

Barbara Pankiewicz
nauczyciel fizyki
III Liceum Ogólnokształcące
w Zamościu
ul. Kilińskiego 15
22-400 Zamość

Emergentne właściwości sztucznych sieci neuronowych

**Opracowała:
Barbara Pankiewicz**

Zamość, 2001

WSTĘP

Od czasu pierwszych prymitywnych maszyn liczących ich twórcy i użytkownicy dokładali starań, aby przestały pełnić funkcję automatycznych kalkulatorów i stały się maszynami „myślącymi”. Jednak już samo pojęcie „maszyny myślącej” jest dyskusyjne. Rozmaitość definicji popularnego określenia „sztuczna inteligencja” jest zdumiewająca. Metody realizacji procesów myślowych w sposób imitujący przypuszczalne mechanizmy ludzkiego intelektu są nie mniej różnorodne. Sieci neuronowe reprezentują jedno z możliwych rozwiązań.

Sieci neuronowe można traktować jako nowoczesne systemy obliczeniowe, które przetwarzają informacje wzorując się na zjawiskach zachodzących w mózgu człowieka. Informacje te mają charakter danych numerycznych, na podstawie których sieć neuronowa np. może posłużyć jako model obiektu o zupełnie nieznanym charakterystyce.

1. Neuron a perceptron

Podstawowym elementem systemu nerwowego jest komórka nerwowa nazywana **neuronem**.

Neuron biologiczny jest komórką zdolną wykonywać pewne obliczenia. Jest on pobudzany przez jedno lub wiele wejść i wytwarza wyjście wysyłane do innych neuronów. Rzeczywisty związek między wejściem a wyjściem może być niezmiernie złożony. Jeden neuron przekazuje pobudzenie innym neuronom przez złącza nerwowe zwane **synapsami**.

Aby przedstawić model neuronu nawiązujący do pierwszych prób sformalizowania opisu działania komórki nerwowej wprowadźmy następujące oznaczenia:

u_1, \dots, u_N - sygnały wejściowe danego neuronu pochodzące od innych neuronów,

w_1, \dots, w_N - wagi synaptyczne,

y - sygnał wyjściowy neuronu,

v - wartość progowa.

Formuła opisująca działanie neuronu wyraża się zależnością:

$$y=1 \text{ gdy } \sum_{i=1}^N w_i u_i \geq v$$

$$y=0 \text{ gdy } \sum_{i=1}^N w_i u_i < v$$

Model może być zapisany w postaci:

$$y = f\left(\sum_{i=0}^N w_i u_i\right) \quad (1.1)$$

gdzie:

$$f(x)=1 \text{ gdy } x \geq 0 \quad (1.2)$$

$$f(x)=0 \text{ gdy } x < 0$$

oraz $w_0=v$, $u_0=-1$

Wzór (1.1) opisuje model neuronu. Model ten został sformułowany w roku 1943 przez McCullocha i Pittsa. Jako funkcję f można przyjąć nie tylko funkcję jednostkową (1.2), ale również inne funkcje progowe postaci:

$$f(x)=1 \text{ gdy } x > 0$$

$$f(x)=-1 \text{ gdy } x < 0$$

lub

$$f(x)=1 \text{ gdy } x > 1$$

$$f(x)=-1 \text{ gdy } x < -1$$

$$f(x) = x \text{ gdy } |x| \leq 1.$$

Model McCullocha - Pittsa jest punktem wyjścia do konstrukcji najprostszej

jednokierunkowej **sieci neuronowej** o nazwie **perceptron**. Sieć taką zaproponował Rosenblatt w 1958r. Perceptron był oparty na modelu biologicznym i umiał się uczyć.

