Politechnika Poznańska

Kierunek Elektronika i Telekomunikacja

Praca dyplomowa inżynierska

Piotr Jastrzębski

Sterowanie polami komutacyjnymi z wykorzystaniem sztucznej inteligencji

Promotor dr hab. inż. Mariusz Żal

Poznań, 2020

Streszczenie

Przedmiotem pracy jest opracowanie modelu uczenia maszynowego wykorzystującego maszynę wektorów wspierających w bibliotece ML.NET, który będzie wykorzystywany do sterowania pracą pól komutacyjnych. Opisano kwestie teoretyczne dotyczące SVM oraz sprawdzono działanie zaproponowanego modelu.

Model odpowiedzialny jest za wybór drogi połączeniowej w polach komutacyjnych i został stworzony dla pól blokowalnych i nieblokowalnych. Dane uczące pochodzą z dwóch algorytmów wybory drogi połączeniowej kolejnościowego oraz minimal index.

Abstract

The subject of this study is development machine learning's model using Support Vector Machines in ML.NET framework which will be used to control the operations of switching fabrics. Theoretical issues concerning SVM were described and operation of the model checked.

Model is responsible for routing control strategies in interconnections network and was create for nonblocking and blocking network. Training data orginate from two routing control strategies sequental and minimal index.

Spis treści

Wstęp

Wprowadzenie

Obecnie przedsiębiorstwa, instytucje czy też zwykli obywatele, użytkownicy sprzętów elektronicznych wysyłają i odbierają ogromne ilości informacji. Aby ten proces mógł sprawnie przebiegać korzystają świadomie lub nie z sieci telekomunikacyjnych i urządzeń sieciowych. Pola komutacyjne dają możliwość przesyłania informacji w żądanym kierunku wykorzystując różnego rodzaju algorytmy i są wykorzystywane w owych urządzeniach, przez co są bardzo ważnym elementem w życiu każdego człowieka, dzięki nim otrzymujemy wiadomości, które są przekazywane konkretnie do nas i mamy możliwość wysłania ich do wybranego adresata z niemal całkowitą pewnością, że odbiorca je otrzyma. Można zaobserwować znaczący wzrost wykorzystania sztucznej inteligencji w naszym życiu, w różnych dziedzinach od marketingu po diagnostykę lekarską. Jedną z dostępnych definicji sztucznej inteligencji jest „*Dziedzina nauki próbująca wyjaśnić i emulować inteligentne zachowania za pomocą metod obliczeniowych*" [1]. Niewiele jednak ona tłumaczy, można przyjąć, że jest to dział informatyki zajmujący się tworzeniem algorytmów zdolnych do przystosowywania się do zmiennych warunków, podejmowania decyzji, czy abstrakcyjnego myślenia [2]. Główną składową sztucznej inteligencji jest uczenie maszynowe, czyli przetwarzanie dużych ilości informacji i wykorzystanie ich do wyznaczenia algorytmów, modeli zachowań czy podejmowania decyzji, charakterystyczne dla uczenia maszynowego jest poprawianie wydajności aplikacji przez przetwarzanie danych. Wykorzystanie takiego podejścia do sterowania pracą pola komutacyjnego będzie przedmiot niniejszej pracy.

Cel pracy

Celem pracy jest opracowanie algorytmu wybory drogi połączeniowej wykorzystującego uczenie maszynowe w zadanych polach komutacyjnych, poruszenie kwestii teoretycznych, które dotyczą tematu pracy oraz porównanie wydajności modelu z algorytmami sterownia polami komutacyjnymi.

Zakres pracy

Na zakres niniejszej pracy składa się zebranie podstawowej wiedzy dotyczącej pól komutacyjnych oraz algorytmów uczenia maszynowego, przygotowanie danych uczących, które będą podstawą w stworzeniu modelu wyboru drogi połączeniowej wykorzystującego maszynę wektorów wspierających. Stworzenie modelu jest wiodącym elementem pracy, który

pozwoli porównać wyniki z niego otrzymane z wynikami używanych w praktyce algorytmów wybory drogi połączeniowej.

Przegląd rozdziałów

Pierwsza część pracy zawiera podstawowe informacje dotyczące pól komutacyjnych, przedstawia ich budowę oraz zasadę działania. Opisano również algorytmy wyboru drogi połączeniowej.

Drugi rozdział skupia się na temacie uczenia maszynowego, zawiera przegląd algorytmów uczenia wraz z przykładami.

Rozdział trzeci opisuje podstawy teoretyczne dotyczące algorytmu maszyny wektorów wspierających, przedstawia wygląd danych uczących i zawiera wyniki przeprowadzonego uczenia.

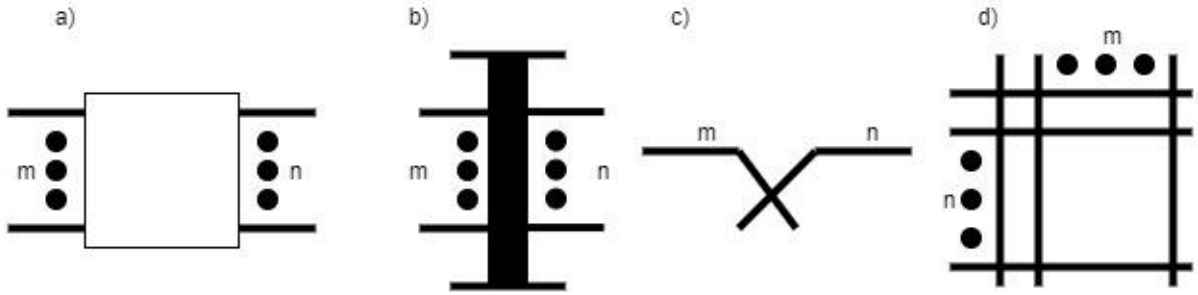
Rozdział czwarty zawiera wyniki przeprowadzonych eksperymentów polegających na porównaniu wydajności modeli z algorytmami wyboru drogi połączeniowej.

ROZDIZAŁ I

Pola komutacyjne

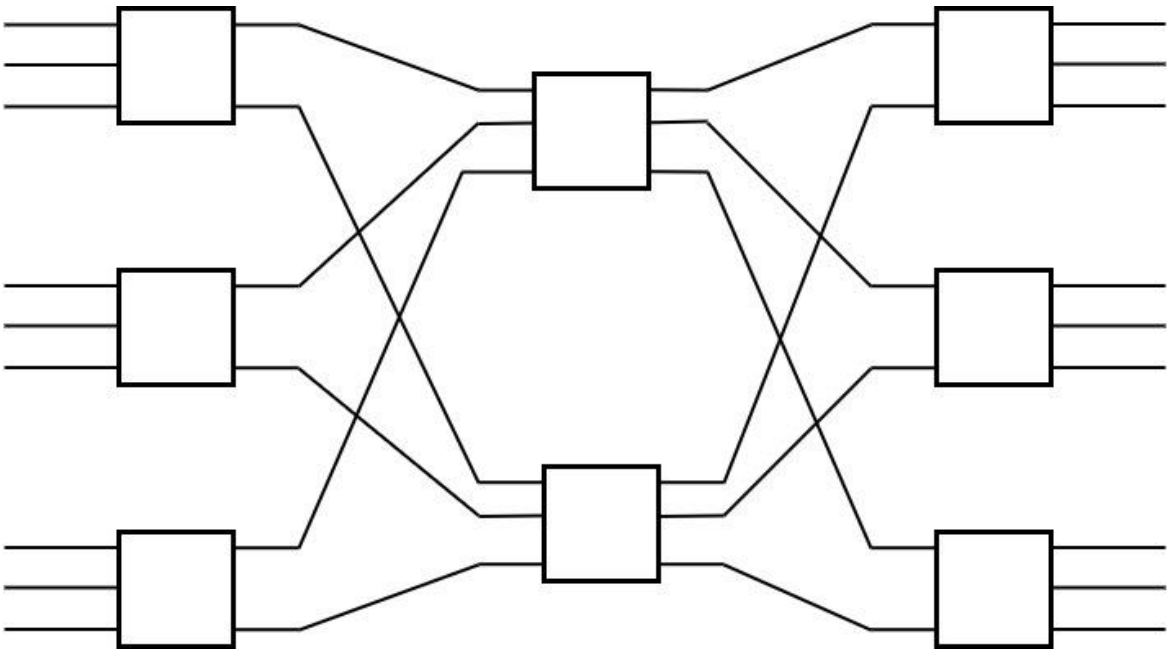
Klasyfikacja pól komutacyjnych

Podstawowym elementem, z którego składa się każde pole komutacyjne jest komutator, który może mieć *m* wejść i *n* wyjść. Przypadek, w którym komutator ma tyle samo wejść, co wyjść, czyli o rozmiarze *m* × *m* nazywamy komutatorem symetrycznym. Komutatory można oznaczać w różny sposób (rys. 1).



Rys.1. Symbole komutatorów o rozmiarze *m* x *n*

Łącząc je ze sobą w odpowiedni sposób możemy zbudować pole komutacyjne (rys. 2).



Rys 2. Pole komutacyjne zbudowane z połączenia komutatorów

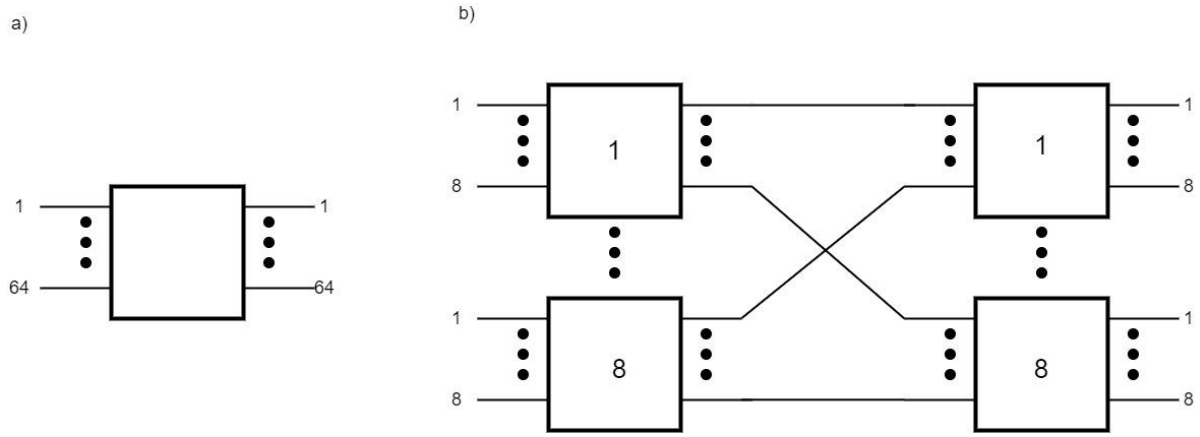
Pola te można podzielić na wiele sposobów, w zależności, jakie cechy i właściwości będą brane pod uwagę. Najogólniejszym spotykanym podziałem jest podział pól ze względu na sposób rozdziału dróg rozmównych. Uwzględniamy w nim pola komutacyjne:

- przestrzenne,
- czasowe,
- częstotliwościowe.

W polach przestrzennych każde połączenie realizowane jest na fizycznie oddzielonej od siebie drodze połączeniowej, w polach czasowych kanały są komutowane przez przesunięcia czasowe, natomiast w polach częstotliwościowych każdemu kanałowi przydzielona jest inna częstotliwość nośna. Wcześniej wspomniane łączenie komutatorów obliguje nas do podziału pól komutacyjnych ze względu na liczbę sekcji, mamy wiec do wyboru pola:

- jednosekcyjne,
- wielosekcyjne.

Jeśli żądane polaczenie jest realizowane przy pomocy jednego punktu komutacyjnego, mamy do czynienia z polem jednosekcyjnym natomiast, jeśli musimy skorzystać z połączenia wyjścia jednego komutatora z wejściem drugiego, korzystamy z pola wielosekcyjnego (rys.3).



Rys. 3. Pole komutacyjne o wymiarze 64x64: a) jednosekcyjne; b) dwusekcyjne

Kolejnym podziałem, który należy uwzględnić łącząc ze sobą komutatory to taki ze względu na stosunek liczby wejść i wyjść, wtedy mamy:

- pola z kompresją,
- pola z ekspansją,
- pola z rozdziałem ruchu.

Pola z kompresją to takie, w których liczba wejść jest większa od liczby wyjść, gdy liczba wejść jest mniejsza od liczby wyjść mówimy o polach z ekspansją, natomiast, gdy liczba wejść i wyjść jest identyczna pola nazywane są polami z rozdziałem ruchu. Najważniejszym podziałem dla tej pracy jest podział ze względu na wystąpienie stanów blokady. „Stan blokady pola komutacyjnego, nazywamy stan, w którym nie można zestawić połączenia między dowolną parą wolne wejście-wolne wyjście a jednocześnie dla rozpatrywanego pola istnieją stany, w których zestawienie takiego połączenia jest możliwe" [3]. Przy takiej definicji wyróżniamy pola:

- nieblokowane w wąskim sensie,
- nieblokowane w szerokim sensie,
- pola przestrajalne,
- pola przepakowalne,
- pola blokowane.

