

Studium Magisterskie

Kierunek Analiza Danych - Big Data

Karol Szerszeń  
 Nr albumu 71273

**Optymalizacja multi-ofertowego marketingu bezpośredniego z wykorzystaniem genetycznych sieci neuronowych**

Praca magisterska

napisana w Instytucie Ekonometrii

pod kierunkiem naukowym

Dr Michała Bernardelli

Warszawa 2017

Spis Treści

1. Wstęp
2. Multi-ofertowy marketing bezpośredni
   1. Marketing bezpośredni
   2. Komunikacja multi-ofertowa
   3. Heurystyka
3. Zbiór danych
   1. Opis zbioru
   2. Dostępne zmienne
   3. Metody doboru zmiennych do modeli
4. Algorytmy genetyczne
   1. Idea algorytmów ewolucyjnych
   2. Podstawowe pojęcia
   3. Operatory
   4. Kodowanie
   5. Zastosowanie
5. Sieci Neuronowe
   1. Idea sieci neuronowych
   2. Topologia
   3. Funkcje aktywacji
   4. Uczenie sieci
   5. Ocena jakości sieci
6. Implementacja
   1. Język programowania, biblioteki, architektura
   2. Złożoność obliczeniowa i wielowątkowość
7. Wyniki
   1. Wydajność obliczeniowa
   2. Jakość modeli

Bibliografia

1. Wstęp

Marketing bezpośredni jest jedną z najstarszych form promocji produktów i usług. Współcześnie zagadnienie

Hipotezy stawiane w pracy:

* Genetycznie sieci neuronowe są dobrym narzędziem do optymalizacji zagadnienia marketingu bezpośredniego
* Porównanie efektywności genetycznej sieci neuronowej w porównaniu do klasycznej sieci neuronowej opartej o algorytm propagacji wstecznej

2. Multi-ofertowy marketing bezpośredni

2.1 Marketing bezpośredni

2.2 Komunikacja multi-ofertowa

<Diagram z mailem>?

2.3 Heurystyka

Wywalić rozdział lub poszerzyć - opisać zbiór danych oraz sam problem optymalizacyjny

Luźne myśli:

* Czy to się w ogóle opłaca? Jaka jest różnica pomiędzy dobraniem ludzi metodą maksimum ze scoringów a jakimś bardziej zaawansowanym modelem?
* Jeśli budujesz populacje to czym są zbiory rozwiązań? Dobór sklepów to jedna rzecz - ale one są dobierane na podstawie ludzi, którzy mają największe prawdopodobieństwo zrobienia zakupów w danych sklepach
* Funkcja celu *f*? Głównym wyznacznikiem skuteczności kampanii marketingowej jest zysk inkrementalny - Incremental
* Możemy mieć dane wielkości koszyków osób, które zrobiły transakcje ale czy to nam nie zmieni zagadnienia w regresję?
* Maksymalizujemy zysk? Czy może liczbę transakcji, niezależnie od zysku/przychodu/prowizji? Na pewno nie akwizycja zupełnie nowych ludzi

3. Zbiór danych

3.1 Opis zbioru danych

Zbiór danych został udostępniony przez firmę Loyalty Partner Polska Sp. z o.o., której głównym produktem jest program lojalnościowy PAYBACK. Dane pochodzą z akcji marketingowej obejmującej wysyłkę e-mail informacji o dostępnych ofertach wyróżnionych sklepów oraz związanych z nimi promocjach.

Zbiór składa się z 226,272 obserwacji, z czego 1772 obserwacje zostały oznaczone jako zdarzenia “pozytywne” czyli zaledwie 0.7%. Definicja zdarzenia pozytywnego to otrzymanie wiadomości e-mail, otworzenie jej oraz dokonanie zakupu w ciągu 14 dni w jednym ze sklepów komunikowanych w tej wysyłce.

3.2 Dostępne zmienne i ich charakterystyki

3.3 Przekształcenia zmiennych

4. Algorytmy genetyczne

4.1 Idea algorytmów ewolucyjnych

Algorytmy ewolucyjne to techniki optymalizacyjne inspirowane analogiami biologicznymi - oparte są na idei doboru naturalnego i ewolucji osobników danej populacji pod wpływem zmieniających się czynników środowiska[[1]](#footnote-1). Presja tegoż środowiska stymuluje proces selekcji naturalnej prowadzący do promowania osobników mających większą szansę na przetrwanie i eliminacji mniej przystosowanych, tym samym zwiększając szanse przetrwania całej populacji. Idea algorytmów ewolucyjnych sięga lat 50 XX wieku, ale matematyczne podstawy zostały opracowane przez Johna H. Hollanda[[2]](#footnote-2) a ich pierwszą wersję opublikował on w 1975 roku. W tej klasycznej już pracy przedstawił on modele pozwalające na opisanie nieliniowych interakcji z jakimi mamy do czynienia w procesie ewolucji i dostosowania do zmieniającego się środowiska. Techniki optymalizacyjne oparte na algorytmach ewolucyjnych używają podejścia opartego nie na pojedynczych rozwiązaniach ale na całych ich populacjach[[3]](#footnote-3) gdzie w pojedynczej iteracji bierze udział zbiór rozwiązań i “ewoluuje” w nowy zbiór rozwiązań w kolejnej iteracji. Gwiazda[[4]](#footnote-4) wyróżnia następujące zalety algorytmów ewolucyjnych:

* brak wymagań co do postaci rozwiązywanego problemu - nie jest wymagane podanie postaci funkcji celu a jedynie jej istnienie lub bardziej ogólnie - istnienie miary, która pozwala na wybranie lepszego z dwóch dostępnych rozwiązań, gdzie termin *lepsze* jest definiowany przez konkretne zagadnienie do którego algorytm jest wykorzystywany)
* zdolność opuszczania lokalnych ekstremów - dzięki rozpatrywaniu populacji rozwiązań oraz mechanizmom samych algorytmów mimo znalezienia lokalnego ekstremum możliwe jest jego opuszczenie i kontynuowanie poszukiwania rozwiązania w innej przestrzeni
* uniwersalność stosowania - brak istnienia dedykowanego rozwiązania danego problemu nie jest przeszkodą do zastosowania algorytmu w jego klasycznej postaci

Dodatkowo algorytmy ewolucyjne operując na populacjach rozwiązań, a nie na pojedynczym, punktowym rozwiązaniu pozwalają na łatwe zastosowanie przetwarzania wielowątkowego co prowadzi do często lepszego czasu obliczeń w porównaniu do klasycznych metod optymalizacyjnych.

Natomiast do głównych wad algorytmów ewolucyjnych możemy zaliczyć[[5]](#footnote-5):

* uniwersalność prowadząca do mniejszej skuteczności niż w przypadku algorytmów dedykowanych
* większa złożoność obliczeniowa - algorytmy ewolucyjne są zwykle wolniejsze od metod zachłannych (obliczenia prowadzone na całych populacjach)
* niedeterministyczność - z założenia algorytm jest losowy, więc odtworzenie rozwiązania bywa niemożliwe
* postać funkcji celu ma duży wpływ na jakość rozwiązania

Algorytmy ewolucyjne są określeniem stosowanym do opisu klasy zagadnień bazujących na zbliżonej metodologii, obejmująca m.in.:

* algorytmy genetyczne
* programowanie genetyczne
* strategie ewolucyjne

W niniejszej pracy zastosowanie znajdzie głównie zagadnienie algorytmów genetycznych oraz strategii ewolucyjnych, głównie ze względu na ich możliwość przeszukiwania przestrzeni decyzyjnej w kilku punktach jednocześnie.

### 4.2 Klasyczny algorytm genetyczny

Algorytmy ewolucyjne charakteryzują się specyficzną nomenklaturą zaczerpniętą z pojęć stosowanych w dziedzinie biologii - genetyce. W klasycznym algorytmie genetycznym*[[6]](#footnote-6)* możemy wyróżnić struktury, zwane chromosomami (*ang. chromosomes)*, będące reprezentantami dziedziny optymalizowanej funkcji a składające się z mniejszych jednostek czyli genów. Zbiór struktur w poszczególnych etapach przetwarzania (w danej iteracji) tworzy populację. Określamy również funkcję przystosowania (*ang. fitness function)* która pozwala na mierzenie jakości dopasowania powyższych struktur podczas tworzenia nowej populacji. Najlepsze z nich “przeżywają” i uczestniczą w procesie reprodukcji przeprowadzanym przez mechanizm selekcji, natomiast pozostałe “wymierają” i nie są uwzględniane w przyszłych populacjach. Geny struktur wyselekcjonowanych do reprodukcji są wybierane przez operator krzyżowania i wymieniane pomiędzy rozwiązaniami rodzicielskimi. Z niewielkim prawdopodobieństwem poszczególne geny mogą zostać poddane mutacji - czyli wymienione na inne. Dzięki powyższym mechanizmom przeszukiwanie przestrzeni rozwiązań w kolejnych iteracjach zostaje zawężone do najbardziej obiecujących - potencjalnych - obszarów przestrzeni decyzyjnej a więc zbliżanie się do rozwiązania optymalnego. Zastosowanie algorytmu genetycznego wymaga również dążenia do zróżnicowania populacji, poprzez operator mutacji, co pozwala na wychodzenie z ekstremum lokalnego. Można to osiągnąć również poprzez maksymalne zróżnicowanie populacji początkowej - maksymalizacji entropii.

*Wykres/diagram schematu działania algorytmu genetycznego*

Powyższy diagram przedstawia działanie klasycznego algorytmu genetycznego - mutacja oznaczona jest jako opcjonalny krok w wyznaczaniu populacji potomków. Inicjalizacja populacji początkowej to tworzenie losowego zbioru na którym wykonywane będą dalsze kroki. Ważne jest, aby charakteryzowała się dużym zróżnicowaniem, gdyż efektywność początkowego algorytmu jest uzależniona właśnie od tego zbioru. Wielokrotne występowanie ciągów genów o wysokiej wartości funkcji przystosowania może prowadzić do całkowitego zdominowania rozwiązania przez właśnie te geny.

