

Wydział Informatyki Katedra Metod Programowania

Programowanie aplikacji biznesowych

Praca Inżynierska

Autor:

#### Michał Marek Zawadzki

Nr albumu: 14304

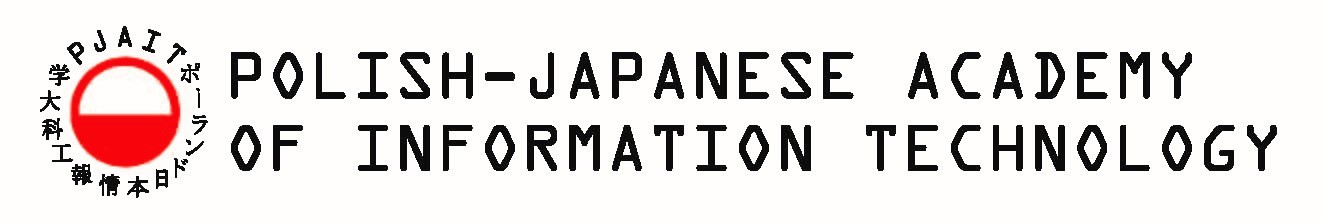
Tytuł:

Optymalizacja hiperparametrów procesu uczenia sieci neuronowych z wykorzystaniem algorytmu genetycznego

Promotor:

#### mgr inż. Piotr Gnyś

Warszawa, Luty, 2022



Faculty of Computer Science Department of Programming Methods Business Application Programming

Sc Thesis

Author:

#### Michał Marek Zawadzki

Student identification number: 14304

Title of the Thesis:

Optimization of Artificial Neuronal Network Learning Process Hiperparameters by Genetic Algorithm

Thesis supervisor:

#### Master of Engineering Piotr Gnyś

Warsaw, February, 2022

Optymalizacja hiperparametrów procesu uczenia sieci neuronowych z wykorzystaniem algorytmu genetycznego

Michał Marek Zawadzki Luty 2022

## Spis treści

##### Wstęp

1.1 Cel pracy . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

1.2 Teza główna . . . . . . . . . . . . . . . . . . . . . . . . . . . .

1.3 Słowa kluczowe . . . . . . . . . . . . . . . . . . . . . . . . . .

1.4 Keywords . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

1.5 Abstrakt . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

1.6 Abstract . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

##### Wpływ cech utworów muzycznych na popularność

1. **Wykorzystanie sieci neuronowych do problemu regresji dla tabelarycznego zbioru danych**
   1. Sztuczne sieci neuronowe a biologia . . . . . . . . . . . . . . .

3.2 Perceptron . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

3.3 Uczenie wielowarstwowych sieci przy użyciu propagacji wstecznej

##### Algorytmy Genetyczne

* 1. Algorytm genetyczny a teoria ewolucji . . . . . . . . . . . . .

4.2 Cykl życia populacji . . . . . . . . . . . . . . . . . . . . . . .

4.3 Inicjacja populacji . . . . . . . . . . . . . . . . . . . . . . . .

4.4 Ewaluacja osobników . . . . . . . . . . . . . . . . . . . . . . .

4.5 Wybór rodziców . . . . . . . . . . . . . . . . . . . . . . . . . .

4.6 Krzyżówki . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

4.7 Mutacje . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

4.8 Wybór potomstwa . . . . . . . . . . . . . . . . . . . . . . . .

4.9 Warunek końcowy . . . . . . . . . . . . . . . . . . . . . . . . .

##### Opis zbioru danych

1. **Zastosowanie dwustopniowego algorytmu uczenia maszynowego do predykcji popularności utworu**
   1. Selekcja i wstępna obróbka danych . . . . . . . . . . . . . . .
   2. Ogólny zarys architektury zastosowanej sieci neuronowej . . .
   3. Inicjacja wag funkcją HE . . . . . . . . . . . . . . . . . . . . .

6.4 Funkcja aktywacji ELU . . . . . . . . . . . . . . . . . . . . . .

* 1. Normalizacja wsadowa (ang. *Batch Normalization*) . . . . . .
  2. Regularyzacja przez Porzucanie (ang. *Dropout*) . . . . . . . .
  3. Optymalizator Adam (ang. *Adaptive Moment Estimation*) . .
  4. Wybór hiperparametrów procesu uczenia sieci neuronowej . . .
  5. Wybór rodziców z wykorzystaniem selekcji proporcjonalnej (ang.

