Systemy Wspomagania Decyzji

Wybór samochodu przy użyciu sieci SOM

Opracowali:

Adrian Zybała 179206

Wojciech Zub 179353

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 odnośnie 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 odległych od optymalnych.

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 poprzez wartości 0 i 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 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 wyjściowych. 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, 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) będący mnożnikiem różnicy pomiędzy wektorem wag neuronu uczone a wektorem wejściowym.
* l – maksymalna promień uczenia wyznaczający okrąg wokół neuronu zwycięskiego, wewnątrz którego wagi neuronów są zmieniane, wykorzystywany również do obliczenia mnożnika w przypadku, kiedy korzystamy z funkcji sąsiedztwa Gaussa.

Początkowe wektory wag są losowane a następnie normalizowane.

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.

Pełny opis sieci otrzymujemy dopiero przedstawiając wybrane funkcje:

* d(v1,v2) – funkcja odległości między wektorami lub neuronami (w naszej sieci wykorzystujemy odległość euklidesową)
* G(n1,n2) – funkcja sąsiedztwa między neuronami (nasza aplikacja umożliwia wybór funkcji Gaussa lub prostokątnej)

1. **Proces uczenia**

Proces uczenia składa się z wielu iteracji, w każdej z nich parametry u i l zmniejszają się a kolejność wektorów w zbiorze wejściowym jest mieszana. Każdy wektor wejściowy musi być znormalizowany dla optymalnego działania sieci.

Po pobudzeniu sieci wektorem wejściowym X podczas współzawodnictwa wygrywa neuron, którego wagi najmniej różnią się od odpowiednich składowych tego wektora. W tym celu wykorzystywana jest funkcja odległości między wektorami. Zwycięzca, neuron w-ty spełnia relację:

Wszystkie neurony znajdujące się w okręgu wyznaczonym przez promień l dla danej iteracji (d(wi,z)<l) aktualizują swoje wagi według funkcji:

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

Najpopularniejszymi funkcjami sąsiedztwa wykorzystywanymi w sieciach Kohonena są funkcje prostokątna i Gaussa. Funkcja prostokątna przyjmuje wartość 1 dla wszystkich neuronów znajdujących się w okręgu wyznaczonym przez parametr l oraz wartość 0 dla wszystkich pozostałych. Wartość funkcji Gaussa dla odległości dwóch neuronów możemy natomiast obliczyć ze wzoru:

1. **Interpretacja wyników**

Po nauczeniu sieci neurony na wyjściach zwracają odległości swoich wektorów od wektora wejściowego z danymi podanymi od użytkownika. Jako wynik zapytania możemy szeregować neurony pod względem wartości ich wyjść. Nadal jednak nie otrzymujemy żadnych informacji o faktycznych pojazdach z bazy danych.

Aby uzyskać oczekiwane informacje o podobnych pojazdach w bazie musimy wzbogacić sieć o pewne informacje. W już nauczonej sieci dla każdego samochodu z bazy danych znajdujemy neuron zwycięski i przypisujemy mu dany samochód.

Po wprowadzeniu przez użytkownika oczekiwanych danych tworzymy listę samochodów wybierając kolejne neurony z uporządkowanej listy wynikowej. Dopóki nie znajdziemy wystarczającej liczby samochodów, dla każdego wybranego neuronu wypisujemy wszystkie przypisane mu samochody.

1. **Korzystanie z aplikacji**

Po uruchomieniu aplikacji użytkownik najpierw musi nauczyć sieć. W przygotowanej do tego zakładce użytkownik może podać wartości maksymalne (początkowe) i minimalne (końcowe) parametrów u i l, liczbę iteracji procesu uczenia oraz czy korzystamy z funkcji sąsiedztwa Gaussa (w przeciwnym wypadku używana jest funkcja prostokątna). Odpowiednie przyciski w tej zakładce służą do wczytania zbioru uczącego z pliku „cars.txt”, losowania początkowych wag oraz rozpoczęcia uczenia sieci. Tutaj też w każdej chwili użytkownik może zapisać obecną sieć do pliku lub wczytać już zapisaną.

Inna zakładka służy do wyszukiwania samochodów w już nauczonej sieci. W górnej części znajdują się modyfikowalne pola odpowiadające parametrom pojazdów oraz przycisk „Szukaj” zwracający wyniki. Lista podobnych samochodów pojawia się w środkowej części, po lewej możemy także zaobserwować mapę neuronów pokolorowanych w odcieniach szarości w zależności od odległości ich wag od wartości podanych przez użytkownika. Po wybraniu pojazdu ze środkowej listy, neuron mu przypisany zaznaczany jest kolorem czerwonym, natomiast w dolnej części aplikacji pokazywane są dokładne parametry dotyczące wybranego samochodu.

1. **Możliwe zmiany**

Aplikacje można zmodyfikować, wzbogacić o:

* Ściąganie danych na bieżąco, najlepiej z gotowej bazy danych (parsowanie strony jest bardzo wymagające czasowo)
* Rozszerzenie listy parametrów (muszą dać się zapisać liczbowo i rzutować na przedział 0-1)
* Zmiana funkcji odległości – np. skorzystanie z miary Manhattan lub maksimum
* Dodanie wag lub minimalizacja parametru (po części wykonywalne obecnie poprzez ustawienie parametru oczekiwanego na wartość 0)