Prezentacja tematu pracy dyplomowej

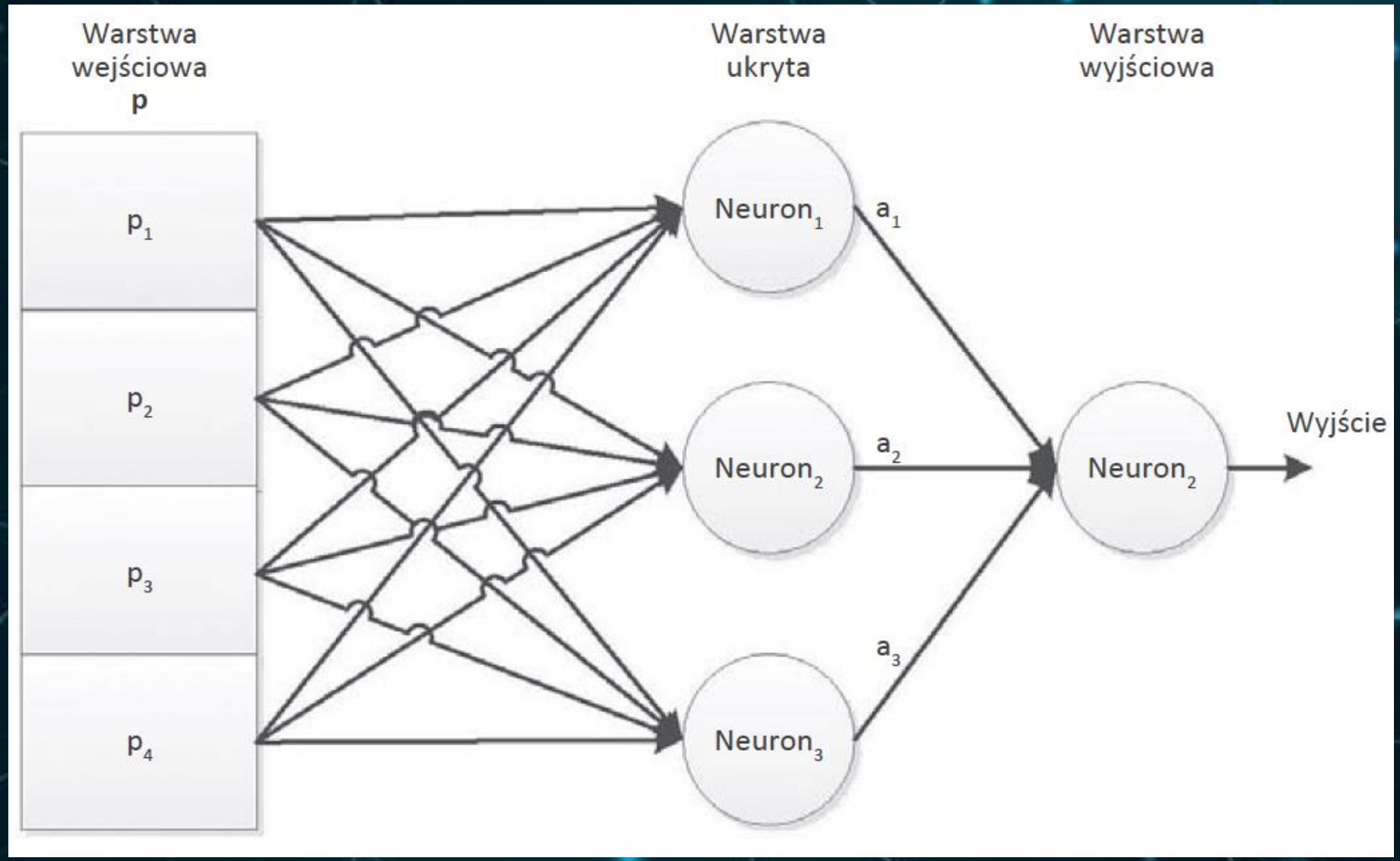
Autor: Patryk Lachowicz

Informacje o temacie

- Temat: Porównanie algorytmu ewolucji różnicowej z algorytmem strategii ewolucyjnej dla problemu liczenia wag w sieci neuronowej.
- Promotor: Pani dr inż. Joanna Kołodziejczyk
- Cel: Wyznaczenie wad i zalet poszczególnych algorytmów ewolucyjnych oraz różnic w wynikach dla problemu liczenia wag sieci neuronowej.