Pole komutacyjne możemy nazwać nieblokowanym w wąskim sensie wtedy, gdy nie występują w nim blokady bez względu na zastosowany algorytm wyboru drogi połączeniowej. Pole nieblokowane w szerokim sennie to takie, w którym możliwe jest ominięcie blokady przez zastosowanie odpowiedniego algorytmu wyboru drogi połączeniowej. W przypadku pól przestrajalnych przy wystąpieniu blokady możliwa jest zmiana zestawionych już dróg połączeniowych, natomiast w przypadku pól przepakowalnych drogi połączeniowe mogą być zmieniane po zakończeniu innych połączeń. Pola, w których nie idzie uniknąć stanów blokady korzystając z wyżej wymienionych metod nazywamy polami blokowanymi.

Algorytmy wyboru drogi połączeniowej

W literaturze przedmiotu można spotkać wiele algorytmów wyboru drogi połączeniowej, najbardziej znane z nich to algorytmy:

- kolejnościowy,
- quasi-przypadkowy,
- Beneša.

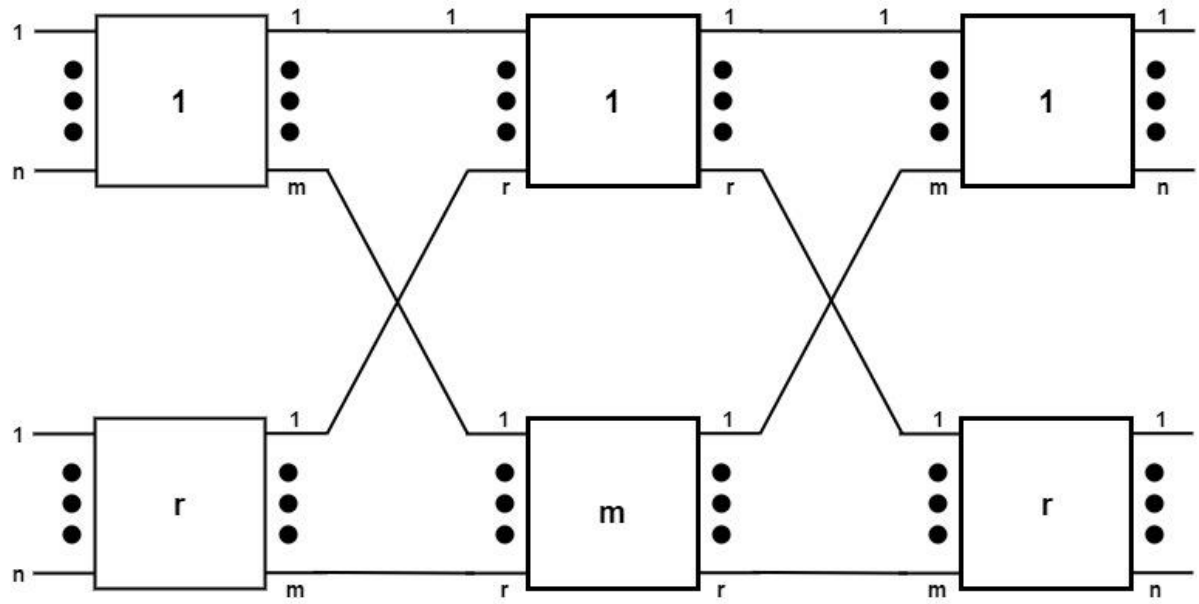
Algorytm kolejnościowy polega na wyborze dla każdego zgłoszenia drogi połączeniowej przez pierwszy wolny i dostępny komutator, kolejność wyboru jest z góry zdefiniowana i stała. Algorytm ten charakteryzuje się większym obciążeniem komutatorów początkowych, często będą one zajęte i algorytm będzie musiał przeglądać komutatory sekcji środkowej.

Algorytm quasi-przypadkowy nie ma ustalonego punktu początkowego, jest on zmienny i jeśli do zestawienia połączenia wykorzystuje się komutator o indeksie *x*, to następne zgłoszenie będzie realizowane przez komutator *x+1* lub jeśli ten będzie zajęty *x+2* lub według schematu *x + kolejny komutator*. W porównaniu do poprzedniego algorytmu ten obciąża komutatory równomiernie, przez co połączenie może być zestawione już w jednym z pierwszych podejść.

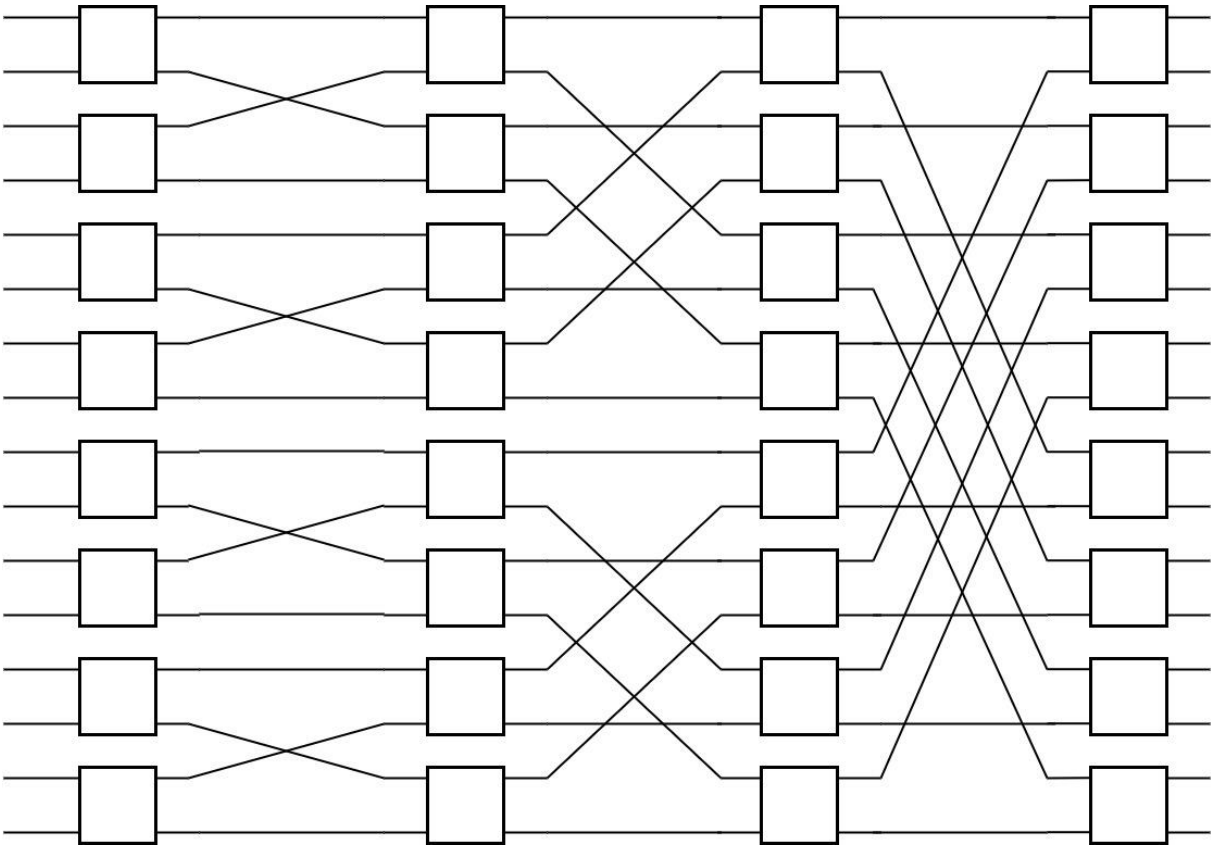
Algorytm Beneša wybiera dla każdego zgłoszenia drogę przez najbardziej obciążony i jednocześnie wolny komutator. Definicją obciążenia w tym przypadku jest liczba realizowanych połączeń w chwili sprawdzania. Istnieje również odmiana tego algorytmu, która unika zestawiania drogi połączeniowej przez całkowicie wolny komutator, o ile jest to możliwe.

Struktury pól komutacyjnych

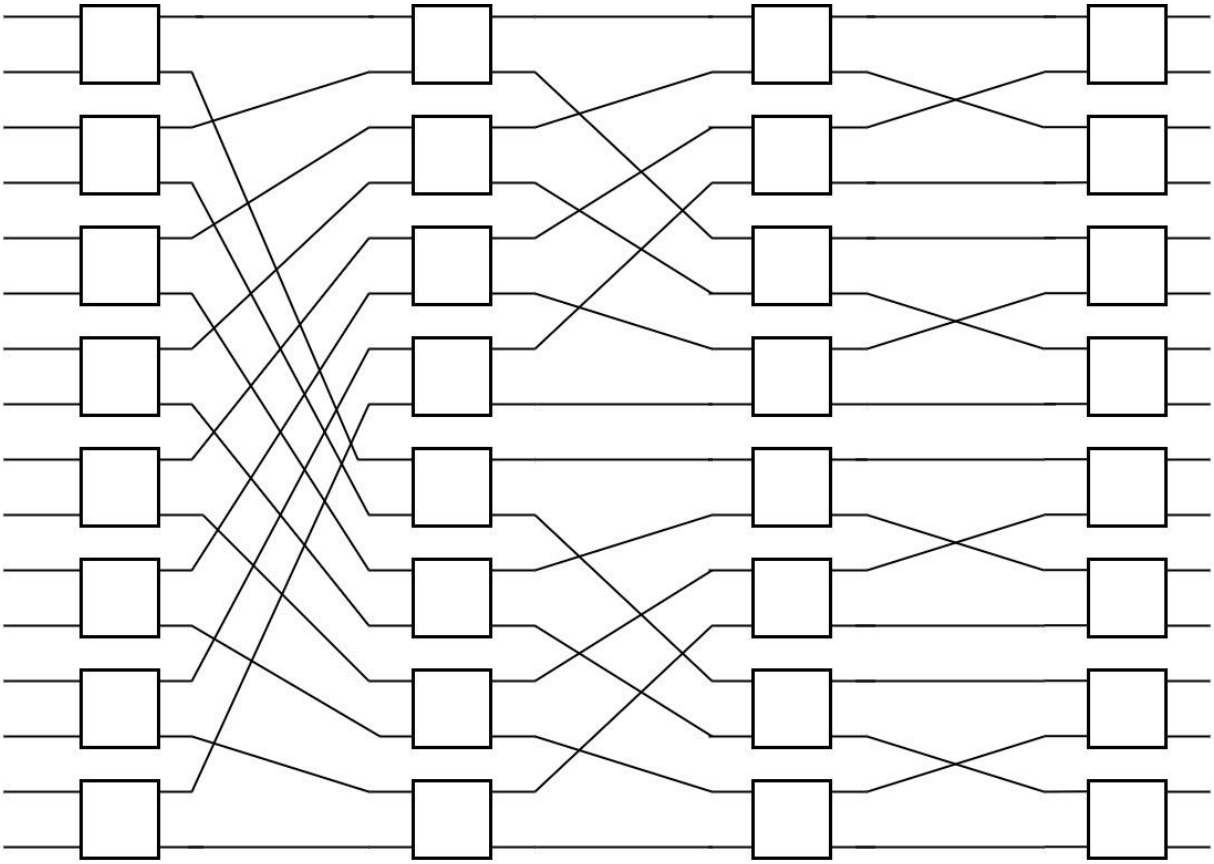
Wyróżniamy wiele struktur pól komutacyjnych, różniących się między sobą własnościami kombinatorycznymi, od pól Closa (rys. 4) przez pola typu banyan do których zaliczamy pola o topologii banyan (rys. 5), baseline (rys. 6) czy omega (rys.7). Trzy ostatnie pola składają się z komutatorów o wymiarze 2x2 i są dla siebie topologicznie równoważne, czyli jeśli jakaś własność jest spełniona dla jednego z nich będzie spełniona również dla pozostałych [4]. Można zauważyć, że składają się z takiej samej liczby elementów, różnią się jedynie sposobem ich połączenia.