Rosenblatt wprowadził wiele odmian perceptronu; ten najprostszy składa się z trzech warstw. Pierwsza to „siatkówka” wejściowa. Jest ona połączona z warstwą drugą, złożoną z tzw. „jednostek kojarzących”, które pełnią funkcje detektora cech. Wreszcie ta warstwa jest połączona z warstwą odpowiedzi wyjściowej. Jednostki (neurony) tego modelu generują sygnał wyjściowy równy sumie ważonej sygnałów wejściowych z uwzględnieniem progu. Uczenia perceptronu dokonuje się powtarzając prezentacje sygnałów próbki na wejściu „siatkówki”, obliczając wyjścia neuronów warstwy środkowej, używając tych wartości do obliczenia wyjść neuronów warstwy odpowiedzi, a następnie modyfikując wagi połączeń między warstwą środkową a warstwą odpowiedzi. Nie zmieniają się wagi łączące warstwę wejściową z warstwą środkową. A więc, chociaż sieć ma trzy warstwy, w rzeczywistości jest raczej siecią dwuwarstwową. Warstwa środkowa wykonuje wielkie zadanie przekształcenia obrazu wejściowego w celu wydobycia cech charakterystycznych. W praktyce stałe wagi łączące wejście z warstwą środkową mogą być ustawione na wykrywanie znanych cech albo mogą być wybrane przypadkowo. Rosenblatt udowodnił, że jeżeli perceptron może nauczyć się zbioru wzorców (przykładów) to ten algorytm uczenia jest zbieżny do zbioru wag, które realizują prawidłową odpowiedź dla każdego wzorca z danego zbioru.

2. Uczenie sieci - metoda wstecznej propagacji błędu

Sieć neuronowa uczy się na ogół na dwa sposoby. Najczęstsze jest **uczenie pod nadzorem**. Zbieramy wiele próbek, które odgrywają rolę przykładów. Każda próbka zbioru uczącego całkowicie określa wszystkie wejścia, jak również wyjścia wymagane przy prezentacji tych danych wejściowych. Dla każdej próbki porównujemy aktualny sygnał wyjściowy sieci z sygnałem wejściowym, który chcielibyśmy otrzymać. Po przetworzeniu całego podzbioru próbek uczących korygujemy wagi łączące neurony w sieci. Tę korekcję prowadzimy w taki sposób, żeby osiągnąć zmniejszenie się miary błędu działania sieci. Inną podstawową metodą uczenia jest **uczenie bez nadzoru**. Tak jak w przypadku uczenia pod nadzorem, mamy zbiór próbek sygnałów wejściowych. Ale sieci nie doprowadzamy do pożądanego wyjścia na te próbki. Zwykle zakładamy, że każdy sygnał wyjściowy pochodzi z jednej spośród kilku klas i że wyjście sieci identyfikuje klasę, do której należy dany sygnał wejściowy. Proces uczenia sieci polega na umożliwieniu wykrycia istotnych cech zbioru uczącego i wykorzystaniu ich do grupowania sygnałów wejściowych na klasy, które sieć potrafi rozróżnić. Modele zdolne do uczenia bez nadzoru są szczególnie cenne w pracach badawczych.

Pierwszą praktyczną metodą uczenia sieci wielowarstwowej jednokierunkowej była **metoda wstecznej propagacji błędu**. Algorytm wstecznej propagacji błędu jest najbardziej znany i najczęściej stosowany w dziedzinie sieci neuronowych. Opis tej metody w pracy Rumelharta i McClellanda (1986) wywołał duże zainteresowanie sztucznymi sieciami neuronowymi.

Przyczyna nazwania algorytmu propagacji wstecznej jest oczywista, gdyż błędy warstwy wyjściowej są sukcesywnie propagowane wstecz przez sieć.

W swej najbardziej podstawowej formie propagacja wsteczna jest tym, co programiści nazywają algorytmem **spadku gradientu**. Gradient funkcji wielu zmiennych jest kierunkiem najbardziej stromego zjeżdżania w dół. Mały krok w tym kierunku da w rezultacie maksymalny przyrost funkcji w porównaniu z jakimkolwiek innym kierunkiem. Ten sam krok

w przeciwnym kierunku da maksymalny spadek funkcji. Funkcją jest błąd sieci na zbiorze uczącym.

