jednej próbce opuszczamy index i

$$P_k = \frac{\exp A_k}{\sum_{k'} \exp A_{k'}} \quad A_k = \sum_j X_j p_{jk}$$

W przypadku dwóch klas

$$P_0 = \frac{\exp A_0}{\exp A_0 + \exp A_1} \quad P_1 = \frac{\exp A_1}{\exp A_0 + \exp A_1}$$

Powierzchnia rozgraniczająca klasy $P_0 = P_1 = 1/2$

$$\frac{\exp A_0}{\exp A_0 + \exp A_1} = \frac{\exp A_1}{\exp A_0 + \exp A_1} = \frac{1}{2}$$

czyli $A_0 = A_1$ czyli

$$\sum_j X_j p_{j0} = \sum_j X_j p_{j1}$$

czyli

$$\sum_j X_j (p_{j0} - p_{j1}) = 0$$

Jest to równanie płaszczyzny w przestrzeni cech X_j. W przypadku dwóch klas powierzchnią klasyfikacji jest więc płaszczyzna. W przypadku większej liczby klas powierzchnia klasyfikacji składa się z wycinków płaszczyzn. Regresja logistyczna jest więc klasyfikatorem liniowym.

13.2 Parametry a wagi i obciążenia

Jeżeli X_{ij} jest macierzą design to

$$A_{ik} = \sum_j X_{ij} p_{jk} = X_{i0} p_{0k} + X_{i1} p_{1k} + X_{i2} p_{2k} + ... = p_{0k} + X_{i1} p_{1k} + X_{i2} p_{2k} + ...$$

ponieważ pierwsza kolumna macierzy design zawiera same jedynki. Macierz danych to macierz design bez pierwszej kolumny. Jeżeli X_{ij} jest macierzą danych to

$$A_{ik} = p_{0k} + X_{i0} p_{1k} + X_{i0} p_{2k} + ... = b_k + X_{i0} w_{0k} + X_{i1} w_{1k} + ... = b_k + \sum_j X_{ij} w_{jk}$$

Współczynniki w_{jk} nazywamy wagami zaś wyrazy wolne b_k obciążeniami.

18 Splot

18.1 Splot a korelacja

Splotem sygnału $x(t)$ z filtrem $f'(\tau)$ nazywamy wielkość

$$(x \operatorname{conv} f')(t) = \int_{-\infty}^{\infty} x(t-\tau')f'(\tau')\mathrm{d}\tau'$$

Podstawiając $\tau = -\tau'$ oraz $f(\tau) = f'(-\tau) = f'(\tau')$ otrzymujemy

$$(x \operatorname{conv} f')(t) = \int_{-\infty}^{\infty} x(t-\tau')f'(\tau')\mathrm{d}\tau' = \int_{-\infty}^{\infty} x(t+\tau)f(\tau)\mathrm{d}\tau = (x \operatorname{corr} f)(t)$$

Ostatnią wielkość nazywamy korelacją sygnału x z filtrem f. Widać, że splot z filtrem jest równy korelacji z filtrem odbitym względem osi rzędnych. Ze względów historycznych w sieciach neuronowych zamiast splotu używa się korelacji, ale nazywa się ją splotem i oznacza gwiazdką

$$(x * f)(t) = \int_{-\infty}^{\infty} x(t+\tau)f(\tau)\mathrm{d}\tau$$

18.2 Splot jednowymiarowy

Rozważmy teraz dyskretny sygnał X_j, gdzie $-\infty < j < +\infty$, orazdyskretny filtr F_l, gdzie $-\infty < l < +\infty$. Ich dyskretny splot to

$$(X * F)_j = \sum_{-\infty \le l < \infty} X_{j+l}F_l \quad -\infty < j < +\infty$$