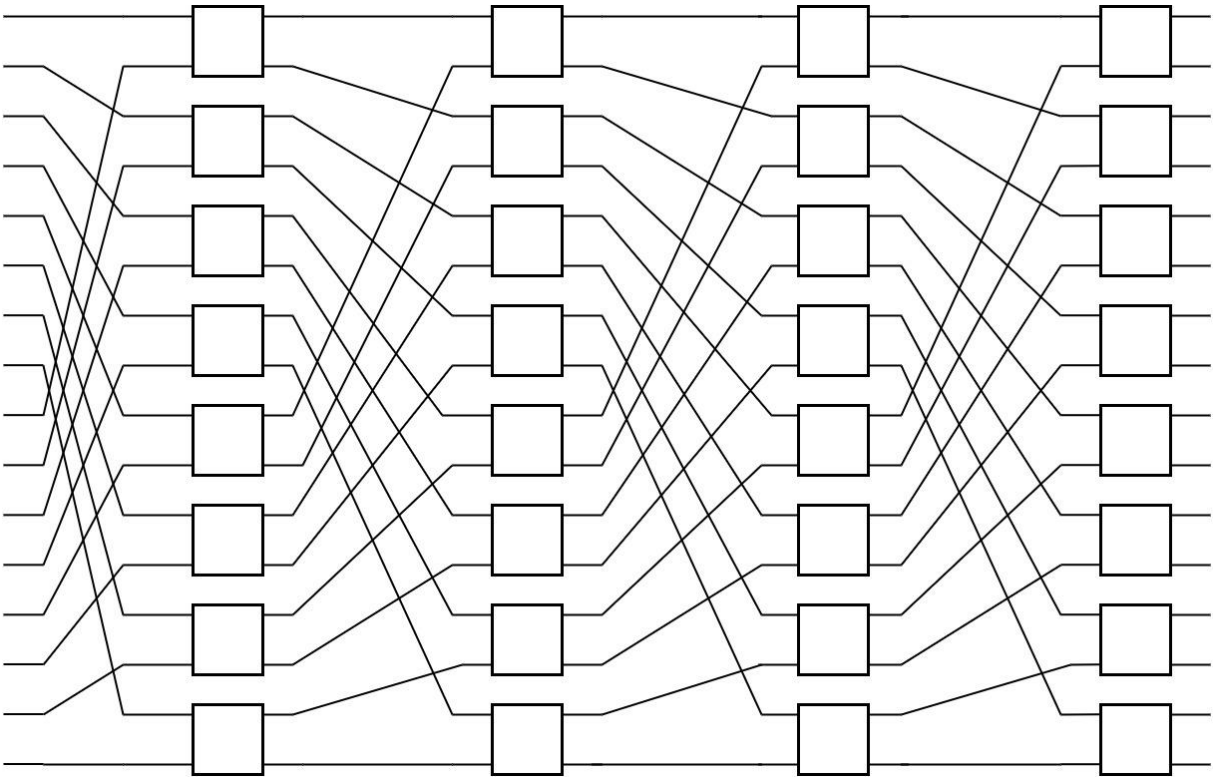
Rys. 4. Trzysekcyjne pole Closa



Rys. 5. Pole komutacyjne typu banyan



Rys. 6. Pole komutacyjne typu baseline



Rys. 7. Pole komutacyjne typu omega

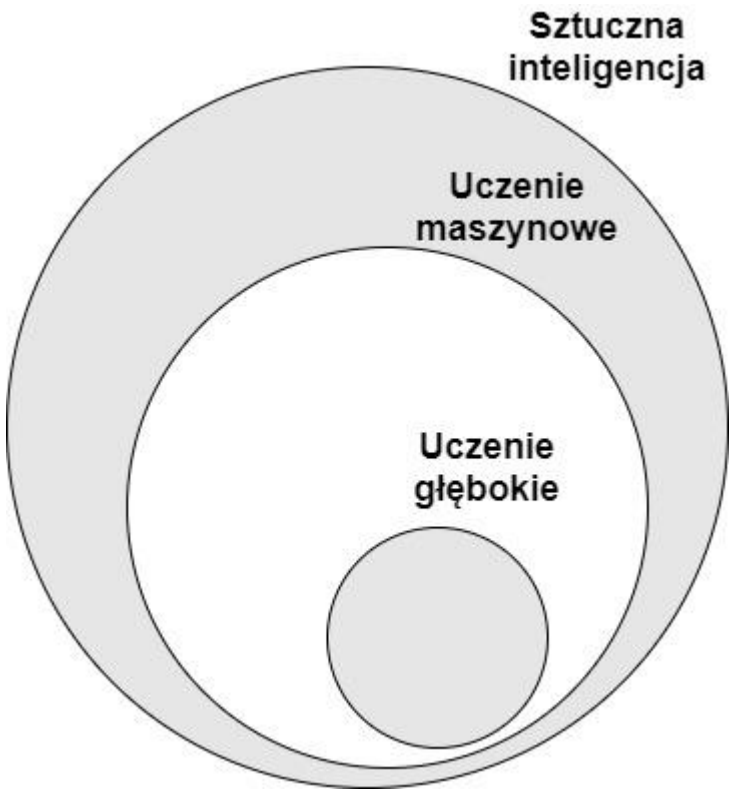
ROZDZIAŁ II

Uczenie maszynowe

Rodzaje uczenia maszynowego

W dzisiejszych czasach sztuczna inteligencja jest wszechobecna, przez personalizowane reklamy, tryby gier aż po autonomiczne pojazdy. Dużą częścią sztucznej inteligencji jest uczenie maszynowe, które będzie głównym tematem pracy. Nie są to jednak pojęcia tożsame, gdyż uczenie maszynowe jest tylko gałęzią sztucznej inteligencji skupiająca się na badaniu algorytmów komputerowych i dostarczaniu im dostępu do danych, aby same się uczyły. Jedną z podkategorii to uczenie głębokie a sztuczna inteligencja to dziedzina skupiająca się na tworzeniu modeli zachowań inteligentnych (rys. 8). Uczenie maszynowe można podzielić na trzy podstawowe kategorię:

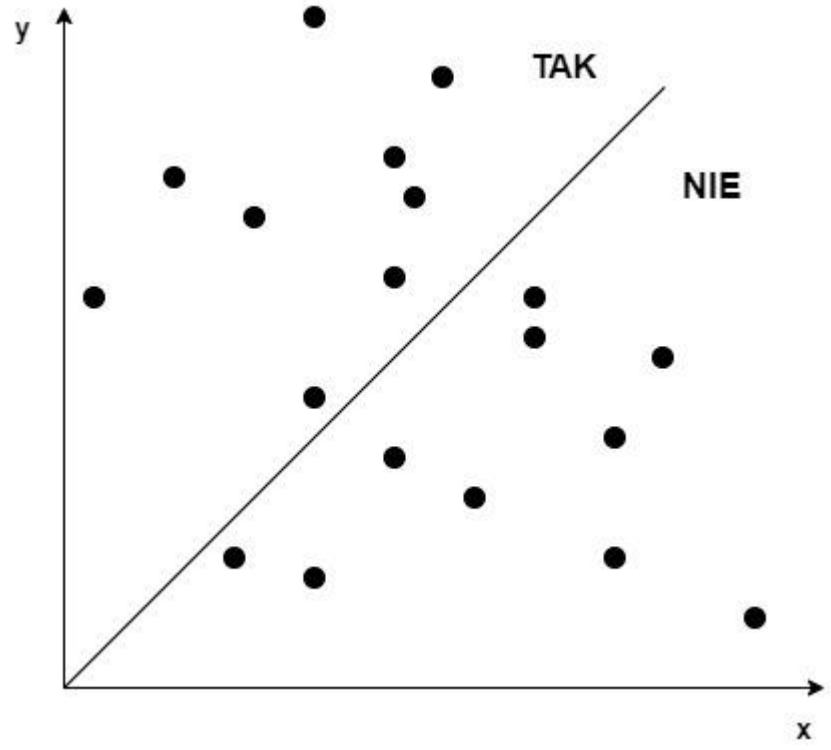
- Uczenie nadzorowane,
- Uczenie nienadzorowane,
- Uczenie ze wzmocnieniem.



Rys. 8. Sztuczna inteligencja a uczenie maszynowe

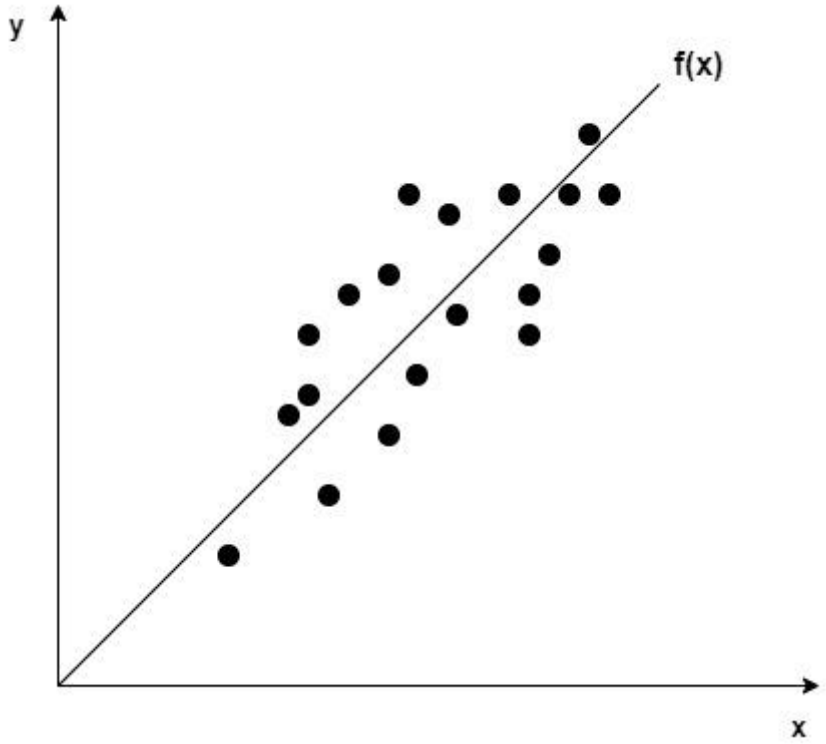
Uczenie nadzorowane

Ten rodzaj uczenia maszynowego jest wykorzystywany do przewidywania wyników na podstawie dostarczonych danych. Jest to swego rodzaju odwzorowanie zbioru wyjściowego na podstawie danych wejściowych, więc aby móc korzystać z tego rodzaju uczenia maszynowego musimy mieć zestaw danych wejściowych i przypisane im dane wyjściowe- oczekiwaną wartość. Ten rodzaj algorytmów korzysta z techniki odchylenia wywiedzionego [5], co oznacza, że algorytm znajduje pewne zależności między danymi i korzysta z nich do przewidywania wartości. Przykładami tego rodzaju uczenia mogą być takie modele jak regresja liniowa, logistyczna czy metody korzystające z struktur drzewiastych. Jednym z praktycznych przykładów wykorzystania takiego modelu uczenia maszynowego jest filtr antyspamowy, zbiór maili zostaje oznaczony w sposób tak/nie, na podstawie tytułów, słów kluczowych, czy budowie wiadomości model wyznacza zależność, według której wiadomości są klasyfikowane, jako spam (rys. 9). W porównaniu do filtrów antyspamowych działających na zasadzie sztywno przypisanych słów kluczowych, zastosowanie uczenia maszynowego znacznie poprawia, jakość filtrowania, z tego względu ze model cały czas jest doskonalony, jeśli nawet wiadomość, na którą oczekiwaliśmy trafia do skrzynki ze spamem, po oznaczeniu jej, jako nie spam model uwzględnia to w tworzonych zależnościach.



Rys 9. Regresja logistyczna

Dobrym, choć nietechnicznym przykładem wykorzystania uczenia nadzorowanego jest wyznaczanie ciężaru maksymalnego stosowanego przez dwuboistów i trójboistów. Dane treningowe składające się z ciężaru, który zawodnicy są w stanie podnieść na pięć razy oraz na raz są przetwarzane i generowana jest funkcja (rys. 10) prognozująca na podstawie zmiennej wejściowej, jaką jest ciężar, który zawodnik jest w stanie zrobić pięć ruchów ciężar maksymalny danego zawodnika.

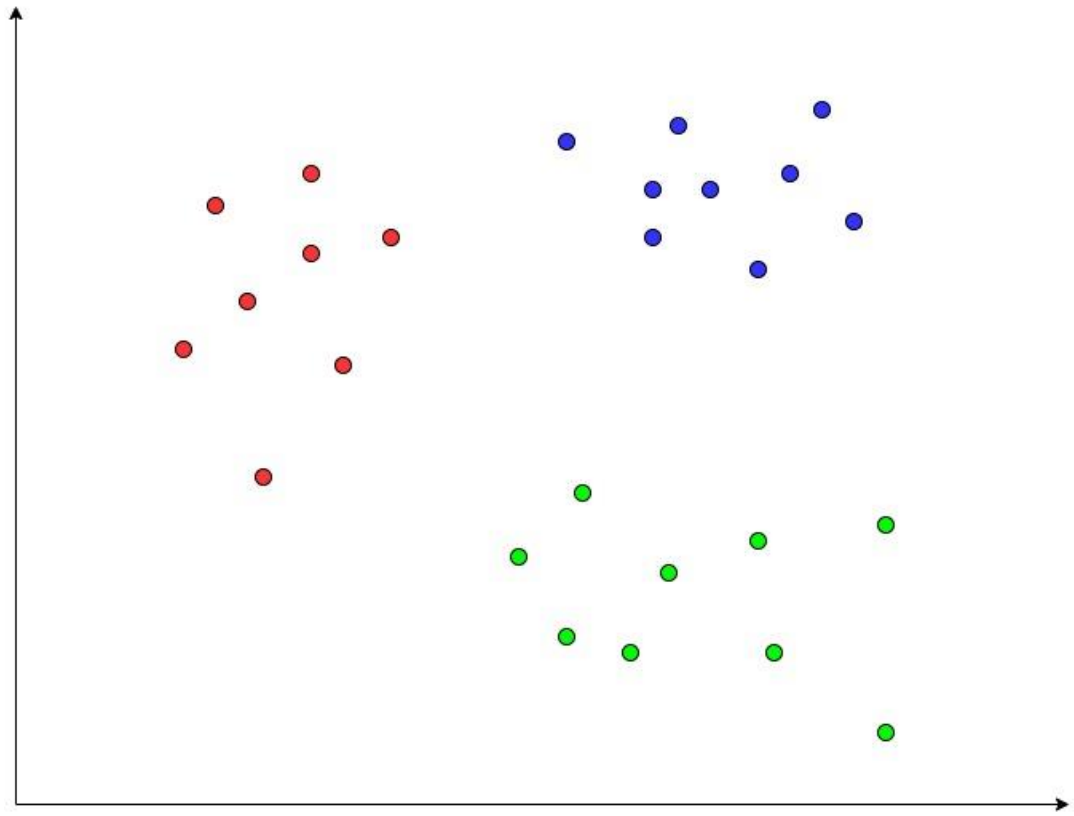


Rys. 10. Regresja liniowa

Uczenie nienadzorowane

Uczenie bez nadzoru w przeciwieństwie do uczenia nadzorowanego nie posiada sklasyfikowanych i etykietowanych danych wyjściowych, korzysta za to tylko ze zbioru wejściowego. Tego rodzaju uczenie maszynowe sprawdza się w szeroko pojętym grupowaniu polegającemu na podzieleniu danego zbioru na kategorie czy grupy, gdzie każda kategoria czy grupa powinna posiadać element bardzo podobny do innych danych z tej samej kategorii a odróżniający go od danych z innej kategorii. Przykładem tego rodzaju algorytmów może być analiza skupień (rys. 11), dzięki której algorytm wyszukuje podobieństwa w danym

zbiorze i dzieli je na kategorię. Popularny serwis umożliwiający oglądanie filmów i seriali po obejrzeniu kilku materiałów zaczyna proponować filmy [6], które mogą nam przypaść do gustu. Gatunek filmu, główny bohater, scenarzysta, reżyser te wszystkie dane wpływają na propozycje, algorytm uczenia maszynowego znajduje pewien schemat, podobieństwo i stwierdza, że skoro oglądaliśmy dramaty z udziałem pewnego aktora, to z chęcią obejrzymy jakąś inną produkcje tego samego gatunku z tym aktorem, jeśli takiej nie ma to bez tego aktora, ale tego reżysera.