Obliczanie gradientu tej funkcji wydaje się mieć sens, zrobimy krok w kierunku przeciwnym (kierunek ujemnego gradientu) i powtarzamy to w miarę potrzeby. Prawie zawsze wykonujemy ruch w kierunku optymalnym w celu zmniejszenia błędu (przynajmniej lokalnie). Możemy oczekiwać, że dojdziemy do położenia błędu minimalnego dość szybko. Dokładna długość kroku, często zwana **współczynnikiem uczenia**, może okazać się krytyczna. Jeśli ta długość okaże się za mała, to zbieżność będzie niezwykle powolna. Jeżeli będzie za duża, to będziemy wykonywać gwałtowne skoki i nigdy nie osiągniemy minimum.

Istnieją dwa bardzo duże mankamenty tej metody. Pierwszy wynika z faktu, że gradient jest skrajnie lokalnym wskazaniem kierunku optymalnej zmiany funkcji. Nawet małe odchylenie może spowodować dużą różnicę kierunku gradientu. Te gwałtowne fluktuacje mogą wywołać poszukiwanie błędu minimalnego w sposób meandryczny, zwiększając odległość tysiące razy (czyli zmniejszając szybkość obliczeń) w stosunku do prostej drogi. Drugim problemem jest trudność w zorientowaniu się z góry, jak daleko posunąć się w kierunku ujemnego gradientu. Jeśli jesteśmy konserwatywni i robimy drobny krok, to niezwykle wiele czasu trzeba, aby wykonać obliczenia dla całości i wykonać obliczanie gradientu. Jeśli wykonamy za długi krok, to możemy doprowadzić w rzeczywistości do wzrostu błędu.

Proponowano dziesiątki modyfikacji podstawowego algorytmu propagacji wstecznej. Najbardziej znane są następujące:

- 1) Zmień współczynnik uczenia (długość kroku w kierunku ujemnego gradientu) w miarę postępów uczenia. Staraj się utrzymać jak najdłuższe kroki bez przeskoków.
- 2) Zmodyfikuj wzór określający pochodną funkcji aktywacji, aby otrzymać trochę większe wartości. Utrudni to uniknięcie w ekstremum wartości wag, gdy pochodne są małe, ucieczka staje się trudna.
- 3) Nie modyfikuj wag jednocześnie na wszystkich warstwach. Modyfikuj wagi między pierwszą (ukrytą) warstwą i oblicz ponownie aktywacje dla tej warstwy. Użyj tych samych aktywacji do obliczenia poprawek dla następnej warstwy itd. Ta metoda po raz pierwszy opisana w pracy Samada (1998) wymaga wiele szczęścia, ale bardzo przyspiesza zbieżność.

Wspólnym motywem tych udoskonalień jest próba bardziej racjonalnego wyjaśnienia wyników otrzymanych doświadczalnie, które są dodatkiem do i tak już empirycznego algorytmu.

3. Generalizacja

Ucząc sieć neuronową dążymy do tego, aby w przyszłości potrafiła ona uogólniać nabytą wiedzę na obce jej, choć podobne problemy. Tę umiejętność nazywa się **generalizacją**. Mówi się, że sieć neuronowa dobrze generalizuje, kiedy odpowiedzi udzielane przez nią dla zestawu danych testowych są prawidłowe lub zawierają się w granicach ustalonego przez nas błędu. Dane testowe muszą pochodzić z tej samej populacji co dane wykorzystane do uczenia sieci ale różne od nich. Proces nauczania można porównać do problemu aproksymacji funkcji nieliniowej. Jest to możliwe, gdyż wielowarstwowy perceptron z ciągłą funkcją aktywacji daje na wyjściu także funkcję ciągłą, więc samą sieć można rozważać jako nieliniowe odwzorowanie wejścia w wyjście. Przy takim rozumieniu generalizacja to po prostu wynik dobrej nieliniowej interpolacji danych wejściowych.

Sieć powinna być zaprojektowana tak, aby potrafiła generalizować nawet wtedy,

gdy dane wejściowe są zupełnie inne niż te użyte do jej uczenia.

Używając zbyt dużej ilości neuronów ukrytych można doprowadzić do efektu przetrenowania sieci. Wtedy sieć działa jak tablica, w której zapamiętane są prawidłowe odpowiedzi dla danych testowych, a nie następuje uogólnienie wiedzy o generującym te odpowiedzi odwzorowaniu.