*Fitness Proportionate Selection / FPS* ) . . . . . . . . . . . . .

* 1. Krzyżówka z jednym punktem przecięcia (ang. *One Point Crossover* )
  2. Mutacja pełzająca (ang. *Creep Mutation*), niejednorodna (ang.

*nonuniform*) . . . . . . . . . . . . . . . . . . . . . . . . . . . .

1. **Rezultaty eksperymentów**
2. **Wnioski**

## Wstęp

#### Cel pracy

Celem pracy jest stworzenie modelu (sieci neuronowej) do predykcji popularności, jaką może uzyskać utwór muzyczny na platformie Spotify, na podstawie

jego cech muzycznych. Wykorzystany zostanie zbiór danych stworzony z wykorzystaniem publicznego API Spotify i opublikowany na platformie Kaggle. Do dobrania optymalnych hiperparametrów procesu uczenia modelu zostanie zastosowany algorytm genetyczny.

#### Teza główna

Algorytm genetyczny może pomóc w doborze zbliżonych do optymalnych hiperparametrów procesu uczenia sieci neuronowej i tym samym przyczynić się znacząco do poprawy dokładności predykcji finalnego modelu - o 1–2% licząc na podstawie metryki MSE (ang. *root mean squared error* pl. błąd

średniokwadratowy) w porównaniu do sieci neuronowych inicjowanych wartościami sugerowanymi jako wyjściowe w literaturze przedmiotu lub dobranych intuicyjnie metodą ‘prób i błędów’.

#### Słowa kluczowe

* + - algorytm genetyczny (ang. *genetic algorithm*) - Metaheurystyka czerpiąca inspirację z ewolucji biologicznej, należąca do grupy algorytmów ewolucyjnych; znajduje zastosowanie przy problemach optymalizacji i wyszukiwania, dostarczając wysokiej jakości rozwiązania poprzez bazowanie na operacjach inspirowanych biologią takich jak mutacja, krzyżówka i selekcja[Mit96].
    - genom (ang. *genome*) - W biologi molekularnej i genetyce określa się tak kompletną informację genetyczną organizmu[Rot19].
    - gen (ang. *gene*) - W biologii gen (od greckiego *genos*) jest podstawową jednostką dziedziczenia; reprezentowany jako sekwencja nukleotydów w DNA, która koduje syntezę produktu genu, albo RNA lub białko[Coma].
    - funkcja dopasowania (ang. *fitness function*) - W algorytmie genetycznym reprezentuje wymagania, do których powinna adaptować się populacja; jest podstawą mechanizmu selekcji i tym samym pośrednio przyczynia się do ogólnej poprawy jakości osobników w kolejnych pokoleniach[ES15].
    - selekcja naturalna (ang. *natural selection*) - Pojęcie wywodzące się z teorii ewolucji Charlesa Darwina i odgrywające w niej kluczową rolę;

determinuje, które osobniki w populacji będą mieć większe szanse na reprodukcję; faworyzuje osobniki, które rywalizują o dostępne zasoby w sposób najbardziej efektywny[ES15].

* + - krzyżówka (ang. *crossover* ) - Operacja stochastyczna polegająca na rekombinacji materiału genetycznego rodziców poprzez losowo wybrane punkty przecięcia[ES15].
    - mutacja (ang. *mutation*) - Operacja stochastyczna wprowadzająca losową zmianę do genotypu osobnika, której wynikiem jest mutant[ES15].
    - sztuczna sieć neuronowa (ang. *artificial neural network*, ANN) - System obliczeniowy przeznaczony do przetwarzania informacji, czerpiący inspirację z biologicznego układu nerwowego[TS15].
    - jednostka liniowa z progiem (ang. *linear threshold unit*, LTU) - Modyfikacja sztucznego neuronu, będąca podstawą prostej sieci neuronowej; wylicza sumę ważoną sygnałów wejściowych, a wobec wyniku stosuje funkcję skokową Heaviside’a[Gé18].
    - propagacja wsteczna (ang. *backpropagation*) - Podstawowy algorytm uczenia głębokich sieci neuronowych; modyfikacja metody gradientu prostego, z użyciem odwrotnego różniczkowania automatycznego[Gé18].