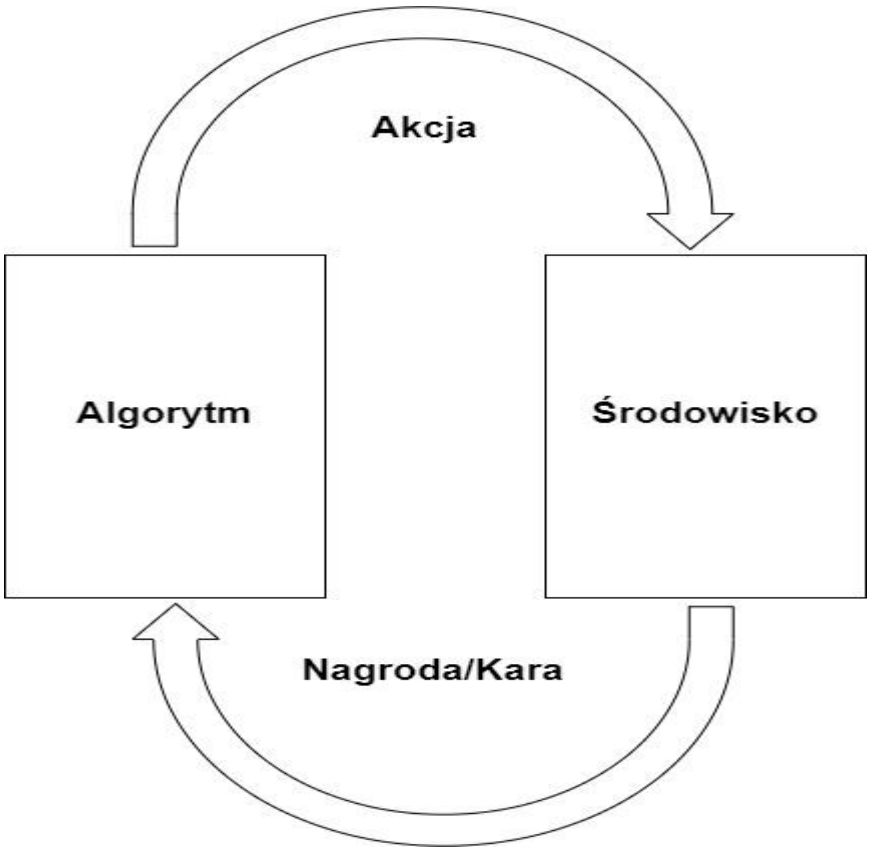
Rys 11. Analiza skupień

Kolejnym obszarem gdzie można zastosować uczenie nienadzorowane jest eksploracja reguł asocjacyjnych, czyli algorytm podpowiada nam, jakie jeszcze przedmioty możemy kupić dokonując zakupów online na podstawie znajdujących się produktów w naszym koszyku. Trzecim sposobem wykorzystania uczenia bez nadzoru, może być wykrywanie anomalii, algorytm analizuję nowe dane i sprawdza czy pasują one do wcześniej wyznaczonych wzorców.

Uczenie ze wzmocnieniem

W tym przypadku uczenie maszyny skierowane jest na konkretny rezultat, celem takich algorytmów jest maksymalizacja wydajności i/lub skuteczności. Uczenie to korzysta z

systemu kar i nagród, jeśli zostanie podjęta odpowiednia decyzja, czyli taka, która da poprawny rezultat lub zbliży nas do poprawnego rezultatu algorytm zostaje nagradzany a jeśli krok oddala nas od oczekiwanych rezultatów dostaję karę. Proces może powtarzać się w nieskończoność (rys. 12), przez co poruszamy się między tym, co algorytm już umie a tym, jakie nowe sytuacje/scenariusz zostają podane na wejście. Przykładem zastosowania tego typu uczenia, mogą być samochody autonomiczne, które cały czas musza współpracować z otoczeniem, szukają przeszkód, ograniczeń prędkości i podejmują decyzję. Ten sposób uczenia bardzo dobrze sprawdza się również przy tworzeniu botów w grach, które mają imitować zachowania graczy.

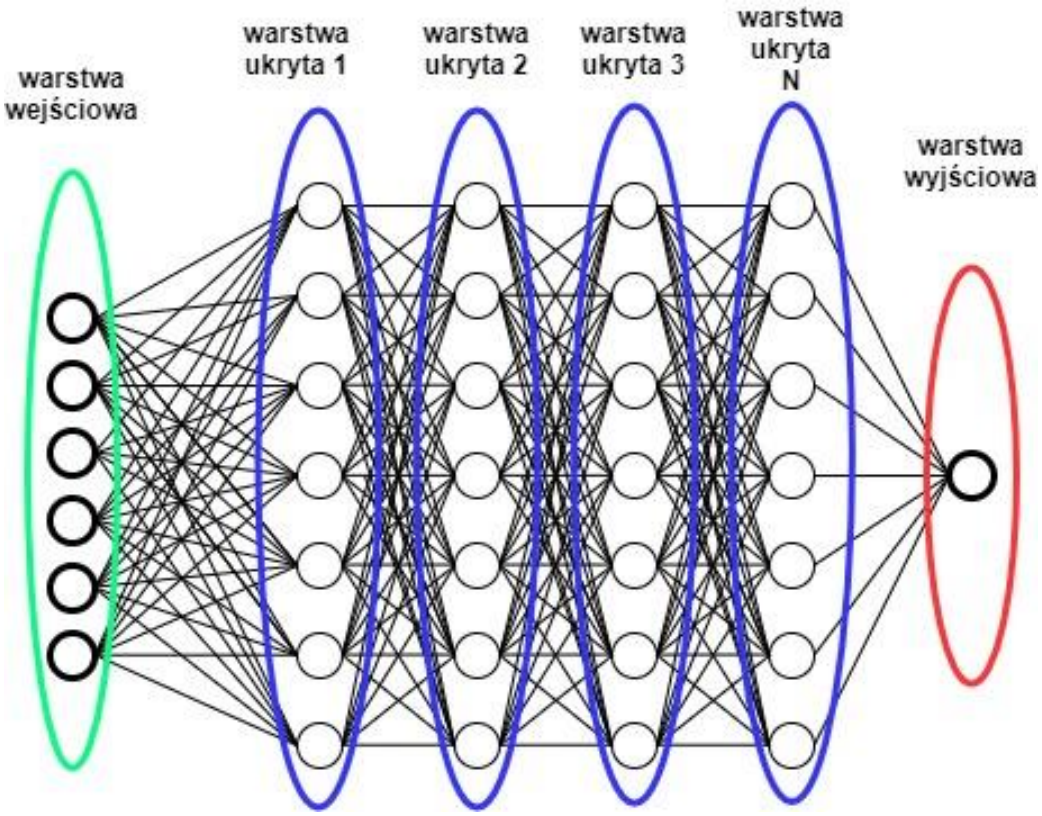


Rys. 12. Schemat działania uczenia ze wzmocnieniem

Uczenie głębokie

Uczenie głębokie jest rodzajem uczenia maszynowego, które do swoich działań wykorzystuje wiele poziomów warstw sieci neuronowych. Dane są przetwarzane nieliniowo, a każda warstwa korzysta z wyników poprzedniej warstwy i przekazuje je dalej, teoretycznie każda warstwa powinna trochę polepszać nasz model, aż w końcu otrzymamy odpowiedni

wynik. Sieci neuronowe, jak zostało wspomniane, korzystają z kilku warstw, wejściowej, ukrytej i wyjściowej, z tym, że ukryta może składać się z nieskończonej liczby warstw, ale rozpatrujemy ją finalnie, jako jedną. Pierwsza warstwa odpowiedzialna jest za zebranie danych i przesłanie ich do warstwy ukrytej, w której zachodzi uczenie się i szukanie zależności między neuronami (rys. 13), a w ostatniej warstwie poznajemy wynik, wnioski utworzone przez sieć neuronową. Sieci neuronowe inspirowane są budową i zasadą działania ludzkiego mózgu i wykorzystywane są do przetwarzania dużych zbiorów danych, działań na obrazach, takich jak rozpoznawanie obrazów, twarzy czy operacjach związany z głosem/muzyką.



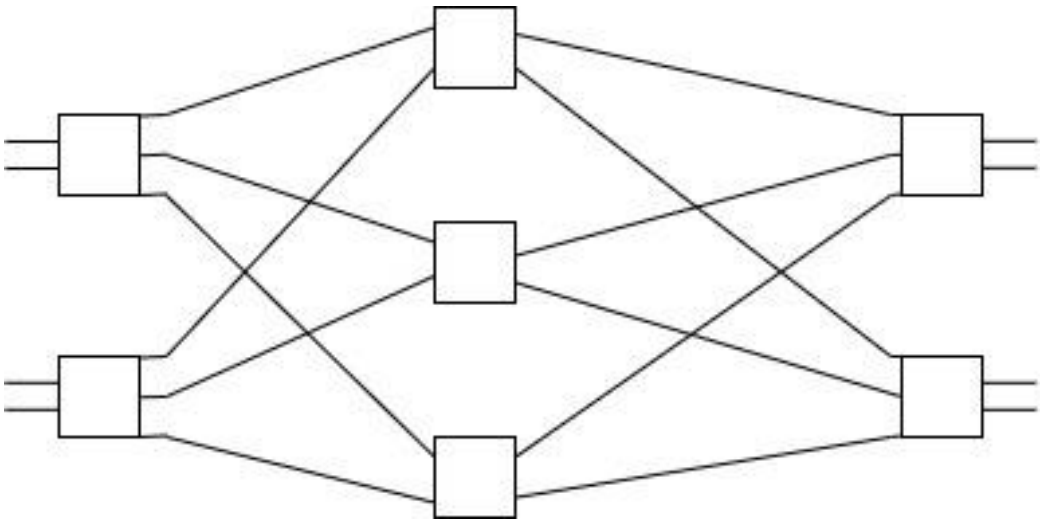
Rys. 13. Reprezentacja warstw w uczeniu głębokim

Rozdział III

Przygotowanie modelu

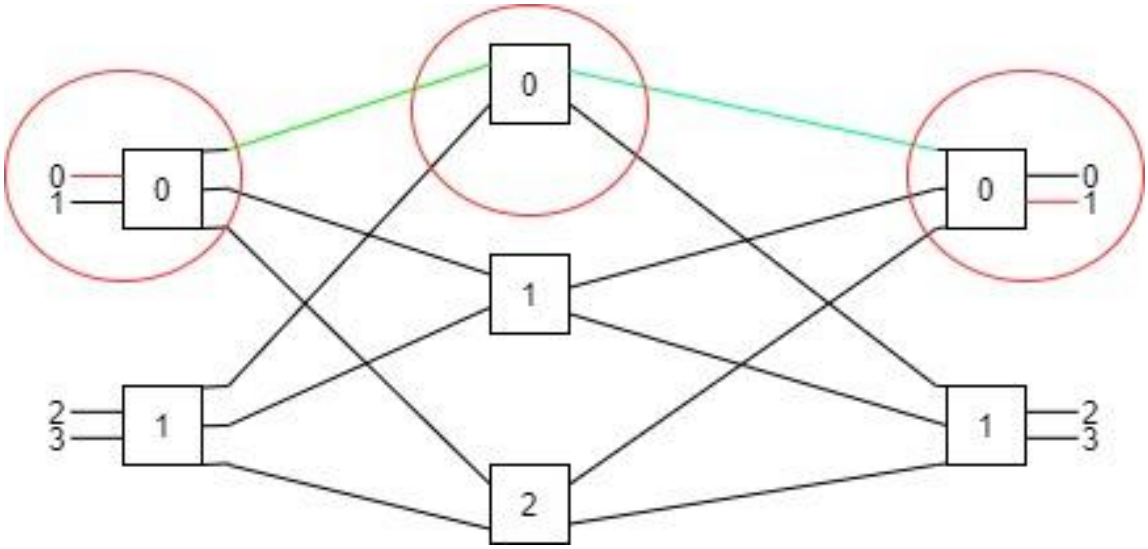
Dane uczące

W celu uczenia modelu potrzebne są dane uczące. W tym przypadku będą to dane przedstawiające ruch w polu komutacyjnym. Dane są sztucznie wygenerowane z narzędzia dostarczonego przez promotora. Narzędzie przyjmuje kilka parametrów wejściowych takich jak, liczba wejść komutatora sekcji pierwszej i wyjść komutatora sekcji ostatniej *n*, liczna komutatorów sekcji środkowej *m*, liczba komutatorów sekcji skrajnych *r*, liczba scenariuszy obciążeń, natężenie ruchu na wejściach pola *A* (Erl/wejście) dla każdego z scenariuszy, liczba zdarzeń oraz nazwa wykorzystywanego algorytmu wyboru drogi połączeniowej. Dodatkowo narzędzie pracuje na strukturze pola Closa. Dane po uporządkowaniu przyjmują format „parametr1” + separator +”parametr2”+ separator itd.. Parametrami odpowiednio w kolejności są, numer wejścia żądanego, numer wyjścia żądanego, numer komutatora sekcji pierwszej, numer komutatora sekcji trzeciej, numer komutatora sekcji środkowej. W dalszej części znajdują się dane przestawiające stan pola sprzed zestawienia danego połączenia w formacie ciągu numerów komutatorów sekcji pierwszej i trzeciej użytych do zestawienia połączenia miedzy poszczególnymi komutatorami rozdzielonego znakiem „:”. Po ustawieniu parametrów *n*=2, *m*=3, *r*=2 otrzymamy następującą strukturę pola komutacyjnego (rys. 14).