Na ogół filtr F_l przyjmuje wartości niezerowe tylko w skończonym przedziale indeksów l, najczęściej od 0 włącznie do pewnego L wyłącznie. Mówimy wtedy, że jest to filtr o długości L, a wyrażenie na splot przyjmuje postać

$$(X * F)_j = \sum_{0 \le l < L} X_{j+l}F_l \quad -\infty < j < +\infty$$

Indeks j przebiega tu od $-\infty$ do $+\infty$, podczas gdy w sieciach mamy do czynienia ze skończoną liczbą cech, indeksowanych od 0 włącznie do pewnego J wyłącznie. W takiej sytuacji indeks $j + l$ musi się zawierać w granicach od 0 włącznie do J wyłącznie. Zatem j może przyjmować wartości od 0 włącznie do $J - L + 1$ wyłącznie

$$(X * F)_j = \sum_{0 \le l < L} X_{j+l}F_l \quad 0 \le j < J - L + 1$$

Wynik splotu jest więc sygnałem o długości $J - L + 1$, zawierającym tyleż nowych cech.

18.3 Splot dwuwymiarowy

Rozważmy teraz sygnał dwuwymiarowy $X_{j_r j_c}$, gdzie $0 \le j_r < J_r$ oraz $0 \le j_c < J_c$, zaś subskrypty r i c oznaczają odpowiednio wiersze i kolumny. Typowym przykładem jest obraz o J_r wierszach i J_c kolumnach. Filtr $F_{l_r l_c}$ ma niezerowe wartości dla $0 \le l_r < L_r$ i $0 \le l_c < L_c$. Splot jest wtedy dwuwymiarowym sygnałem

$$(X * F)_{j_r j_c} = \sum_{0 \le l_r < L_r} \sum_{0 \le l_c < L_c} X_{(j_r+l_r)(j_c+l_c)}F_{l_r l_c} \quad 0 \le j_r < J_r - L_r + 1, 0 \le j_c < J_c - L_c + 1$$

Jeżeli $L_r = L_c$ to filtr nazywamy kwadratowym. W praktyce spotykamy prawie wyłączne filtry kwadratowe i do tego przypadku się ograniczymy. Przyjmiemy więc oznaczenie $L = L_r = L_c$.

19 Filtr Sobla do wykrywania krawędzi

20 Sieci splotowe

21 Gotowe warstwy i modele sekwencyjne

22 Warstwy pooling

23 Architektura LeNet5

24 Powierzchnia decyzyjna w dwóch wymiarach (numpy)

25 Przeuczenie

26 Przechowywanie parametrów modelu na dysku

27 Wczesny stop

28 Regularyzacja

29 Warstwy dropout

30 Tryb treningowy i ewaluacyjny

31 Normalizacja danych wejściowych

32 Warstwy normalizacyjne

33 Moduł do obróbki obrazów PIL

34 Zliczanie obiektów na rysunku

35 Zbiór i ładowarka danych

36 Duże zbiory danych

37 Uczenie sieci na dużych zbiorach danych

38 Zbiór danych do klasyfikacji kolorowych zdjęć CIFAR10

39 Architektura VGG

39.1 Padding

Zgodnie z powyższym wzorem jeżeli obraz $X_{j_r j_c}$ ma J_r wierszy i J_c kolumn, zaś filtr $F_{l_r l_c}$ ma rozmiar L, to ich splot ma $J_r - L + 1$ wierszy i $J_c - L + 1$ kolumn. Splot jest więc mniejszy od oryginału. Często jest jednak pożądane by rozmiary splotu były takie same jak oryginalnego obrazu. Aby to osiągnąć stosuje się tak zwany padding. Dla ustalenia uwagi rozważmy filtr o często spotykanym rozmiarze $L = 3$. W takiej sytuacji splot ma $J_r - 3 + 1 = J_r - 2$ wierszy i $J_c - 3 + 1 = J_c - 2$ kolumn, czyli jest o 2 wiersze i 2 kolumny mniejszy od oryginanału. Padding polega na tym, że do oryginalnego obrazu $X_{j_r j_c}$ dodajemy sztucznie wiersze o $j_r = -1$ oraz $j_r = J_r$ i kolumny o $j_c = -1$ oraz $j_c = J_c$ i wypełniamy je w jakiś sposób, najczęściej stałą wartością zero. Następnie stosujemy nieco inny wzór na splot

$$(X * F)_{j_r j_c} = \sum_{0 \le l_r < 3} \sum_{0 \le l_c < 3} X_{(j_r + l_r - 1)(j_c + l_c - 1)} F_{l_r l_c} \quad 0 \le j_r < J_r, 0 \le j_c < J_c$$