### 4.3 Operatory

4.3.1 Selekcja

Operator selekcji pozwala na istnienie procesu określającego, które rozwiązania (chromosomy) przetrwają i będą miały szansę na przekazanie genów populacji potomnej, a które “wymrą”. Głównym celem tego operatora jest stawianie nacisku na wybór dobrze przystosowanych rozwiązań i eliminacja źle przystosowanych podczas utrzymywania wielkości populacji na stałym poziomie. Istnieje wiele rodzajów operatorów selekcji, ale ich wspólną charakterystyką jest próba zbalansowania przystosowania oraz różnorodności osobników w danej populacji[[7]](#footnote-7). Faworyzowanie przystosowania względem różnorodności - nacisku selektywnego (*ang. selective pressure*) - prowadzi do całkowitej dominacja jednego chromosomu nad innymi - zjawiska stłoczenia (*ang. crowding*). Metody eliminujące problem stłoczenia zaproponowali m.in. De Jong[[8]](#footnote-8) sugerując, że nowe osobniki powinny zastępować te najbardziej do nich podobne oraz Goldberg i Richardson[[9]](#footnote-9), którzy przyjęli funkcję podziału przystosowania, zmniejszającą wartość przystosowania danego chromosomu wprost proporcjonalnie do skali podobieństwa do innych chromosomów, tym samym nagradzając różnorodność.

Podstawowym i najczęściej stosowanymi operatorami selekcji są:

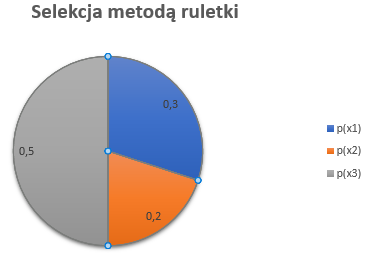
* metoda ruletki
* selekcja rankingowa
* selekcja turniejowa

W poniższej pracy zastosowana zostanie metoda ruletki, która swą nazwę zawdzięcza analogii do losowania za pomocą koła kasynowej ruletki. Ogólny schemat tej metody wygląda następująco:

1. Obliczenie sumy wartości funkcji celu (przystosowania):

f(x) =

1. Obliczenie wkładu każdego osobnika w sumę: p() = f() / f(x)
2. Traktujemy wartości p() jako rozkład prawdopodobieństwa i n-krotnie losujemy osobniki zgodnie z tym rozkładem



Rysunek Selekcja metodą ruletki, Źródło: opracowanie własne

Selekcja chromosomu może być postrzegana jako obrót koła ruletki, dzięki czemu zostaje wybrany chromosom, należący do wybranego fragmentu koła. Im większy jest fragment koła, tym prawdopodobieństwo wyboru jest większe a tym samym wprost proporcjonalne do wartości funkcji przystosowania danego chromosomu - im lepsze dane rozwiązanie tym częściej będzie wybierane.

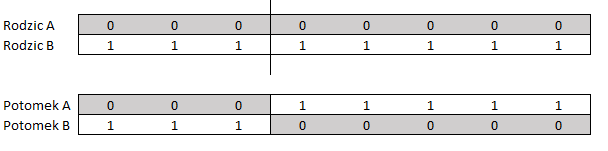
4.3.2 Krzyżowanie

Krzyżowanie to proces wymiany informacji zapisanej w genach poszczególnych genotypów pochodzących od różnych rozwiązań rodzicielskich. Pozwala on na otrzymywanie nowych kombinacji cech - zarówno łącznie lepszych niż oba rozwiązania rodzicielskie jak i z możliwością bycia gorszym od każdego z nich. Krzyżowanie jest dominującym operatorem w algorytmach genetycznych - decyduje o konieczności utrzymania różnorodnej oraz licznej populacji. Istnieje wiele metod krzyżowania jednak w poniższej pracy zastosowany zostanie operator krzyżowania prostego, którego schemat został przedstawiony poniżej.

Krzyżowanie proste[[10]](#footnote-10) polega na wybraniu losowej liczby *n* - punktu krzyżowania. Następnie zastosowana jest następująca reguła:

* pierwszy potomek otrzymuje pierwsze *n* genów od pierwszego rozwiązania rodzicielskiego, natomiast pozostałe od drugiego
* drugi potomek otrzymuje pozostałe geny obu rozwiązań rodzicielskich czyli pierwsze *n* od drugiego oraz pozostałe od pierwszego

Poniższy rysunek obrazuje podziały chromosomów rozwiązań rodzicielskich oraz tworzenie nowych chromosomów przy pomocy operatora krzyżowania, z wartością *n* = 3:



Rysunek Tworzenie nowych chromosomów w wyniku operatora krzyżowania, Żródło: opracowanie własne