Rys. 14. Wygenerowana struktura pola komutacyjnego przy podanych parametrach

Po uruchomieniu narzędzia i wyłuskaniu jednego przykładowego wiersza wygenerowanych danych mamy: 0;1;0;0;0;. W takim przypadku mamy zestawione połączenie z wejścia 0 na wyjście 1 gdzie w sekcji pierwszej korzystamy z komutatora numer 0 tak samo na wyjściu, dodatkowo w sekcji środkowej skorzystamy z komutatora 0 do zestawienia połączenia (rys. 15). Był to jeden z pierwszych wierszy, więc nie mamy do odczytu stanu pola sprzed zestawienia połączenia.



Rys. 15. Interpretacja danych uzyskanych z narzędzia do generowania ruchu w polu komutacyjnym

Maszyna wektorów wspierających

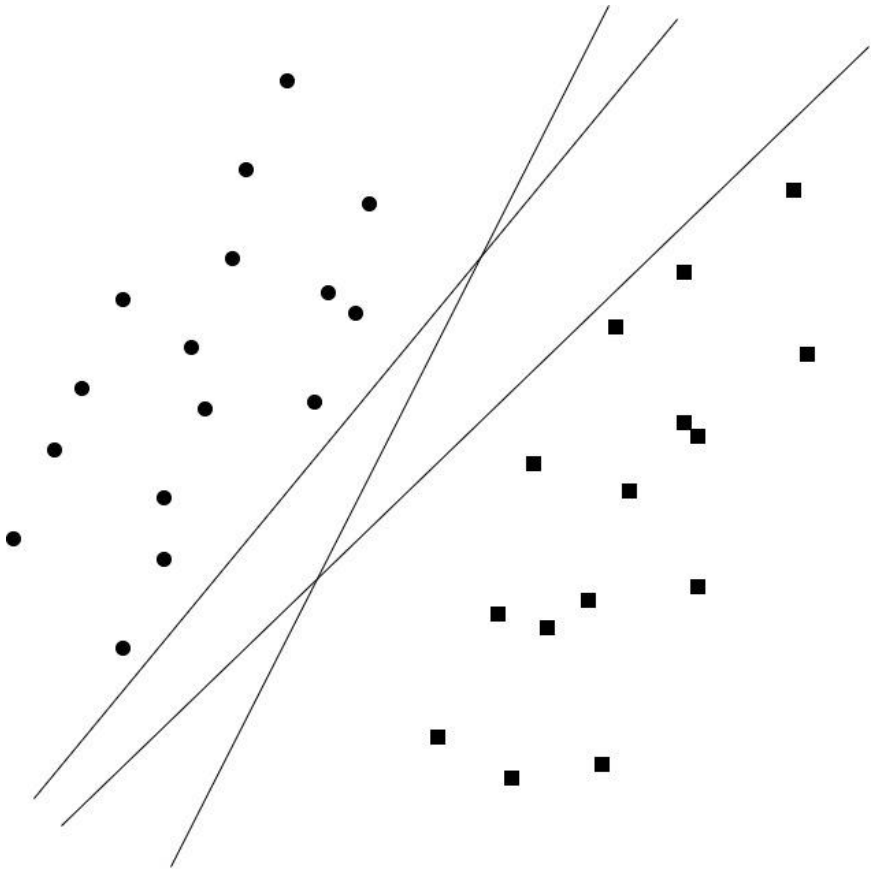
Maszyna wektorów wspierających jest bardzo dokładną i jednocześnie skomplikowaną matematycznie metodą klasyfikacji binarnej. Polega ona na wyznaczeniu hiperpłaszczyzny, która podzieli zbiór danych na dwie klasy, które reprezentowane są liczbowo przez wartości 1 i -1. W podstawowej formie zbiór ten musi być liniowo separowalny (rys. 16), wszystkie punkty ze zbioru muszą spełniać warunki:

$$wx_i + b \geq 1, dla\ y = 1$$

$$wx_i + b \leq 1, dla\ y = -1, gdzie$$

w- normalna względem hiperpłaszczyzny,

x- punkt w przestrzeni/wektor atrybutów.



Rys. 16. Dane separowalne liniowo

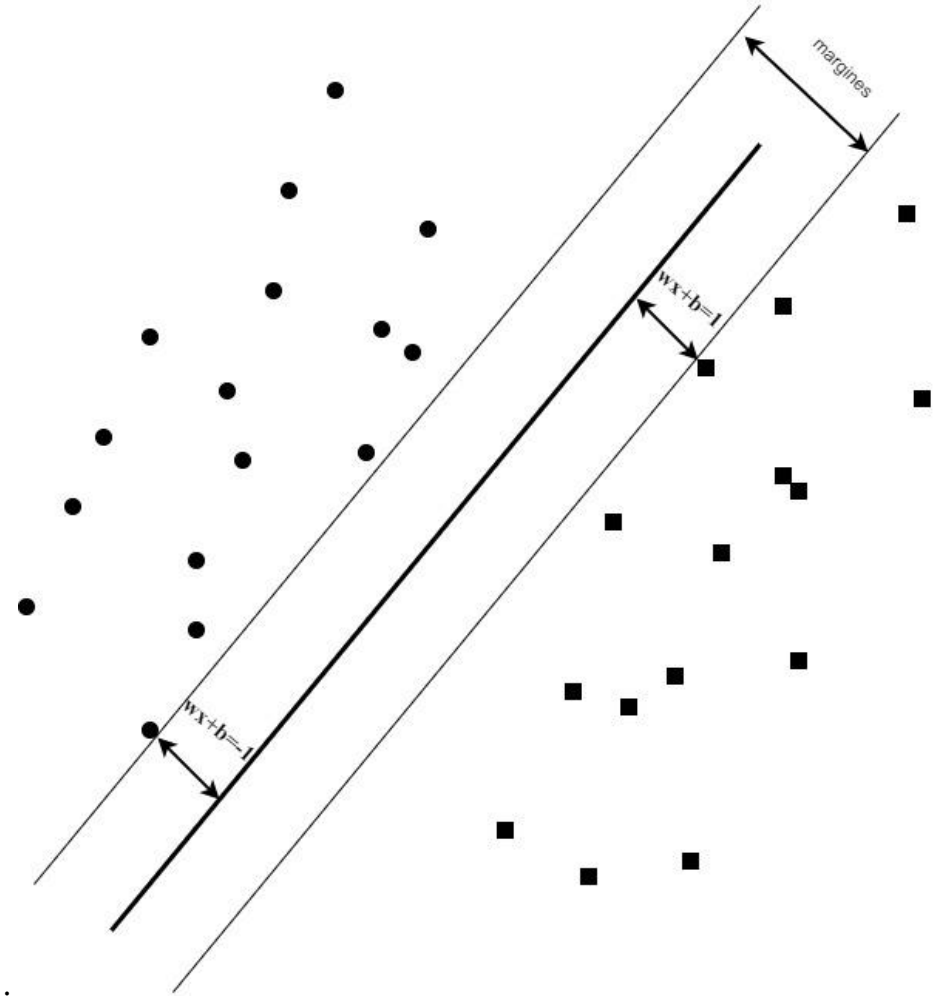
W przeciwieństwie do zwykłej klasyfikacji binarnej w SVM (ang. **S**upport **V**ector **M**achines) szukamy jak największego marginesu przy podziale zbioru, ze względu na możliwość uniknięcia przeuczenia oraz lepsze właściwości klasyfikacji (rys. 17). Przy szerokim marginesie mamy większą swobodę w ustawieniu granicy, co przełoży się na mniejsze zmiany w wynikach klasyfikacji.

Aby znaleźć jak najszerszy margines należy wziąć odległość między wektorami wsparcia, która wynosi [7]:

$$\frac{2}{\|w\|}$$

i korzystając z twierdzenia, że problem maksymalizacji można rozważyć, jako minimalizację formy odwróconej odległość wektorów można zapisać, jako:

$$\frac{\|w\|}{2}$$



Rys. 17. Margines w SVM

Następnie przekształcając ją w

$$\frac{\|w\|^2}{2}$$

,

aby usunąć funkcję nieliniową z pierwiastkiem, ponieważ norma w jest to skalar z wektora wag, który ma postać:

$$\|w\| = \sqrt{{w_1}^2 + \cdots + {w_p}^2}$$

i poddać ją minimalizacji. Zatem

$$min = \frac{\|w\|^2}{2}$$

przy warunkach ograniczających

$y_i(wx_i + b) \geq 1$, dla $i = 1,2,3, \ldots, N$

jest to problem optymalizacji kwadratowej z liniowymi ograniczeniami, które można rozwiązać metodą mnożników Lagrange'a. Zatem,

$$L(w, b, a) = \frac{\|w\|^2}{2} - \sum_{i=0}^{N} \alpha_i (y_i(wx_i + b) - 1)$$

,

różniczkując funkcje Lagrange'a L względem w oraz b i wykorzystując przy przekształceniach ograniczenia Karush-Kuhn-Tucker na mnożniki, mówiące, że

$$\alpha_i \geq 0$$

oraz

$$\alpha_i [y_i(w * x_i + b) - 1] = 0$$

Przechodzi się na postać dualną funkcji L.

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (x_i * x_j)$$

przy ograniczeniach

$$\alpha_i \geq 0, \forall_i \sum_{i=1}^{N} \alpha_i y_i = 0$$

,

następnie stosując odpowiednie przekształcenia można wyliczyć poszczególne wagi

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

i po podstawieniu do pierwotnego wzoru otrzymujemy płaszczyznę separującą w postaci

$$\sum_{i=1}^{N} \alpha_i y_i x_i * x + b = 0$$

.

Ostateczna funkcja decyzyjna przyjmuje postać:

$$f(x) = sign(\sum_{i=1}^{N} \alpha_i y_i x_i * x + b)$$

,

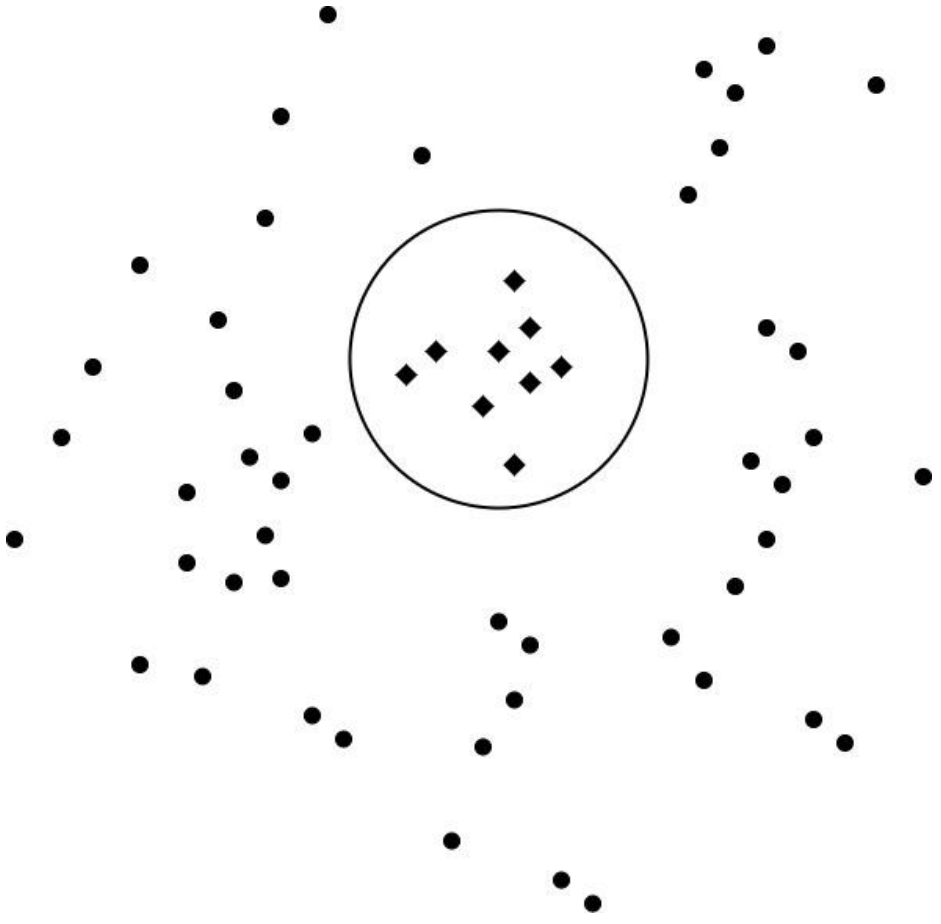
gdzie na ostateczną postać hiperpłaszczyzny wpływają wyłącznie wektory nośne ($\alpha_i > 0$) co jest konsekwencją zastosowanego wcześniej przekształcenia z wykorzystaniem ograniczeń Karush-Kuhn-Tucker.