Taki splot ma tyle samo wierszy i kolumn, co oryginalny obraz. Jeżeli $L = 5$ to dodajemy po dwa sztuczne wiersze i dwie sztuczne kolumny z każdej strony obrazu i tak dalej. Najczęściej stosujemy filtry o L nieparzystym. Sztucznie dodane wiersze i kolumny nie są naprawdę przechowywane w pamięci, lecz obliczenia są wykonywane tak, jakby rzeczywiście tam były.

58 Procesy decyzyjne Markowa

58.1 Środowisko, agent, epizod, wartość stanu

Prawdopodobieństwo znalezienia się w stanie s' i otrzymania nagrody r jeżeli będąc w stanie s wykonamy akcję a:

$$p(s', r|s, a)$$

Normalizacja

$$\sum_{s',r} p(s', r|s, a) = 1$$

Prawdopodobieństwo znalezienia się w stanie s' jeżeli będąc w stanie s wykonamy akcję a:

$$p(s'|s, a) = \sum_{r} p(s', r|s, a)$$

Prawdopodobieństwo otrzymania nagrody r jeżeli będąc w stanie s wykonamy akcję a:

$$p(r|s, a) = \sum_{s'} p(s', r|s, a)$$

Epizod rozpoczynamy w pewnym stanie S_0 i wykonujemy kolejne akcje

$$S_0 \to A_0 \to (S_1, R_1) \to A_1 \to (S_2, R_2) \to ...$$

Kolejne akcje podejmujemy zgodnie z przyjętą polityką. Polityka to prawdopodobieństwo wykonania akcji a jeżeli jesteśmy w stanie s:

$$\pi_i(a|s)$$

Na ogół epizod kończymy po znalezieniu się w jednym ze stanów końcowych s_1, s_2, ... Liczbę podjętych akcji T nazywamy długością epizodu. Ze względu na probabilistyczny charakter procesu epizody rozpoczęte od tego samego stanu mogą mieć różne długości. Ostatnia akcja A_{T-1} prowadzi do stanu S_T i nagrody R_T. Dla każdego stanu pośredniego S_t rozważamy sumę wszystkich nagród otrzymanych od opuszczenia tego stanu do końca epizodu:

$$G_t = R_{t+1} + ... + R_T$$

Wielkość tę nazywamy po angielsku return, gain, lub cumulated reward, zaś po polsku zyskiem lub skumulowaną nagrodą. Wartość stanu s to średni zysk po przejściu przez ten stan:

$$v_i(s) = \mathbb{E}_i \left[G_t|S_t = s\right]$$

Wartość oczekiwaną liczymy dla ustalonej polityki π_i więc wielkość ta zależy od polityki.

59 Wyznaczanie wartości stanów i akcji z równań Bellmana

59.1 Równania Bellmana dla zagadnienia predykcji

Wartość akcji a w stanie s to średni zysk po wykonaniu tej akcji w tym stanie:

$$q_i(s,a) = \mathbb{E}_i\left[G_t | S_t = s, A_t = a\right]$$

Wartość oczekiwaną liczymy dla ustalonej polityki π_i więc wielkość ta zależy od polityki. Zajmiemy się teraz wyznaczeniem wartości wszystkich stanów $v_i(s)$ oraz wszystkich akcji $q_i(s,a)$ dla zadanej polityki π_i. Jest to tak zwane zagadnienie predykcji. Wartości stanów $v_i(s)$ i akcji $q_i(s,a)$ są ze sobą powiązane. Po pierwsze:

$$v_i(s) = \mathbb{E}_i\left[G_t | S_t = s\right] = \sum_a \pi_i(a|s)\mathbb{E}_i\left[G_t | S_t = s, A_t = a\right] = \sum_a \pi_i(a|s) q_i(s,a)$$

