Systemy Wspomagania Decyzji

Wybór samochodu przy użyciu sieci SOM

Opracowali:

Adrian Zybała

Wojciech Zub

1. **Opis problemu**

Celem aplikacji jest wspomaganie użytkownika przy podjęciu decyzji o wyborze samochodu. Użytkownik chce kupić samochód jak najlepiej spełniający jego oczekiwania lecz nie ma preferencji co do marki czy modelu. Wie natomiast jakich parametrów i wartości oczekuje od przyszłego pojazdu.  
W opracowanym rozwiązaniu przyjęte zostało ograniczenie polegające na traktowaniu wartości mniejszych i większych od preferowanych jako w równym stopniu złych.

1. **Sposób opisu samochodu**

Każdy samochód opisany jest zestawem parametrów w postaci wektora liczb. Liczby te są rzutowane na przedział 0-1 (np. w przypadku ceny, roku produkcji, przebiegu), wartości logiczne reprezentowane są jako 0/1 (np. w przypadku posiadania/ nie posiadania ABS, klimatyzacji). Informacje o marce i modelu są przechowywane w systemie tylko w celu prezentacji wyników i nie uczestniczą w procesie uczenia.

Baza danych o samochodach przechowywana jest w pliku tekstowym o następującym formacie:

Marka; Model; Cena; Rok produkcji; Przebieg; Skrzynia(czy automatyczna); Moc; Pojemnosc silnika; Paliwo(czy diesel); Drzwi(czy 4-drzwiowy); dodatki(ABS, klimatyzacja, centralny zamek itd.);

1. **Sieć SOM**

Sieć Kohonena została nazwana przez jego twórcę samoorganizującym odwzorowaniem (Self-Organizing Map - SOM) lub samoorganizującym odwzorowaniem cech (Self-Organizing Feature Map - SOFM) i jest najbardziej popularnym typem sieci określanej mianem samoorganizującej.

W sieci tej mamy do czynienia z uczeniem konkurencyjnym. Oznacza to, że użytkownik ma do dyspozycji jedynie wzorce wejściowe, nie posiada natomiast żadnych wzorców wyjścia. Zadaniem sieci w trakcie procesu uczenia jest wytworzenie takich wzorców. Sieć ma na celu utworzenie takiej struktury, która w najlepszy sposób będzie odwzorowywała zależności w przestrzeni wektorów wejściowych.

Każdy neuron jest powiązany z pozostałymi neuronami pewną relacją sąsiedztwa, która określa strukturę sieci. Przykładowe topologie sieci to siatka prostokątna, siatka heksagonalna, łańcuch otwarty i łańcuch zamknięty (w tym wypadku siatka NxN). Neurony można opisać za pomocą wektorów wag dla wszystkich wejść.

* Parametry sieci

Sieć charakteryzuje się parametrami:

* u – parametr uczenia sieci (0-1)
* l – maksymalna promień uczenia

W procesie samoorganizacji na każdym etapie należy wyłonić zwycięzcę - neuron, którego wektor wagowy różni się najmniej od wektora na wejściu x. Aby móc określić, która odległość jest najmniejsza należy dobrać odpowiednią metrykę, w jakiej mierzona będzie odległość między wektorem v1 a v2. W tym wypadku jest to miara euklidesowa.

Początkowe wektory wag są losowane i są one znormalizowane.

Sieć opisywana jest funkcjami:

* d(v1,v2) – funkcja odległości między wektorami lub neuronami
* G(n1,n2) – funkcja sąsiedztwa między neuronami (Gaussa lub prostokątna)

1. **Proces uczenia**

Proces uczenia składa się z wielu iteracji, w każdej z nich parametry u i l zmniejszają się, zbiór wejściowy jest mieszany a wektor wejściowy normalizowany.

Po pobudzeniu sieci wektorem wejściowym X podczas współzawodnictwa wygrywa neuron, którego wagi najmniej różnią się od odpowiednich składowych tego wektora. Zwycięzca, neuron w-ty spełnia relację:

Wszystkie neurony spełniające zależność d(wi,z) < l aktualizują swoje wagi według funkcji:

We wzorze tym oddzielono współczynnik uczenia i każdego neuronu od jego odległości względem wektora z, uwzględnionej przez funkcję sąsiedztwa G(wi,z). Wagi neuronów spoza sąsiedztwa nie podlegają zmianom.

1. **Interpretacja wyników**

* Neurony na wyjściach zwracają odległości swoich wektorów od wektora wejściowego z danymi podanymi od użytkownika.
* Jako wynik zapytania szeregujemy neurony pod względem wartości ich wyjść.
* W tym momencie nadal nie otrzymujemy danych o pojazdach
* Po nauczeniu sieci dla każdego samochodu z bazy znajdujemy neuron zwycięski i przypisujemy mu dany samochód.
* Wybieramy neurony w kolejności z listy wynikowej dopóki nie wypiszemy wystarczającej liczby samochodów.
* Dla każdego wybranego neuronu wypisujemy wszystkie samochody mu przypisane.

1. **Możliwe zmiany**

Aplikacje można zmodyfikować, wzbogacić o:

* Ściąganie danych na bieżąco, najlepiej z bazy danych (parsowanie strony jest mało wydajne)
* Rozszerzenie listy parametrów (muszą dać się zapisać liczbowo i rzutować na przedział 0-1)
* Zmiana funkcji odległości, dodanie wag lub minimalizacja parametru