4.3.3 Mutacja

Operator mutacji polega na zmianie wartości jednego z genów danego chromosomu na inny, np. dla chromosomu [11001] mutacji może ulec gen trzeci w wyniku czego powstanie chromosom [11101]. Mutacja zachodzi z niewielkim prawdopodobieństwem *p*, ustalanym przy inicjalizacji algorytmu, najczęściej nieprzekraczającym kilku procent. Prawdopodobieństwo mutacji może również być modyfikowane w czasie - podobnie jak operator temperatury w innym algorytmie optymalizacyjnym - wyżarzaniu symulowanym[[11]](#footnote-11). Jedną z metod wyboru genów do mutacji jest wylosowanie liczby w przedziale [0, 1] dla każdego genu i zastosowanie mutacji dla tych, gdzie wylosowana liczba jest mniejsza od prawdopodobieństwa p.

### 4.4 Funkcja przystosowania

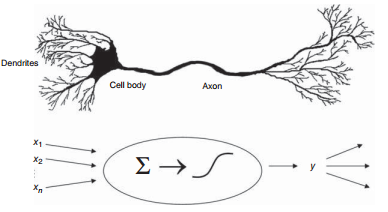
Funkcja przystosowania służy do oceny jakości przystosowania danego chromosomu - określa jak dobrze dany chromosom rozwiązuje dany problem. Pozwala ona na porównywanie poszczególnych rozwiązań pomiędzy sobą - jej wartość jest konieczna w przypadku zastosowania operatora selekcji. Funkcja przystosowania jest wybierana na podstawie problemu do którego jest stosowany algorytm - będzie ona inna przy problemie komiwojażera oraz inna przy optymalizacji wag sieci neuronowej.

Funkcja przystosowania w problemie zastosowania algorytmu genetycznego do optymalizacji wag sieci neuronowej będzie odzwierciedlać poprawność klasyfikacji przez sieć z wagami określonymi daną iteracją algorytmu.

4. Sieci Neuronowe

4.1 Idea Sieci Neuronowych

Sztuczne sieci neuronowe zostały stworzone aby odzwierciedlać strukturę ludzkiego mózgu w celu stworzenia sztucznej inteligencji. Ludzki umysł składa się z około 1011 neuronów, każdy połączony do średnio 10,000 innych neuronów, co daje około 1015 połączeń. Sieci neuronowe są próbą odtworzenia nieliniowego sposobu uczenia obecnych w sieciach neuronowych występujących w przyrodzie. Poniższy rysunek[[12]](#footnote-12) przedstawia prawdziwy neuron, który używa dendrytów do zbierania bodźców od innych neuronów i łączy informacje wejściowe, generując nieliniową odpowiedź i przesyłając ją przez akson do innych neuronów.



Rysunek Schemat ludzkiego neuronu oraz prostej sieci neuronowej, Źródło: Data Mining and Predictive Analytics, Larose, D. T. , s.340

Rysunek przedstawia również postać sztucznego neuronu, który obecny jest w większości sieci neuronowych. Informacje wejściowe (xi) są przechwytywane z nadrzędnych warstw neuronów (lub zbioru danych w przypadku pierwszej warstwy), następnie zbierane przez funkcję kombinacji oraz przekształcane przez nieliniową funkcję aktywacji w celu stworzenia informacji wyjściowej, która jest przekazywana do kolejnej warstwy neuronów. Pierwszym prosty model neuronu został zaproponowany przez McCullocha i Pittsa w 1943 roku [[13]](#footnote-13)- udało im się matematycznie opisać pojedynczy neuron w sposób, który pozwalał na zrozumienie podstawowych procesów uczenia zachodzących w mózgu. Sukces tego podejścia polegał na prostocie rozwiązania, które wprawdzie nie było specjalnie potężne, ale dało narzędzie, które w połączeniu w sieć stały się osobną dziedziną nauki a ich możliwości przerosły oczekiwania twórców.

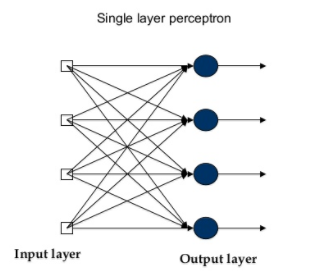
Do zalet sieci neuronowych możemy zaliczyć przede wszystkim ich ogólny charakter – nie ma dużego znaczenia jaki problem próbujemy rozwiązać, sieć będzie się sprawdzać tak samo dobrze. Dzięki tej ogólności można je stosować do rozwiązywania problemów, dla których nie istnieją wyspecjalizowane sposoby radzenia sobie z nimi lub są one zbyt trudne do rozwiązania metodami analitycznymi, np. zagadnienia modelowania cen na rynkach papierów wartościowych czy też zachowania silników indukcyjnych[[14]](#footnote-14). Dodatkowo sieci pozwalają na pominięcie długiego i często bardzo skomplikowanego procesu programowania danego rozwiązania zastępując je procesem uczenia sieci.