Krótko o sieciach neuronowych

Sieci neuronowe są to struktury składające się z połączonych ze sobą neuronów. Zawierają one 3 rodzaje warstw: jedna warstwa wejściowa(zawiera dane wejściowe np. dane z pomiarów), warstwy ukryte(w tych warstwach zachodzi proces uczenia się sieci) oraz jednej warstwy wyjściowej(prezentuje ona wyniki końcowe sieci). Każdy neuron z warstwy L połączony jest z każdym neuronem warstwy L+1, a każde takie połączenie posiada odpowiednią wagę. To właśnie te wagi są modyfikowane w procesie uczenia się. W bardziej skomplikowanych sieciach każdy neuron posiada również tzw. Tendencyjność (z angielskiego Bias), który również podlega modyfikacją.

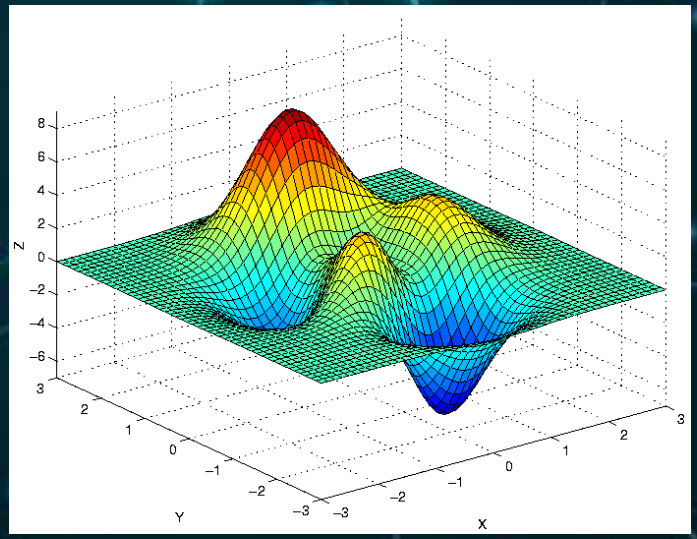

Warstwa wejściowa
p
Warstwa ukryta
Warstwa wyjściowa
p1
p2
p3
p4
Neuron1
Neuron2
Neuron3
a1
a2
a3
Neuron2
Wyjście

Uczenie się sieci neuronowych

Standardowy proces uczenia sieci neuronowych zakłada znajomość danych wejściowych X oraz odpowiadającym im wynikom Y. Dane otrzymane na wyjściu są więc porównywane ze znanymi nam danymi Y, a dokładnie wyliczany jest błąd E, będący różnicą wyników. Uczenie więc polega na odpowiednim modyfikowaniu wag, aż otrzymany błąd będzie odpowiednio niski.

Sposoby modyfikowania wag

Sposób w jaki wagi są modyfikowane jest tak naprawdę „sercem” sieci neuronowej. Problem znalezienia odpowiedniej wagi jest problemem znalezienia globalnego minimum gradientu prostego. Zwykłe sprawdzanie po kolei każdej możliwej kombinacji wag jest zdecydowanie wykluczone ze względu na wielowymiarowość problemu. Powszechnie stosowanymi metodami są np. propagacja wsteczna z użyciem metoda gradientu sprzężonego, zmodyfikowana propagacja wsteczna lub Marquadt algorithm. Wiele prac podczas eksperymentów przyjmowało małej wielkości sieci, gdzie niektóre metody gradientowe są ekstremalnie efektywne, co pozbawiało sensu testowanie metod ewolucyjnych. Metody ewolucyjne wyróżniają się w tym przypadku możliwością wpływania np. na strukturę sieci lub zasad uczenia, gdzie metody gradientowe operują tylko na wagach i preferencjach(bias).



Algorytmy ewolucyjne

Są to algorytmy stosowane do rozwiązywania problemów optymalizacyjnych, a więc i dobór odpowiednich wag. Operują one na populacji N osobników, gdzie każdy osobnik jest potencjalnym rozwiązaniem problemu. W zadaniu uczenia sieci neuronowej osobnik, to zestaw wag. Przetwarzanych jest więc jednocześnie wiele zestawów wag, poprzez dokonywanie na nich odpowiednich metod mutacji i krzyżowania. W rozwiniętych sieciach przestrzeń potencjalnych rozwiązań jest zwykle tak duża, że nie jest możliwe w dostatecznie krótkim czasie sprawdzenie wszystkich rozwiązań. Uzasadnione więc staje się stosowanie technik probabilistycznych, a algorytmy ewolucyjne są właśnie jedną z nich.

Algorytm ewolucji różnicowej

Algorytm ewolucji różnicowej (Differential Evolution) jest to prosty, a jednocześnie skuteczny algorytm ewolucyjny rozwiązujący problem globalnej optymalizacji ciągłej. Jest on prosty w implementacji oraz bardzo efektywny.