Na jakość generalizacji mają wpływ trzy zasadnicze czynniki:

- 1) złożoność problemu,
- 2) architektura (budowa) sieci,
- 3) długość i jakość ciągu uczącego.

Na pierwszy z nich nie mamy żadnego wpływu. W praktyce najczęściej spotykamy się też z sytuacją, gdy nie mamy żadnego wpływu na wybór architektury sieci..

Pozostaje nam, w związku z tym, tylko możliwość takiego doboru długości ciągu uczącego, by uzyskać sieć o najlepszej generalizacji.

Sieć prawie dokładnie dostarcza generalizacji przy założeniu, że spełnione są dwa warunki:

- 1) ułamek błędów popełnionych na ciągu uczącym jest mały,
- 2) liczba przykładów użytych do nauki jest wystarczająco duża.

4. Predykcja

Jednym z najważniejszych zadań stawianych przed sieciami neuronowymi jest **predykcja**, czyli przewidywanie określonych danych wyjściowych na podstawie danych wejściowych, często także po samodzielnym ustaleniu związków łączących dane wyjściowe z wejściowymi. Ważną zaletą sieci neuronowych jako narzędzia prognozującego jest fakt, że w wyniku procesu uczenia sieć może nabyć zdolność przewidywania sygnałów wyjściowych bez konieczności stawiania w sposób jasny hipotez o naturze związku pomiędzy wejściowymi danymi i przewidywanymi wynikami, a więc sieć może nauczyć się prognozować sygnały wyjściowe także wtedy, gdy korzystający z niej nic nie wie o naturze związków łączących przesłanki z wnioskami.

Sieci neuronowe ze względu na dobre właściwości uogólniające dobrze nadają się do rozwiązywania różnego rodzaju zadań predykcyjnych, np. do predykcji obciążeń godzinnych polskiego systemu elektroenergetycznego.

Cechą charakterystyczną obciążeń ww. systemu jest powtarzalność wzorców charakteryzujących obciążenia godzinne, odpowiadające różnym typom dnia tygodnia i miesiąca. Każda godzina dnia charakteryzuje się swoją specyfiką. Duży wpływ na dokładność predykcji ma znajomość obciążenia z ostatniej godziny, która jest dobrym wskaźnikiem, co może zdarzyć się w godzinie następnej.

Przy predykcji obciążenia godzinnego sieć zawiera tylko jeden neuron wyjściowy, liniowy, dostarczający informacji, jakie będzie obciążenie systemu w następnej godzinie.

Liczba węzłów wejściowych sieci neuronowej zależy od czynników branych pod uwagę w procesie predykcji. Wiąże się to ściśle ze sposobem przygotowania danych wejściowych do sieci. Pierwszym czynnikiem uwzględnianym w procesie predykcji jest typ dnia. Przy rozróżnieniu 4 typów wystarczą 2 węzły do ich binarnego zakodowania: (0, 0) - sobota, (0, 1) - niedziela, (1, 0) - poniedziałek, (1,1) - pozostałe dni tygodnia. Przy uwzględnianiu dwóch typów dni: dzień roboczy i dzień świąteczny, wystarczy użycie jednego węzła wejściowego o przypisanej wartości zerowej odpowiadającej świętu i wartości jedynkowej odpowiadającej dnu robocemu.

W modelu predykcji uwzględnia się obciążenia systemu o danej godzinie i kilka godzin wstecz dla dni poprzedzających predykcje. Jeśli $p(i, t)$ oznaczy się pobór mocy systemu odpowiadający i -temu dnu o godzinie t , to zastosowany model nieliniowy można opisać równaniem:

$p(i, t) = f(W, y(i, t-1), y(i-1, t), \dots, y(i-1, t-k), y(i-2, t), \dots, y(i-d, t-k))$, w którym W prezentuje wektor wag sieci, $y(j, \tau)$ - Znane obciążenie systemu w dniu j -tym o godzinie τ , a k i d - liczbę odpowiednio godzin i dni wstecz, branych pod uwagę w procesie predykcji.