Podstawową wadą sieci neuronowych jest niewielka interpretowalność wyników. W przeciwieństwie do klasycznych metod regresji nie jesteśmy w stanie powiedzieć o ile zmieni się odpowiedź klasyfikacji, jeśli zmienimy wartość jednego z atrybutów. Nieco pomaga w interpretacji analiza wrażliwości, jednak daje ona tylko informacje o tym, który z parametrów wejściowych jest najbardziej istotny, natomiast nie pozwala na określenie jak bardzo zmieni się wartość wyjściowa.

Sieci neuronowe, tak jak wcześniej wspomniano, znalazły zastosowania w wielu dziedzinach nauki. Potharst, Kaymak i Pijls[[15]](#footnote-15) zastosowali sieci neuronowe do doboru grupy docelowej dla akcji charytatywnej. Natomiast Yang, Liu i Coid[[16]](#footnote-16) użyli ich jako jednej z technik do klasyfikacji poważnych przestępstw oraz określania prawdopodobieństwa recydywy wśród skazanych. Jednym z najpopularniejszych zastosowań sieci neuronowych jest przewidywanie cen instrumentów finansowych, ze względu na nieprzewidywalność rynków oraz ciężkie do określenia zależności, gdzie jako przykład można podać pracę Dunisa[[17]](#footnote-17), w której zastosował kilka hybrydowych rozwiązań łączących sieci neuronowe z algorytmami genetycznymi, uzyskując dobre rezultaty.

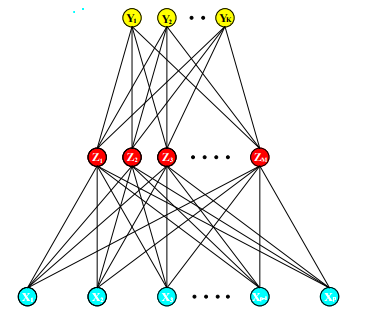
4.2 Topologia

Topologia sieci neuronowych (architektura) to określenie obrazujące strukturę danej sieci czyli sposób połączenia poszczególnych neuronów oraz miejsc, w których wykonywane są obliczenia. Istnieją trzy podstawowe klasy architektury sieci[[18]](#footnote-18).

Najprostszą architekturą jest jednowarstwowa jednokierunkowa sieć (*ang.* single-layer feedforward network) gdzie neurony ułożone są w warstwy. W tej najprostszej formie możemy wyróżnić warstwę wejściową, która przesyła informacje bezpośrednio do warstwy wyjściowej, ale nigdy na odwrót. Nazwa „jednowarstwowa” odnosi się do jednej tylko warstwy wykonującej obliczenia – wyjściowej. 

Rysunek . Jednowarstwowa jednokierunkowa sieć neuronowa, Źródło: https://www.slideshare.net/MohammedBennamoun/artificial-neural-network-lect4-single-layer-perceptron-classifiers

Drugą klasą sieci neuronowych, która odróżnia się od poprzedniej przez obecność warstwy *ukrytej* jest najczęściej spotykana wielowarstwowa jednokierunkowa sieć (ang*. multilayer feedforward network*, lub równoznacznie *Multi Layer Perceptron - MLP*), która zostanie użyta również w tej pracy. W tej architekturze warstwa ukryta, czy też neurony ukryte, odpowiada za większość obliczeń. Określenie *warstwa ukryta* odnosi się do faktu, iż neurony tej warstwy nie są widziane ani od strony wejściowej sieci, ani od warstwy wyjściowej. Główną funkcją jaką spełniają neurony warstwy ukrytej to przechwytywanie informacji z warstwy wejściowej oraz przekształcanie ich w sygnał wyjściowy za pomocą nieliniowych funkcji, zwanych funkcjami aktywacji. Sygnały wejściowe w warstwie wejściowej sieci dostarczają informacji o wektorze wejściowym, który jest przekształcany przez funkcję aktywacji i przekazywany do drugiej warstwy, którą w tym przypadku jest pierwsza warstwa ukryta. Sygnały wyjściowe z tej warstwy przekazywane są do kolejnej, przekształcane przez funkcję aktywacji i tak dalej, dla całej struktury sieci, aż do końcowej warstwy wyjściowej. Zestaw danych wyjściowych z ostatniej warstwy – warstwy wyjściowej – to ostateczny wynik, odpowiedź, sieci na problem zadany przez dane wejściowe. W ogólnym przypadku sieć posiadająca *m* źródeł danych, *h1* neuronów pierwszej warstwy ukrytej, *h2* neuronów drugiej warstwy ukrytej oraz *q* neuronów w warstwie wyjściowej definiujemy jako sieć *m - h1 - h2 - q*. Z reguły takie sieci są w pełni połączone, tzn. każdy neuron jest połączony z każdym innym neuronem warstwy poprzedzającej oraz następnej. Poniższy rysunek prezentuje właśnie taką architekturę, gdzie przez Xp oznaczone zostały neurony warstwy wejściowej, Zm neurony warstwy ukrytej oraz Yk to neurony warstwy wyjściowej.