Polega on na wylosowaniu populacji N osobników, czyli zestawów wag. W zależności od wariantu algorytmu wybierane są odpowiednie osobniki(x_{r1},x_{r2},…,x_{rj}), a następnie na ich podstawie tworzony jest osobnik zmutowany u_i, który następnie podlega krzyżowaniu z aktualnym osobnikiem x_i. Otrzymany w ten sposób osobnik o_i porównywany jest z x_i poprzez wyliczenie błędów E danej sieci dla tych zestawów wag. Do następnej generacji wybierany jest lepszy osobnik. Cały proces jest powtarzany, aż nie osiągniemy odpowiednio niskiego błędu lub założoną z góry ilość generacji.

Pseudokod DE

Zainicjować wartości parametrów C_r, F i N_p
Zainicjować populację $\mathbf{X} \leftarrow \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{N_p}\}$
while warunek zatrzymania nie spełniony **do**
  for all $i \in \{1, 2, ..., N_p\}$ **do**
    # reprodukcja
    $\mathbf{x}_{i_1} \leftarrow$ reprodukcja $(i, \mathbf{X})$
    # mutacja różnicowa
    $\mathbf{u}_i \leftarrow$ mutacja $(F; \mathbf{x}_{i_1}, \mathbf{X})$
    # krzyżowanie wymieniające
    $\mathbf{o}_i \leftarrow$ krzyżowanie $(C_r; \mathbf{x}_i, \mathbf{u}_i)$
    # lokalna selekcja
    if $f(\mathbf{o}_i) \leq f(\mathbf{x}_i)$ **then**
      $\mathbf{x}_i \leftarrow \mathbf{o}_i$
    end if
  end for
end while
return $\arg\min_{\mathbf{x}_i} f(\mathbf{x}_i)$

Algorytm strategii ewolucyjnych

Algorytm strategii ewolucyjnych (Evolution Strategy) jest to również popularna metoda ewolucyjna. W przypadku tego algorytmu każdemu wektorowi x, czyli naszemu potencjalnemu rozwiązaniu, przyporządkujemy pomocniczy wektor s. Para (x,s) stanowi wtedy kod genetyczny osobnika.

Głównymi różnicami tego algorytmu w porównaniu do DE jest mutacja, krzyżowanie oraz ilość osobników na których pracujemy w każdej iteracji. Dzięki mutacji i krzyżowaniu do N osobników rodzicielskich dopisujemy M nowych osobników potomnych. Pod koniec spośród N+M osobników wybieramy N najlepszych, w tym przypadku posiadających najmniejszy błąd E.

Pseudokod ES

```
P = initialize(mi)                          // inicjalizacja wartości początkowych
while not terminalContition():               // główna pętla
        offspring = []                       // wyczyść tablice wyselekcjonowanych osobników
        for i:1 .. LAMBDA:                   // wygeneruj λ potomków
                parents=marriage ( P , p)
                s = sReconbination(parents)  // rekombinacja na parametrach endogennych
                y = yRecombination(parents)  // rekombinacja na wektorze parametrów
                s = sMutation(s)             // mutacja parametrów endogennych
                y = yMutation(s, y)          // mutacja wektora parametrów
                f = F(y)                     // obliczenie funckji przystosowania
                 offspring = [offspring, osobnik(y, s, f)] // dodaj osobnika do tablicy osobników
         end
          P = selection(offspring, P , mi)   //  uaktualnij populację osobników
end
```

Porównanie algorytmów

Udowodnienie wyższości jednego algorytmu nad drugim, nawet dla ograniczonego problemu nie jest łatwym zadaniem. Algorytmy uczące uważane są za nieliniowe algorytmy optymalizujące, więc przy porównaniu powinno się brać pod uwagę takie czynniki jak: efektywność(effectiveness), skuteczność(efficacy) oraz odporność(robustness).

Poza tym należy również porównać czas osiągnięcia optimum przy trenowaniu. Branie pod uwagę ilości epok może prowadzić do błędnych wyników gdyż każdy algorytm ma inny przebieg epok, a tym samym czasy jednej epoki mogą się znacznie różnić.

Istniejące opracowania problemu

Istnieje wiele wypracowań poruszających problem doboru metod do liczenia wag w sieciach neuronowych, wykorzystując między innymi algorytmy ewolucyjne. Wiele z nich porównuje te metody z innymi znanymi i dobrze radzącymi sobie metodami np. gradientowymi.

Nie natrafiłem jednak na wypracowania porównujące ze sobą różne metody ewolucyjne, co by mogło oznaczać, że nie są one powszechnie stosowane w tym przypadku, zwłaszcza jeżeli istnieje wiele innych metod równie lub nawet lepiej przystosowanych do znajdywania optimum tego problemu.

Koniec