Z wielu eksperymentów numerycznych przeprowadzonych dla polskiego systemu elektroenergetycznego wynika, że najlepsze wyniki odpowiadały wektorom wejściowym 8- lub 9- elementowym (1 lub 2 węzły do kodowania typu dnia, 3 węzły dla obciążenia o godzinie t dla trzech ostatnich dni, 3 węzły dla obciążenia trzech ostatnich dni o godzinie $t-1$ i jeden węzeł dla obciążenia tego samego dnia o godzinie $t-1$).

Najtrudniejszy jest dobór liczby neuronów w warstwie ukrytej. Zbyt mała liczba neuronów ukrytych uniemożliwia zredukowanie błędu uczenia do odpowiednio niskiego poziomu. Zbyt duża ich liczba jest również zła ze względu na duży błąd uogólniania. Taka sieć nie przedstawiałaby w praktyce żadnej wartości.

Ostatni problem w przygotowaniu danych to podział danych na część uczącą (sprawdzającą) i testującą. Przy dostępnej bazie danych z lat 1993 i 1994 oraz dla skrócenia czasu uczenia trenowanie sieci przeprowadzono na połowie danych jednego roku, przy czym baza danych uczących obejmowała wszystkie sezony: lato, zimą, wiosną i jesień. Przy większej dostępnej bazie danych można odpowiednio powiększyć bazę uczącą uwzględniając w pierwszej kolejności dane najnowsze i odrzucając wielkości nietypowe, stanowiące istotne odchylenie od normy.

W efekcie proces prognozowania obciążenia można przedstawić w następujący sposób:

- dobór architektury sieci neuronowej;
- wybór danych uczących i struktury wektorów wejściowych;
- trening sieci neuronowej;
- testowanie sieci na zbiorze sprawdzającym i przeprowadzenie ewentualnie dalszego douczenia;
- użycie sieci jako predyktora obciążenia godzinnego w fazie odtworzeniowej (właściwy etap użytkowania);
- ewentualna adaptacja sieci po upływie pewnego okresu, np. roku.

Badania eksperymentalne predykcji na bazie danych polskiego systemu elektroenergetycznego z lat 1993 i 1994 przeprowadzono przy zastosowaniu programu Netteach.

ZAKOŃCZENIE

Sztuczne sieci neuronowe bywają na ogół lepsze od innych metod, gdyż odznaczają się zarówno mocnymi podstawami teoretycznymi, jak i użytecznością w praktyce. Dowolny problem, który można rozwiązać metodą tradycyjnego modelowania albo metodami statystycznymi może być na ogół skuteczniej rozwiązany przy użyciu sieci neuronowych dzięki ich emergentnym właściwościom, tj. zdolności do uczenia się, uogólniania nabytej wiedzy czy prognozowania oraz innym cechom takim jak zdolność do klasyfikacji czy redukcja zakłóceń. Sieć ma właściwość sztucznej inteligencji. Wytrenowana na wybranej grupie danych uczących potrafi skojarzyć nabytą wiedzę i wykazać dobre działanie nawet na danych nie uczestniczących w procesie uczenia. Sztuczne sieci neuronowe to przyszłość sieci komputerowych.

LITERATURA

1. Osowski S., *Sieci neuronowe w ujęciu algorytmicznym*, WNT, Warszawa 1996.
2. Tadeusiewicz R., *Sieci neuronowe*. Akademicka Oficyna Wydawnicza, Warszawa 1993.
3. Korbicz J., Obuchowicz A., Uciński D., *Sztuczne sieci neuronowe, podstawy i zastosowania*. Akademicka Oficyna Wydawnicza, Warszawa 1994.
4. Rutkowska R., Piliński M., Rutkowski L., *Sieci neuronowe, algorytmy genetyczne i systemy rozmyte*. Wydawnictwo Naukowe PWN, 1997.
5. Masters T., *Sieci neuronowe w praktyce*, WNT, Warszawa 1996.
6. Okrojek A., *Generalizacja i uwierzytelnianie* - referat Internet.