Rysunek Jednowarstwowa jednokierunkowa sieć neuronowa, Źródło:[[19]](#footnote-19)

Sieci rekurencyjne (ang. *recurrent networks*) różnią się od pozostałych architektur sposobem przepływu informacji – w tej sieci istnieje co najmniej jedna pętla przekazująca informacje między warstwą wyjściową a wejściową. Jeśli sieć posiada jedną warstwę wejściową i jedną wyjściową to sygnał z każdego neuronu warstwy wyjściowej jest przekazywany do pozostałych neuronów warstwy wejściowej. Pętla informacyjna (ang. *feedback loop)* może zawierać również przepływy informacji wyjściowej danego neuronu do niego samego (ang. *self-feedback loop)*. Obecność pętli informacyjnych ma znaczący wpływ na sposób uczenia się sieci oraz na jej wydajność i dokładność. Co więcej sieci rekurencyjne korzystają z gałęzi złożonych z elementów opóźnionych w czasie, co skutkuje nieliniowym zachowaniem, zakładając, że sieć zawiera nieliniowe elementy.

W związku z ogromnym zainteresowaniem sieciami neuronowymi w latach 90-tych XX wieku powstało wiele innych topologii, jednakże ich zastosowanie z reguły ograniczało się do rozwiązywania jednego, konkretnego problemu. Większość sieci neuronowych współcześnie stosowanych opiera się na powyższych trzech schematach, różniąc się nieznacznie od pierwowzoru takimi hiper-parametrami jak rodzaj funkcji aktywacji (sieci o radialnej funkcji aktywacji – *ang.* radial basis function networks[[20]](#footnote-20)), liczbą neuronów w warstwie ukrytej czy też liczbą warstw ukrytych.

4.3 Funkcje aktywacji

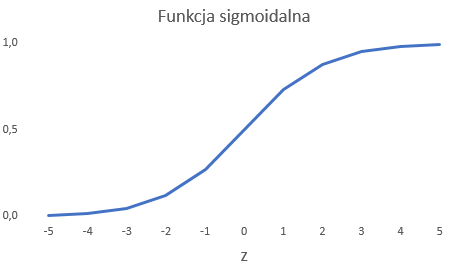
Funkcje aktywacji to funkcje służące do przekształcenia sygnału wejściowego neuronu do postaci nieliniowej. Dla każdej sieci można przypisać inne funkcje aktywacji dla poszczególnych warstw a nawet neuronów, natomiast w większości wypadków stosuje się jedną funkcję aktywacji per sieć. Jako informację wejściową traktuje się sumę sygnałów wejściowych oraz przypisanych im wag[[21]](#footnote-21):

Funkcja taka musi spełniać kilka założeń[[22]](#footnote-22):

* nie może być stała
* musi być ograniczona
* musi być monotonicznie rosnąca
* oraz różniczkowalna

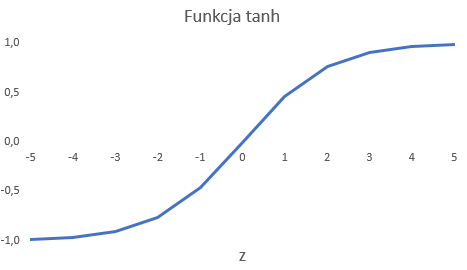
Funkcja aktywacji również powinna odzwierciedlać rodzaj odpowiedzi czy też danych wyjściowych, których model ma dostarczać – jeśli zjawisko modelowane jest binarne takiej też odpowiedzi powinna dostarczyć funkcja. W przypadku klasyfikacji z kilkoma kategoriami również musi ona zwracać każdą z możliwych odpowiedzi. Biorąc pod uwagę powyższe własności współcześnie najczęściej stosowanymi funkcjami aktywacji są[[23]](#footnote-23):

* funkcja sigmoidalna:



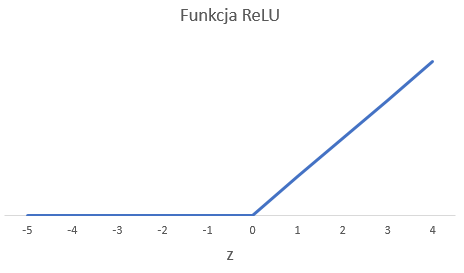
Rysunek Funkcja sigmoidalna

* tangens hiperboliczny:



Rysunek Funkcja tangens hiperboliczny

* ReLU (ang. *restricted linear unit*):



Rysunek Funkcja ReLU

Poza faktem spełniania wyżej wymienionych założeń powyższe funkcje mają też inne zalety. Funkcja sigmoidalna dla *z* blisko zera wykazuje się liniowością, natomiast im *z* bardziej oddala się od zera tym wartości są bliższe odpowiednio 0 lub 1, stając się prawie funkcją stałą. Funkcja tangens hiperboliczny zachowuje się w bardzo podobny sposób, z tym że wartości skrajne to -1 i 1. Funkcja ReLU znajduje zastosowanie m.in. w przetwarzaniach obrazu, głównie ze względu na wydajność obliczeń. Jest ona też standardową funkcją w części zastosowań (np. w pakiecie scikit-learn[[24]](#footnote-24) języka Python).