Zatem wartość stanu to średnia wartość akcji w tym stanie. Po drugie zysk spełnia relację rekurencyjną:

$$G_t = R_{t+1} + R_{t+2} + ... + R_T = R_{t+1} + G_{t+1}$$

skąd:

$$\begin{aligned} q_i(s,a) &= \mathbb{E}_i\left[G_t | S_t = s, A_t = a\right] \\ &= \mathbb{E}_i\left[R_{t+1} + G_{t+1} | S_t = s, A_t = a\right] \\ &= \mathbb{E}_i\left[R_{t+1} | S_t = s, A_t = a\right] + \mathbb{E}_i\left[G_{t+1} | S_t = s, A_t = a\right] \end{aligned}$$

Dwa człony sumy są równe:

$$\mathbb{E}_i\left[R_{t+1} | S_t = s, A_t = a\right] = \sum_r p(r|s,a) r = \sum_{s',r} p(s',r|s,a) r$$

$$\mathbb{E}_i\left[G_{t+1} | S_t = s, A_t = a\right] = \sum_{s'} p(s'|s,a)\mathbb{E}_i\left[G_{t+1} | S_{t+1} = s'\right] = \sum_{s'} p(s'|s,a) v_i(s') = \sum_{s',r} p(s',r|s,a) v_i(s')$$

Ostatecznie

$$q_i(s,a) = \sum_{s',r} p(s',r|s,a)\left(r + v_i(s')\right)$$

Zatem wartość danej akcji w danym stanie to średnia nagroda za wykonanie tej akcji plus średnia wartość stanu, do którego ta akcja prowadzi. Wyprowadzone tu dwa związki:

$$v_i(s) = \sum_a \pi_i(a|s) q_i(s,a)$$

$$q_i(s,a) = \sum_{s',r} p(s',r|s,a)\left(r + v_i(s')\right)$$

to równania Bellmana na wartości stanów i akcji przy zadanej polityce.

59.2 Value iteration

W praktyce wartości stanów i akcji przy zadanej polityce wyznaczamy iteracyjnie metodą punktu stałego. Wychodzimy od dowolnych wartości akcji $q_i(s,a)$, najczęściej zerowych. Na ich podstawie z pierwszego równania Bellmana wyznaczamy wartości stanów:

$$v_i(s) = \sum_a \pi_i(a|s) q_i(s,a)$$

Stąd z drugiego równania Bellmana wyznaczamy wartości akcji w następnej iteracji:

$$q_{i+1}(s,a) = \sum_{s',r} p(s',r|s,a)\left(r + v_i(s')\right)$$

Czynności te powtarzamy do osiągnięcia zbieżności, czyli do momentu, gdy:

$$v_{i+1}(s) = v_i(s) \quad q_{i+1}(s,a) = q_i(s,a)$$

Równości te oznaczają, że tak wyznaczone $v_i(s)$ oraz $q_i(s,a)$ spełniają równania Bellmana, a zatem są poszukiwanymi wartościami stanów i akcji.

60 Wyznaczanie optymalnej polityki z równań Bellmana

60.1 Twierdzenie o ulepszaniu polityki

Celem jest znalezienie takiej polityki, która daje największe skumulowane nagrody, czyli największe wartości stanów. Okazuje się, że istnieje jedna optymalna polityka, która daje największe wartości wszystkich stanów jednocześnie. Załóżmy, że mamy pewną politykę π_i oraz odpowiadające jej wartości akcji $q_i(s,a)$. Intuicyjnie najlepiej jest wybierać akcję o największej wartości. Nazywa się to polityka zachłanna:

$$\pi_{i+1}(a|s) = \delta(a, \operatorname*{argmax}_{a'} q_i(s,a'))$$

Jeżeli kilka akcji ma takie same wartości, to można je wybierać z dowolnymi prawdopodobieństwami sumującymi się do jedności. W takiej sytuacji polityka zachłanna może być stochastyczna lub deterministyczna. Jeżeli jest tylko jedna akcja o największej wartości, to polityka zachłanna jest deterministyczna. Twierdzenie o ulepszaniu polityki mówi, że π_{i+1} daje nie mniejsze wartości wszystkich stanów niż π_i:

$$v_{i+1}(s) \geq v_i(s)$$

Narzuca się zatem następująca procedura iteracyjna. Zaczynamy od dowolnej polityki π_i. Znajdujemy dla niej wartości akcji q_i. Na ich podstawie wyznaczamy politykę zachłanną π_{i+1}. Czynności te powtarzamy wielokrotnie. Ponieważ liczba polityk deterministycznych jest skończona, więc po skończonej liczbie iteracji otrzymujemy równości:

$$v_{i+1}(s) = v_i(s) \quad q_{i+1}(s,a) = q_i(s,a) \quad \pi_{i+1}(a|s) = \pi_i(a|s)$$

Druga część twierdzenia o ulepszaniu polityki mówi, że równości te zachodzą jedynie jeśli π_i daje największe możliwe wartości v_i. Opisana procedura prowadzi zatem do polityki optymalnej. Politykę tę oraz odpowiadające jej wartości akcji i stanów oznaczamy gwiazdką:

$$v_*(s) \quad q_*(s,a) \quad \pi_*(a|s)$$

60.2 Policy iteration

W praktyce optymalną politykę wyznaczamy iteracyjnie metodą punktu stałego. Zaczynamy od dowolnych wartości akcji $q_i(s,a)$, najczęściej zerowych. Na ich podstawie wyznaczamy politykę zachłanną:

$$\pi_{i+1}(a|s) = \delta(a, \operatorname*{argmax}_{a'} q_i(s,a'))$$

Następnie obliczamy wartości stanów:

$$v_{i+1}(s) = \sum_a \pi_{i+1}(a|s) q_i(s,a)$$

Na ich podstawie znajdujemy wartości akcji w następnej iteracji:

$$q_{i+1}(s,a) = \sum_{s',r} p(s',r|s,a)\,(r + v_{i+1}(s'))$$

Czynności te powtarzamy do osiągnięcia zbieżności, czyli do momentu, gdy:

$$v_{i+1}(s) = v_i(s) \quad q_{i+1}(s,a) = q_i(s,a) \quad \pi_{i+1}(a|s) = \pi_i(a|s)$$

Zgodnie z drugą częścią twierdzenia o ulepszaniu polityki równości oznaczają, że tak wyznaczona $\pi_i(a|s)$ jest polityka optymalną, zaś $v_i(s)$ oraz $q_i(s,a)$ są odpowiadającymi jej wartościami stanów i akcji.

61 Gra w pchełki

62 Q-Learning

62.1 Równania Bellmana dla zagadnienia kontroli

Równania Bellmana na wartości stanów i akcji obowiązują dla dowolnej polityki. Dla polityki optymalnej przyjmują postać:

$$v_*(s) = \sum_a \pi_*(a|s) q_*(s,a)$$

$$q_*(s,a) = \sum_{s',r} p(s',r|s,a)\,(r + v_*(s'))$$

Oprócz tego dla polityki optymalnej zachodzi zależność:

$$\pi_*(a|s) = \delta(a, \operatorname*{argmax}_{a'} q_*(s,a'))$$

Podstawiając tę zależność do pierwszego równania Bellmana dostajemy:

$$v_*(s) = \max_a q_*(s,a)$$

Przy optymalnej polityce wartość stanu jest więc równa wartości najlepszej akcji, jaką w tym stanie można wykonać.