W przypadku, gdy dane nie są separowalne liniowo (rys. 18) w SVM wykorzystuje się transformacje do wysoce wielowymiarowej przestrzeni, czyli projekcji danych z oryginalnego zbioru do zbioru o większej liczbie wymiarów, co pozwala na wyznaczenie hiperpłaszczyzny separującej (rys. 19). Projekcja ta nazywana jest transformacją Φ. W takim przypadku funkcja z mnożnikami Lagrange'a wygląda następująco:

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\Phi(x_i) * \Phi(x_j))$$

natomiast funkcja klasyfikująca:

$$f(x) = sign(\sum_{i=1}^{N} \alpha_i y_i \Phi(x_i) \times \Phi(x) + b)$$



Rys. 18. Dane nieseparowalne liniowo

Przy nieliniowym wykorzystaniu SVM problemem może okazać się duża złożoność obliczeniowa, ponieważ należy zastąpić sumę iloczynów w funkcji celu wielowymiarowymi funkcjami przekształceń.. Rozwiązaniem tego problemu jest zastąpienie iloczynu funkcji transformujących funkcjami jądrowymi (ang. kernel tricks). Funkcja jądra [8] zdefiniowana, jako:

$$K(x, y) = \Phi(x)\Phi(y)$$

przedstawia podobieństwo w sensie algebraicznym dwóch wektorów(np. ich iloczyn skalarny) z funkcją $\Phi(x)$ przemnożoną przez $\Phi(y)$. Najczęściej stosowane typy jąder związane z SVM to:

- Gaussowskie

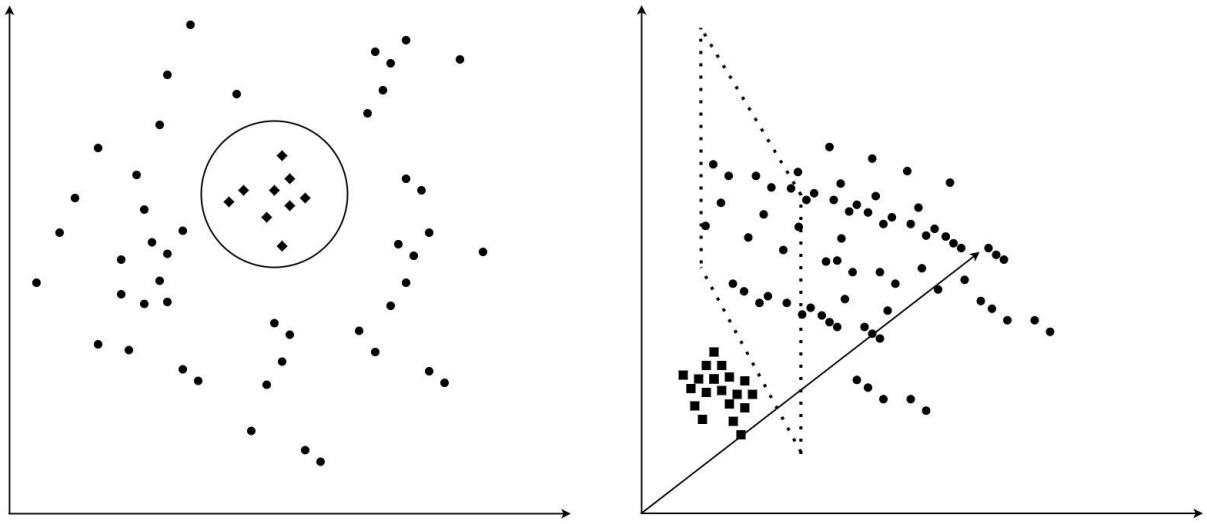
$$K(x_i, x_j) = \exp\left\{-\frac{(x_i - x_j)}{2\sigma^2}\right\},$$

- wielomianowe

$$K(x_i, x_j) = (x_i * x_j + d)^p ,$$

- sigmoidalne

$$K(x_i, x_j) = tgh(\kappa x_i * x_j - \delta).$$

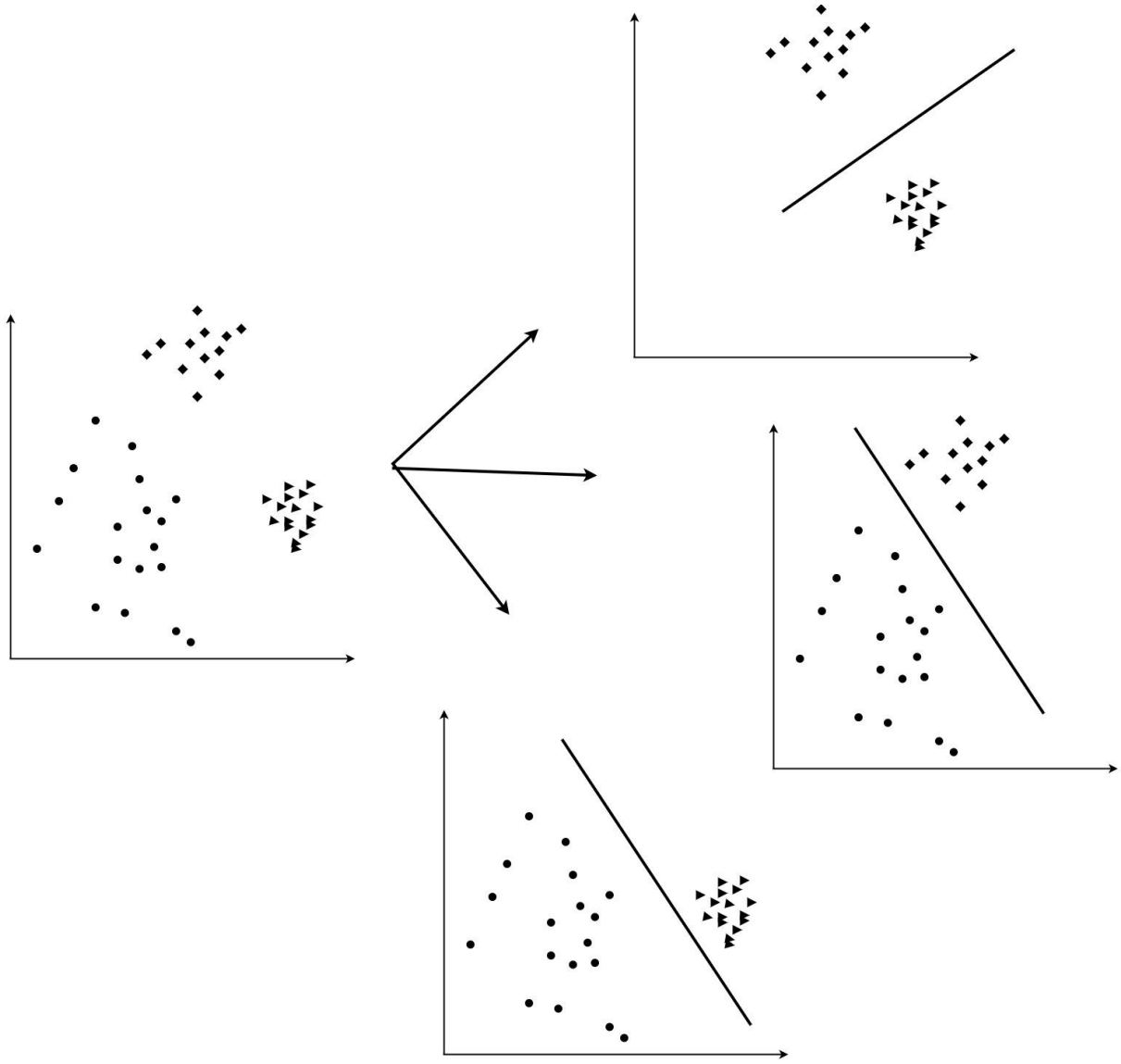


Rys. 19. Transformacja do wysoce wielowymiarowej przestrzeni

Jak zostało wyżej wspominane SVM to klasyfikator binarny, czyli taki, który dokonuje klasyfikacji na dwóch klasach. W praktyce bardzo często klasyfikacji podlegają dane, w których występują więcej niż dwie klasy, rozwiązaniem tego problemu jest użycie jednego z algorytmów:

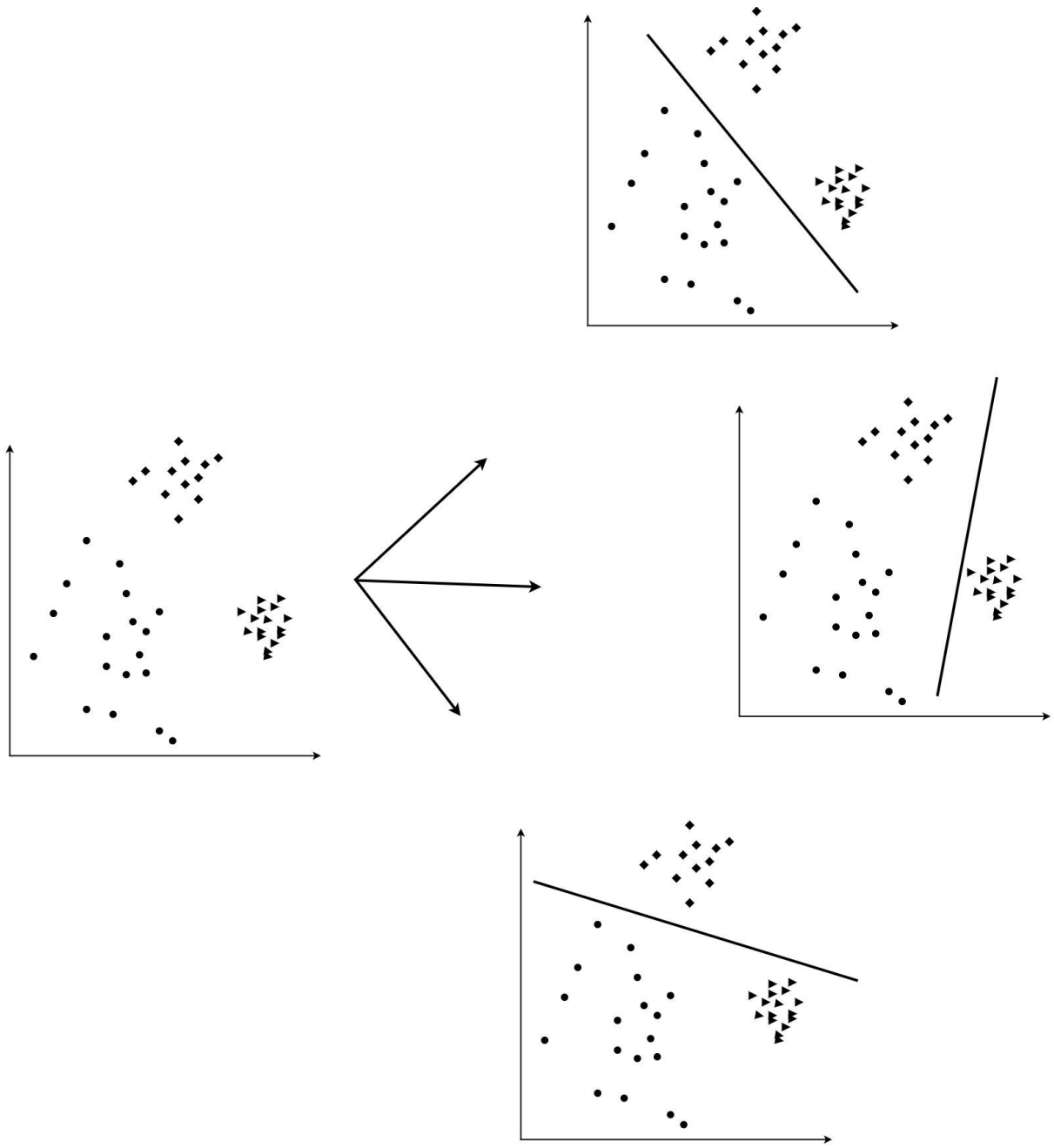
- *One Versus One ,*
- *One Versus All.*

W przypadku *One Versus One* tworzony jest klasyfikator binarny dla każdej pary klas istniejących w danym zbiorze (rys. 20), który decyduje, do której klasy dana próbka z zbioru należy. Decyzja o przynależności do danej klasy zostaje podjęta na podstawie największej ilości wystąpień w jednej z nich.



Rys. 20. Algorytm One Versus One

Natomiast w *One Versus All* tworzony jest klasyfikator dla każdej klasy (rys. 21), który rozpoznaje przynależność tylko do jednej z nich. Ostateczna decyzja zostaje podjęta na podstawie najwyższej wartości dla poszczególnej próbki w danej klasie.