4.4 Uczenie sieci

W zadaniach klasyfikacji przez model dobrze dopasowany do danych rozumie się model, który minimalizuje funkcję szacującą błąd klasyfikacji, zatem jest to zagadnienie optymalizacyjne. W przypadku sieci neuronowej błąd ten definiuje się jako[[25]](#footnote-25):

,

gdzie :

* – wartość rzeczywista *j* dla rekordu *d*
* – wartość *j* dla rekordu *d,* oszacowana przez neuron z wagami W.

Metodą, która służy do optymalizacji wag w sieci neuronowej do uzyskania minimalnej wartości błędu zdefiniowanego powyżej służy algorytm propagacji wstecznej (ang. *backpropagation algorithm*), przedstawiony po raz pierwszy w 1986 na łamach magazynu *Nature[[26]](#footnote-26).* Podstawą algorytmu jest informacja o tym, w jaki sposób zmienia się błąd E(W) przy zmianie wartości jednej wagi o jednostkę – na tej podstawie możliwe jest minimalizowanie błędu w danych wyjściowych[[27]](#footnote-27). Metoda propagacji wstecznej przeszukuje przestrzeń rozwiązań i ewaluuje zbiór wag w danym kroku na podstawie wartości funkcji E metodą spadku gradientowego (ang. *gradient descent*), który można przedstawić jako:

gdzie:

* - współczynnik szybkości uczenia, ,
* – sygnał wejściowy *i* do neuronu *j*

Jeśli neuron *j* należy do warstwy wejściowej to , w przeciwnym wypadku jest to wartość przekazana z poprzedzającej warstwy. Jeśli neuron *j* należy do warstwy wyjściowej to:

,

gdzie:

* – pochodna cząstkowa funkcji aktywacji *f* względem sumy wag .

Jeśli neuron *j* należy do warstwy ukrytej to:

,

gdzie to gradient wag warstwy wyjściowej.

Ze względu, że do obliczeń jako pierwsza wymagana jest wartość zmiana wag sieci zaczynana jest od ostatniej warstwy, przekazując tę wartość do warstw poprzednich w celu wyliczenia – czyli dokonywana jest propagacja wsteczna tej wartości, stąd nazwa algorytmu. Na podstawie powyższych równań aktualizowane są wagi dla sieci w iteracji *k*:

Kryterium stopu algorytmu jest osiągnięcie ustalonej z góry liczby iteracji lub ustalonego poziomu błędu bliskiemu zeru. Oczywistą wadą algorytmu jest złożoność obliczeniowa związana z koniecznością obliczania pochodnych cząstkowych.

W poniższej pracy podejmuję się próby zastąpienia algorytmu propagacji wstecznej algorytmem genetycznym do wyznaczenia optymalnych wartości wag, ze względu na jego mniejszą złożoność obliczeniową.

5. Implementacja

6. Wnioski

7. Bibliografia

1. “Big Data-Driven Marketing: How Machine Learning Outperforms Marketers’

Gut-Feeling,” http://web.media.mit.edu/~yva/papers/sundsoy2014big.pdf

2. Artificial Intelligence for Marketing. Practical Applications. Jim Sterne, 2017, Wiley

3. Evolutionary Optimization Algorithms. Dan Simon, 2013

4. Adaptation in Natural and artificial systems, John H. Holland, MIT Press, 1992

5. Algorytmy ewolucyjne i ich zastosowania, Ewa Figielska, na: [*http://zeszyty-naukowe.wwsi.edu.pl/zeszyty/zeszyt1/Algorytmy\_Ewolucyjne\_I\_Ich\_Zastosowania.pdf*](http://zeszyty-naukowe.wwsi.edu.pl/zeszyty/zeszyt1/Algorytmy_Ewolucyjne_I_Ich_Zastosowania.pdf)

6. Variants of Evolutionary Algorithms for Real World Applications, Raymond Chiong, Thomas Weise, Zbigniew Michalewicz, Springer, 2012