62.2 Podejście tabelaryczne

Dla polityki optymalnej zachodzą równania Bellmana:

$$v_*(s) = \sum_a \pi_*(a|s) q_*(s,a)$$

$$q_*(s,a) = \sum_{s',r} p(s',r|s,a)\,(r + v_*(s'))$$

oraz dodatkowo zależność:

$$v_*(s) = \max_a q_*(s,a)$$

Podstawiając tak zapisane $v_*(s)$ do drugiego równania Bellmana dostajemy:

$$q_*(s,a) = \sum_{s',r} p(s',r|s,a)\left(r + \max_{a'} q_*(s',a')\right)$$

Choć w ogólności Q-Learning działa dla środowisk probabilistycznych, to tutaj ograniczymy się do deterministycznych, czyli takich, że w każdym kroku stan końcowy s' i nagroda r są jednoznacznie wyznaczone przez stan początkowy s i podjętą akcję a:

$$p(s',r|s,a) = \delta\left(s' - s'(s,a)\right)\delta\left(r - r(s,a)\right)$$

Podstawiając to do uzyskanego wyżej równania otrzymujemy:

$$q_*(s,a) = r(s,a) + \max_{a'} q_*(s'(s,a),a')$$

Jest to podstawowe równanie metody Q-Learning. Rozwiązujemy je iteracyjnie metodą punktu stałego. Wychodzimy od dowolnych $q_i(s,a)$, najczęście zerowych, i wyznaczamy kolejne przybliżenie:

$$q_{i+1}(s,a) = r(s,a) + \max_{a'} q_i(s'(s,a),a')$$

Czynności te powtarzamy do uzyskania równości:

$$q_{i+1}(s,a) = q_i(s,a)$$

Równość oznacza, że $q_i(s,a)$ spełnia równanie metody Q-Learning. Powyższą regułę iteracyjną można stosować do całej macierzy $q_{i+1}(s,a)$. W praktyce jednak postępujemy inaczej. Wybieramy losowo stan s oraz akcję a i z reguły iteracyjnej wyliczamy tylko jeden element $q_{i+1}(s,a)$. Czynności te powtarzamy dla innej losowej pary stan-akcja. Jeżeli po odpowiednio dużej liczbie iteracji uda nam się uzyskać zbieżność, czyli powyższą równość dla każdej pary stan-akcja, to znaczy, że równość ta zachodzi dla całej macierzy, czyli uzyskane $q_i(s,a)$ spełnia równanie metody Q-Learning.

63 Uczenie epizodyczne

64 Polityka epsilon-zachłanna

65 Znajdowanie drogi w labiryncie

66 Nagroda dyskontowana

66.1 Czynnik dyskontujący

Zysk w zagadnieniach epizodycznych:

$$G_t = R_{t+1} + ... + R_T$$

Zysk w zagadnieniach ciągłych:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ...$$

gdzie $\gamma \in [0, 1]$ to czynnik dyskontujący, po angielsku discount factor. Im ten czynnik mniejszy tym mniejsze znaczenie mają nagrody otrzymane w przyszłości. Związek rekurencyjny:

$$G_t = R_{t+1} + \gamma G_{t+1}$$

Definicje wartości stanów i akcji pozostają bez zmian:

$$v_i(s) = \mathbb{E}_i \left[G_t | S_t = s \right]$$

$$q_i(s, a) = \mathbb{E}_i \left[G_t | S_t = s, A_t = a \right]$$

Równania Bellmana na wartości stanów i akcji przybierają postać:

$$v_i(s) = \sum_a \pi_i(a|s) q_i(s, a)$$

$$q_i(s, a) = \sum_{s', r} p(s', r|s, a) \left(r + \gamma v_i(s') \right)$$

Polityka zachłanna:

$$\pi_{i+1}(a|s) = \delta(a, \operatorname*{argmax}_{a'} q_i(s, a'))$$

Polityka optymalna to taka, która maksymalizuje wartości wszystkich stanów. Polityka optymalna jest zachłanna względem odpowiadających jej wartości akcji:

$$\pi_*(a|s) = \delta(a, \operatorname*{argmax}_{a'} q_*(s, a'))$$

$$v_*(s) = \max_a q_*(s, a)$$

Równanie metody Q-Learning:

$$q_*(s, a) = r(s, a) + \gamma \max_{a'} q_*(s'(s, a), a')$$

67 Utrzymywanie pionowego pręta na palcu