Rys. 21. Algorytm One Versus All

Implementacja

Model został stworzony w środowisku Visual Studio 2017 w języku C# z wykorzystaniem biblioteki ML.NET, najnowszej i obecnie nadal rozwijanej biblioteki przez firmę Microsoft poświęconej wyłącznie uczeniu maszynowemu. Jako klasyfikator został zaimplementowany SVM z użyciem modułu Local Deep, który został opracowany przez firmę Microsoft Research w celu poprawy działania i przyspieszenia nieliniowego SVM. Jako że dane uczące były niebinarne użyto algorytmu *OneVersusAll*

Wyniki uczenia

Dla każdego modelu wygenerowano macierz błędów oraz metrykę mikro i makro trafności (ang. Accuracy). Dla pól o rozmiarze *v*(*4,3,3*) oraz algorytmu kolejnościowego metryki wyglądają następująco:

Tab. 1. Macierz błędów modelu dla pola *v*(*4,3,3*) oraz algorytmu kolejnościowego

	0	1	2	Recall
0	329	45	63	0,7529
1	68	172	68	0,7375
2	76	54	343	0,7252
Precision	0,6912	0,7876	0,7146	

- makro trafność 67,89%,
- mikro trafność 73,38%.

Natomiast dane dla pól o rozmiarze *v*(*4,3,3*) oraz algorytmu minimal index prezentują się następująco:

Tab. 2. Macierz błędów modelu dla pola *v*(*4,3,3*) oraz algorytmu minimal index

	0	1	2	Recall
0	1799	3	0	0,9983
1	15	575	17	0,9473
2	8	22	378	0,9265
Precision	0,9874	0,9583	0,9570	

- makro trafność 95,74%,
- mikro trafność 97,69%.

Metryki mają się analogicznie dla pól o rozmiarze *v*(*2,3,4*) i tak dla algorytmu kolejnościowego wyglądają one tak oto:

Tab. 3. Macierz błędów modelu dla pola *v(2,3,4)* oraz algorytmu kolejnościowego

	0	1	2	Recall
0	332	49	50	0,7703
1	46	320	33	0,8020
2	46	56	326	0,7617
Precision	0,7830	0,7529	0,7971	

- makro trafność 77,80%,
- mikro trafność 77,74%.

Natomiast dla algorytmu minial index :

Tab. 4. Macierz błędów modelu dla pola *v(2,3,4)* oraz algorytmu minimal index

	0	1	2	Recall
0	530	8	2	0,9815
1	19	316	6	0,9267
2	10	10	55	0,7333
Precision	0,9481	0,9461	0,8730	

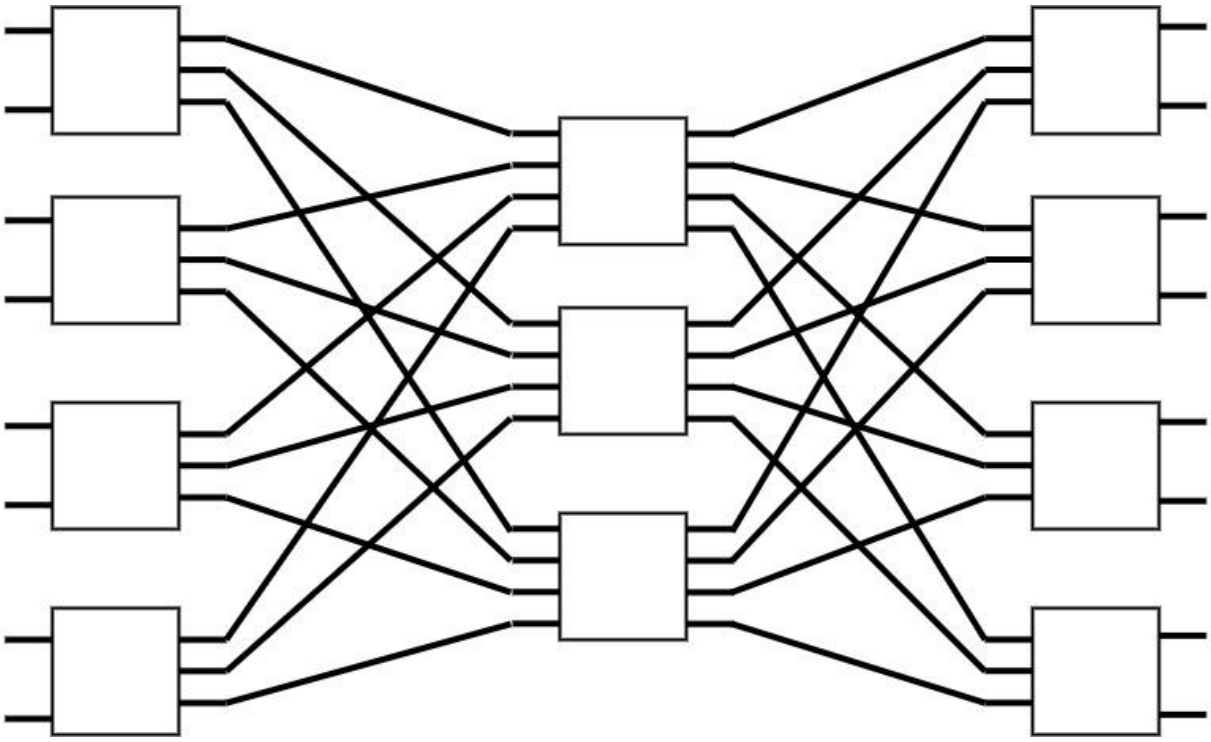
- makro trafność 88,05%,
- mikro trafność 94,25%.

Rozdział IV

Walidacja modelu

Pola nieblokowalne

Model został przetestowany na polach nieblokowalnych o rozmiarze *v*(*2,3,4*) (rys. 22). Dane uczące zostały wygenerowane na podstawie dwóch algorytmów: kolejnościowego oraz Beneša.

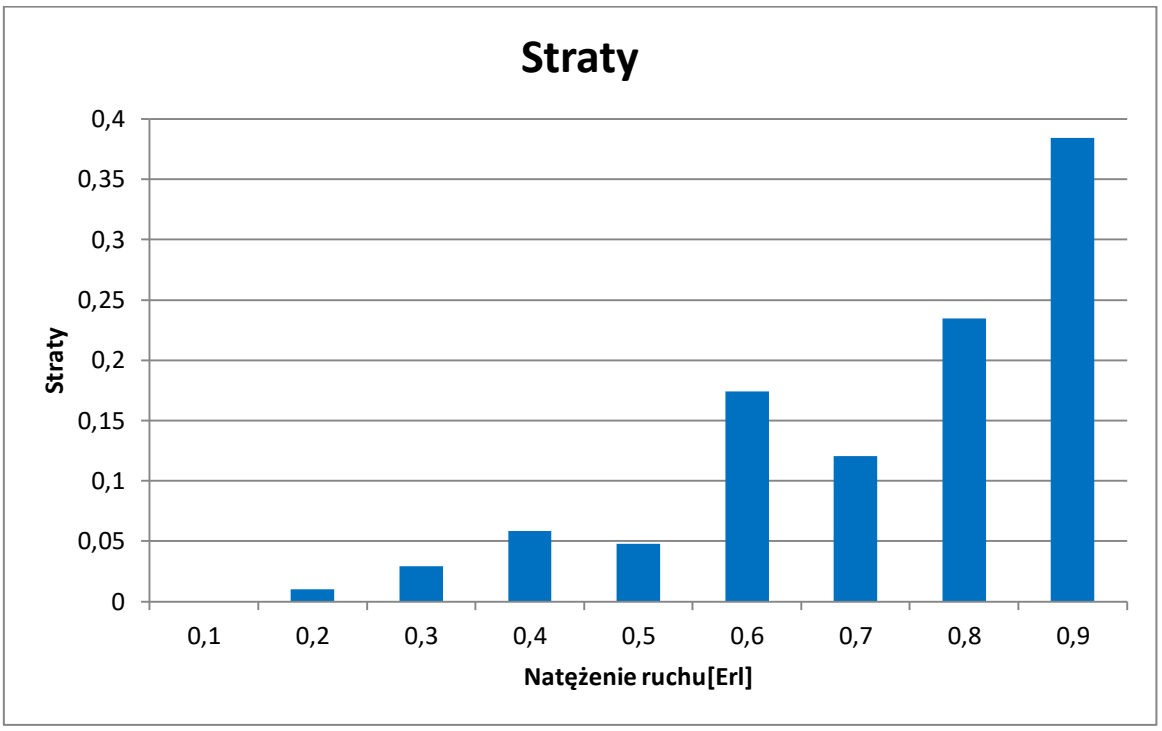


Rys. 22. Pole o rozmiarze *v(2,3,4)*

Dla obu tych algorytmów straty, czyli współczynnik liczby połączeń do zablokowanych do liczby połączeń żądanych wynosi zero. W przypadku, gdy algorytm wyboru drogi połączeniowej zastępował model uczenia maszynowego stworzony na podstawie danych z algorytmu kolejnościowego straty dla następujących obciążenia pola (Erl) wynosiły kolejno (rys. 23):

- O,1Erl- 0%
- 0,2Erl- 1%
- 0,3Erl≈ 3%
- 0,4 Er≈6%
- 0,5Erl≈5%

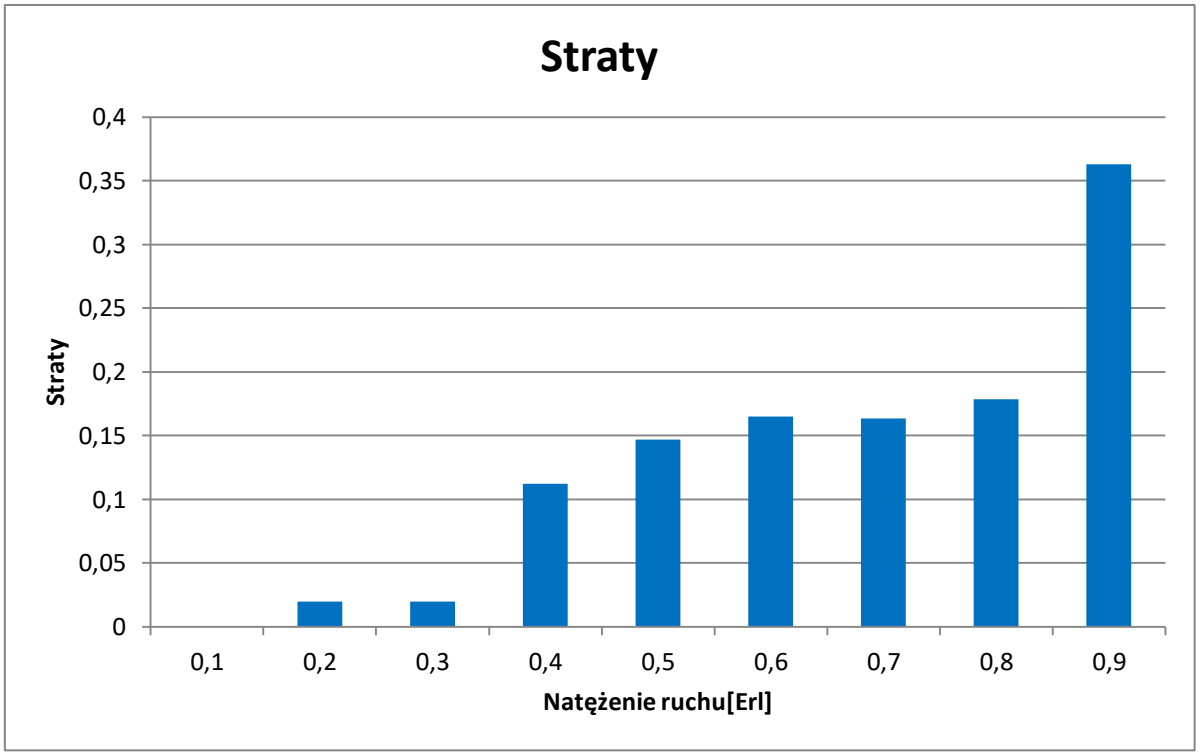
- 0,6Erl≈18%
- 0,7Erl≈13%
- 0,8Erl≈24%
- 0,9Erl-0,38%



Rys. 23. Straty dla modelu wygenerowanego na podstawie algorytmu kolejnościowego dla pola *v(2,3,4)*

W sytuacji, gdy model został stworzony na podstawie danych wygenerowanych przy użyciu algorytmu minimalindex straty wygląda następująco (rys. 24):