7. Z. Michalewicz. ***Algorytmy genetyczne + struktury danych = programy ewolucyjne.*** WNT, Warszawa 2003.

8. D.E. Goldberg. ***Algorytmy genetyczne i ich zastosowania.*** WNT, Warszawa 1995.

9. Arabas J., 2004, Wykłady z algorytmów ewolucyjnych, Wydawnictwo Naukowo-Techniczne, Warszawa.

10. Computation Intelligence Techniques for Trading and Investment, CH. Dunis, New York, Routledge, 2014

1. E. Figielska, *Algorytmy ewolucyjne i ich zastosowanie*  [↑](#footnote-ref-1)
2. J. H. Holland, *Adaptation in Natural and artificial systems*, Cambridge 1992, MIT Press [↑](#footnote-ref-2)
3. K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms,*  2001, Wiley [↑](#footnote-ref-3)
4. T. D. Gwiazda, *Algorytmy ewolucyjne w rozwiązywaniu nieliniowych problemów decyzyjnych,* 2002, Wydawnictwa Naukowe Wydziału Zarządzania Uniwersytetu Warszawskiego [↑](#footnote-ref-4)
5. [http://edu.pjwstk.edu.pl/wyklady/nai/scb/wyklad10/w10.htm] [↑](#footnote-ref-5)
6. *Algorytmy genetyczne, ewolucyjne i metaheurystyki,* T. Trzaskalik (red.), Katowice, Wydawnictwo Akademii Ekonomicznej w Katowicach, 2005, s.13-14 [↑](#footnote-ref-6)
7. *Metody i modele eksploracji danych*, D.T. Larose, Hoboken, Wiley 2006 [↑](#footnote-ref-7)
8. *An analysis of the behavior of a class of genetic adaptive systems, praca doktorska,* K. De Jong, University of Michigan, Ann Arbor, 1975 [↑](#footnote-ref-8)
9. *Genetic algorithms with sharing for multi-modal function optimization,* D. Goldberg, J. Richardson w: *Genetic Algorithms and Their Applications: Proceedings of the 2nd International Conference on Genetic Algorithms,* J. Greffenstette (red.), Lawrence Erlbaum Associates, Hillsdale, 1987 [↑](#footnote-ref-9)
10. *Algorytmy genetyczne, ewolucyjne i metaheurystyki,* T. Trzaskalik (red.), Katowice, Wydawnictwo Akademii Ekonomicznej w Katowicach, 2005, s.18 [↑](#footnote-ref-10)
11. *Applied Simulated Annealing,* R. V. V. Vidal, Berlin, Springer,1993 [↑](#footnote-ref-11)
12. *Data Mining and Predictive Analytics,* D. T.Larose, C. Larose, Hoboken, Wiley, 2015 [↑](#footnote-ref-12)
13. *Modelowanie matematyczne i symulacje komputerowe w naukach społecznych,* K. Winkowska-Nowak (red.), Warszawa, Wydawnictwo SWPS Academica, 2007 [↑](#footnote-ref-13)
14. *O celowości zastosowania sieci neuronowych w problemach związanych z elektrotechniką,* R. Tadeusiewicz, w: *Przegląd Elektrotechniczny,* R. 85NR 2/2009 [↑](#footnote-ref-14)
15. *Neural Networks for Target Selection in Direct Marketing,* R. Potharst, U. Kaymak, W.H.L.M. Pijls, w: *ERIM Report Series Research in Management*, na: <http://hdl.handle.net/1765/83> [↑](#footnote-ref-15)
16. *Applying Neural Networks and other statistical models to the classification of serious offenders and the prediction of recidivism,* M. Yang, Y. Liu, J. Coid, w: *Ministry of Justice Research Series,* nr 6/10, 2010 [↑](#footnote-ref-16)
17. *Computation Intelligence Techniques for Trading and Investment*, CH. Dunis, New York, Routledge, 2014 [↑](#footnote-ref-17)
18. *Neural Networks and Learning Machines,* S. Haykin, Pearson Prentice Hall, Upper Saddle River, 2009 [↑](#footnote-ref-18)
19. *The Elements of Statistical Learning, Data Mining, Inference, and Prediction*, T. Hastie, R. Tibshirani, J. Friedman, Springer, New York, 2009, s.393 [↑](#footnote-ref-19)
20. *Introduction to Radial Basis Function Networks,* M. J. L. Orr, na: https://www.cc.gatech.edu/~isbell/tutorials/rbf-intro.pdf [↑](#footnote-ref-20)
21. *Data Mining – Theories, Algorithms and Examples,* N. Ye, New York, CRC Press, 2014, s. 63 [↑](#footnote-ref-21)
22. *Neural Networks and Learning Machines,* S. Haykin, Pearson Prentice Hall, Upper Saddle River, 2009 [↑](#footnote-ref-22)
23. *Fundamentals of Deep Learning,* N. Buduma, Sebastopol, O’Reilly Media, 2017 [↑](#footnote-ref-23)
24. http://scikit-learn.org/stable/modules/generated/sklearn.neural\_network.MLPClassifier.html [↑](#footnote-ref-24)
25. *Data Mining – Theories, Algorithms and Examples,* N. Ye, New York, CRC Press, 2014, s. 80 [↑](#footnote-ref-25)
26. *Learning Representations by BackPropagating Errors,* D .E. Rumelhart, G. E. Hinton, R. J. Williams, w: *Nature* nr 323, 1986 [↑](#footnote-ref-26)
27. *Inteligentna Sieć, Algorytmy przyszłości,* D. McIlwrath, H. Marmanis, D. Babenko, Gliwice, Helion, 2017 [↑](#footnote-ref-27)