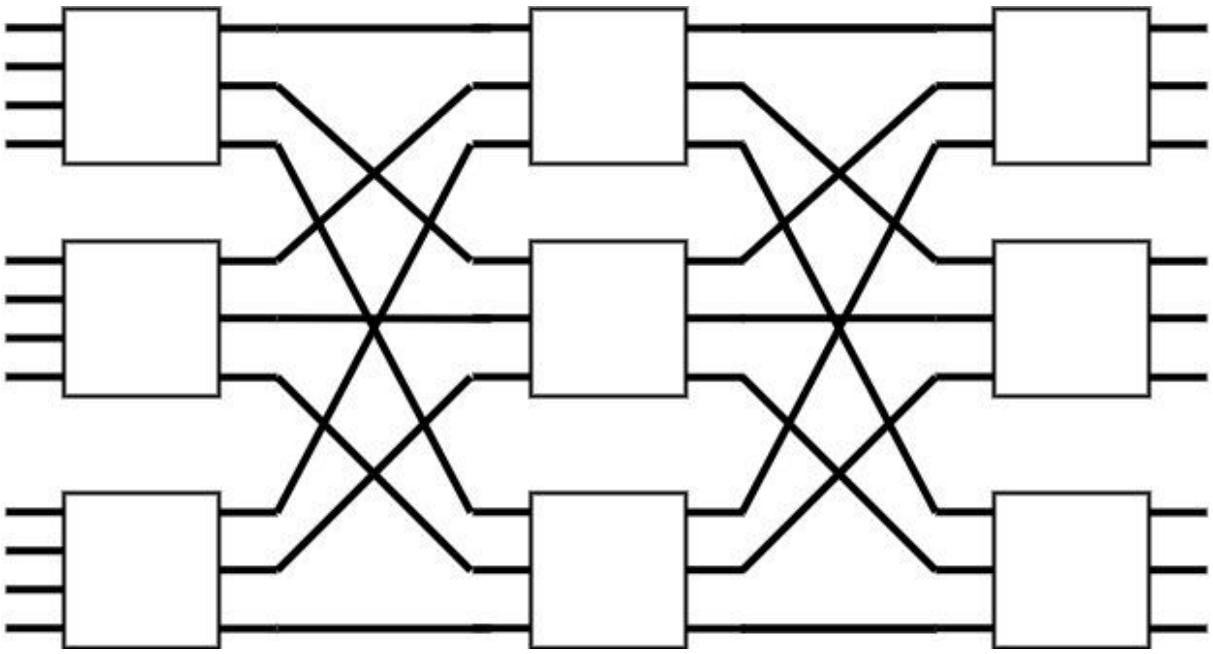
- O,1Erl- 0%
- 0,2Erl- 2%
- 0,3Erl- 2%
- 0,4 Er-≈11%
- 0,5Erl ≈15%
- 0,6Erl ≈17%
- 0,7Erl ≈17%
- 0,8Erl≈18%
- 0,9Erl-0,37%



Rys. 24.Straty dla modelu wygenerowanego na podstawie algorytmu minimalindex dla pola *v(2,3,4)*

Pola blokowalne

Model został również przetestowany na polach blokowalnych o rozmiarze *v(4,3,3)* (rys. 25). Dane uczące zostały wygenerowane na podstawie dwóch algorytmów: kolejnościowego oraz Beneša.



Rys. 25. Pole o rozmiarze v(4,3,3)

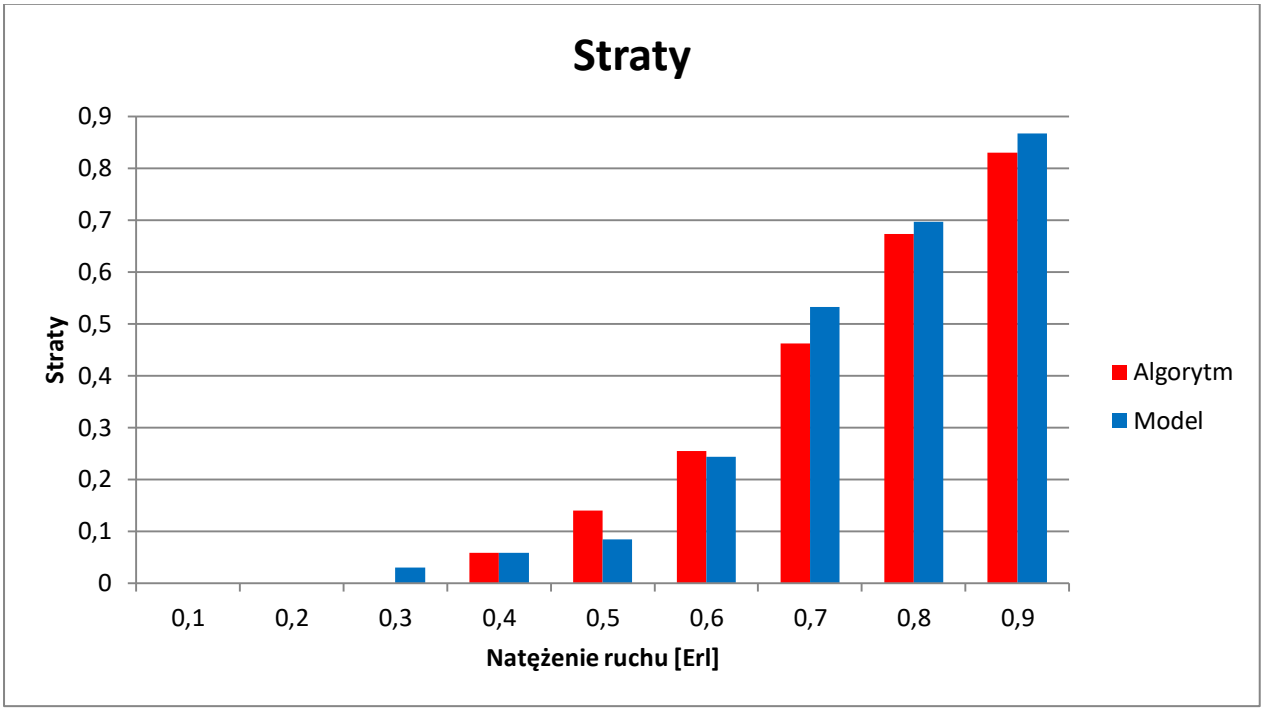
Dla algorytmu kolejnościowego oraz modelu stworzonego na podstawie danych pochodzących z algorytmu kolejnościowego starty wyglądały następująco (rys. 26),

dla algorytmu:

- O,1 Erl- 0%
- 0,2 Erl- 0%
- 0,3 Erl- 0%
- 0,4 Er≈6%
- 0,5 Erl≈14%
- 0,6 Erl≈25%
- 0,7 Erl≈46%
- 0,8 Erl≈67%
- 0,9 Erl-83%

dla modelu:

- O,1 Erl- 0%
- 0,2 Erl- 0%
- 0,3 Erl- 3%
- 0,4 Erl≈6%
- 0,5 Erl≈8%
- 0,6 Erl≈24%
- 0,7 Erl≈53%
- 0,8 Erl≈69%
- 0,9 Erl-87%



Rys. 26. Straty dla algorytmu i modelu kolejnościowego

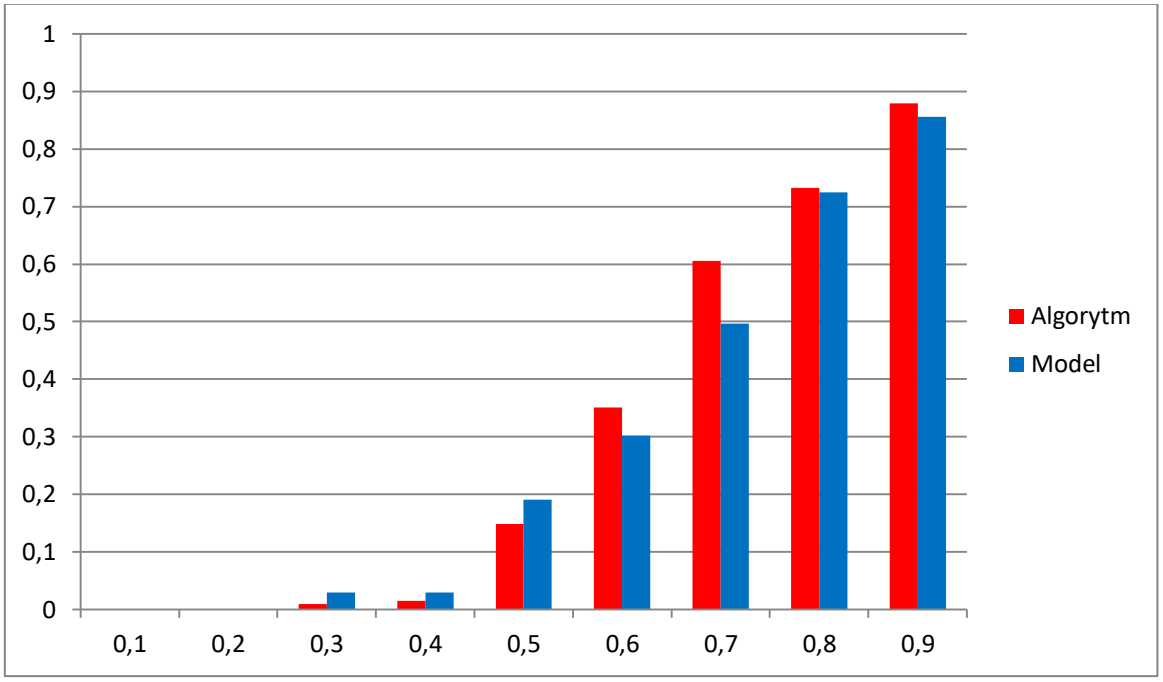
Dla algorytmu minimal index oraz modelu stworzonego na podstawie danych pochodzących z niego starty wyglądały następująco (rys. 27),

dla algorytmu:

- O,1 Erl- 0%
- 0,2 Erl- 0%
- 0,3 Erl≈1%
- 0,4 Erl-≈2%
- 0,5 Erl- ≈15%
- 0,6 Erl- ≈35%
- 0,7 Erl- ≈60%
- 0,8 Erl-≈73%
- 0,9 Erl-88%

dla modelu:

- O,1 Erl- 0%
- 0,2 Erl- 0%
- 0,3 Erl≈3%
- 0,4 Erl≈3%
- 0,5 Erl ≈19%
- 0,6 Erl ≈30%
- 0,7 Erl ≈50%
- 0,8 Erl≈73%
- 0,9 Erl≈86%



Rys. 27. Straty dla algorytmu i modelu minimal index

Wnioski

W ramach niniejszej pracy przygotowano model uczenia maszynowego opartego na danych z algorytmów wyboru drogi połączeniowej oraz przeprowadzono badania w celu porównania ich wydajności.

Pierwszy model jest oparty o dane z algorytmu kolejnościowego dla nieblokowalnych pól Closa *v*(*2,3,4*). Algorytm wyboru drogi połączeniowej nie przynosi żadnych strat gdyż pole spełnia warunki nieblokowalności w wąskim sennie, natomiast model nie poradził sobie tak dobrze. Gdy natężenie ruchu nie przekracza 0,5 Erl wyniki są obiecujące, ponieważ nie przekraczają one 5 %, lecz na wyższym obciążeniu sieci sięgają ponad 30%.

Drugim modelem był ten oparty o algorytm minimal index dla nieblokowalnych pól Closa *v*(*2,3,4*) i tak samo jak w poprzednim przypadku, algorytm wyboru drogi połączeniowej nie przynosi żadnych strat, analogiczne wyniki można zaobserwować w przypadku modelu uczenia maszynowego, z tą różnicą, że model gorzej sobie radzi na niskim obciążeniu sieci. Obiecujące wyniki są tylko dla obciążenia sięgającego 0,3 Erl, w przedziale 0,4 – 0,8 nie przekraczają one 20% i są rozłożone bardziej równomiernie niż w poprzednim modelu.

Kolejny przedstawiony model jest stworzony dla pól Closa niespełniających warunków nieblokowalności o rozmiarze *v(3,4,4)* opartym o algorytm kolejnościowy. Z wyników porównania wywnioskować można, że przynosi on zbliżone straty, w kilku scenariuszach obciążenia nawet niższe niż algorytm.

Ostatnim z porównywanych modeli jest ten dla nieblokowalnych pól Closa *v(2,3,4)* stworzony na danych z algorytmu minimal index. W tym przypadku model okazał się odrobinę lepszy niż algorytm sterowania na wyższych obciążeniach sieci, do poziomu 0,5 Erl algorytm był nieznacznie wydajniejszy.

Otrzymane wyniki jasno pokazują, że stworzony model uczenia maszynowego dla pól nieblokowalnych spełnia swoje zadanie tylko dla niskich obciążeń sieci, na wyższych niestety nie radzi sobie tak dobrze jak dedykowane do tego algorytmy wyboru drogi połączeniowej. Jeśli jednak spojrzymy na pola blokowalne, model radzi sobie tak samo dobrze jak algorytmy, dla pewnych scenariuszy nawet lepiej, co pozwala stwierdzić, że jest możliwe stworzenie modelu, który w pewnych warunkach będzie lepszym rozwiązaniem niż użycie algorytmów.

Bibliografia

1. Schalkoff, R. J. (1990). *Artificial Intelligence: An Engineering Approach.* McGraw-Hill College.
2. www.psychologia.net/artykul/php?level=784 data otwarcia: 23.03.2020
3. A. Jajszczyk (1998). Wstęp do telekomutacji. Wydawnictwa naukowo-techniczne.
4. A. Pattavina (1998). Switching Theory Architecture and Performance in Broadband ATM Networks. John Wiley & Sons
5. Matt R. Cole (2018) Hands-On Machine Learning with C#: Build smart, speedy, and reliable data-intensive applications using machine learning
6. Iansiti, Marco, Lakhani, Karim R. (2020) Competing in the Age of AI : Strategy and Leadership When Algorithms and Networks Run the World .
7. J. STEFANOWSKI, SVM – Support Vector Machines Metoda wektorów nośnych, http://www.cs.put.poznan.pl/jstefanowski/ml/SVM.pdf data otwarcia: 23.05.2020
8. Nello Cristianini, John Shawe-Taylor (2000) An Introduction to Support Vectro Machines and other kernel based learninig methods