#### Keywords

* + - *genetic algorithm* - Metaheuristics inspired by biological evolution, belonging to the group of evolutionary algorithms; used in optimization and search problems, providing high-quality solutions by relying on biology-inspired operations such as mutation, crossover and selection[Mit96].
    - *genome* - In molecular biology and genetics this term indicates a complete genetic information of an organism[Rot19].
    - *gene* - In biology, a gene (from the Greek *genos*) is the basic unit of inheritance; represented as a sequence of nucleotides in DNA that codes for the synthesis of a gene product, either RNA or protein[Coma].
    - *fitness function* - In the genetic algorithm, it represents the requirements to which the population should adapt; it is the basis of the selection mechanism and thus contributes to the overall improvement of the quality of individuals in subsequent generations[ES15].
    - *natural selection* - Concept derived from Charles Darwin’s theory of evolution, playing a key role in it; determines which individuals in the population will have a better chance to reproduce; it favors individuals that compete for available resources most efficiently[ES15].
    - *crossover* - A stochastic operation that recombines the parents’ genetic material through randomly selected intersection points[ES15].
    - *mutation* - A stochastic operation that introduces a random change to an individual’s genotype, which results in creation of a mutant[ES15].
    - *artificial neural network*, ANN - A computing system designed to process information, inspired by the biological nervous system[TS15].
    - *linear threshold unit*, LTU - Modification of an artificial neuron, which is the base unit of a simple neural network; computes a weighted sum of the inputs and applies a Heaviside step function to the result[Gé18].
    - *backpropagation* - Basic algorithm for learning deep neural networks; modification of the gradient descent method, with the use of inverse automatic differentiation[Gé18].

#### Abstrakt

W sekcji pierwszej opisano szczegółowo składowe kształtujące preferencje muzyczne. W dalszej części omówiono znaczenie i sposób działania platform muzycznych, ze szczególnym uwzględnieniem szwedzkiego Spotify. Stamtąd też pochodzi zbiór danych będący przedmiotem badań niniejszego opracowania. Następnie wspomniano o najbardziej znanych eksperymentach z szacowaniem popularności utworów na bazie ich cech muzycznych. Na sam koniec oceniono szanse na stworzenie wysokiej klasy modelu, biorąc pod uwagę posiadane dane i narzędzia.

W sekcji drugiej skrótowo opisano historię powstania sztucznych sieci neuronowych i biologiczne inspiracje, które za tym stały. Następnie omówiono budowę i sposób działania perceptronu, czyli jednej z pierwszych implementacji sieci neuronowych. Na zakończenie przedstawiono zasadę działania algorytmu propagacji wstecznej - podstawową metodę uczenia głębokich sieci neuronowych.

Sekcja trzecia jest wprowadzeniem do tematu algorytmów genetycznych. Opisano w niej, w telegraficznym skrócie odkrycia Charlesa Darwina i Gregora Johanna Mendla, które następnie posłużyły Alanowi Turingowi jako baza dla jego ‘ewolucyjnego poszukiwania’ - czyli pierwowzoru wspomnianego algorytmu.

W dalszej części omówiono cykl życia populacji w ramach algorytmu. Potem kolejno opisano jego kluczowe komponenty: inicjację populacji, ewaluację osobników, selekcję rodziców, krzyżowanie, mutację genomu, wybór potomstwa i warunek końcowy.

Sekcja czwarta to opis wybranego zbioru danych. Po przedstawieniu najważniejszych informacji opisano szczegółowo wszystkie cechy, wykorzystując do tego celu dokumentację deweloperską udostępnioną przez Spotify. Dodatkowo dla wszystkich parametrów dyskretnych utworzono czytelne histogramy, a dla flag i kategorii

* wykresy kołowe.

W sekcji piątej opisano przygotowania do przeprowadzenia eksperymentu w tym wstępną obróbkę danych, bazową architekturę sieci neuronowej (SN) wykorzystaną w algorytmie, wybrane elementy SN, selekcję hiperparametrów do wyznaczenia oraz najważniejsze komponenty implementacji algorytmu genetycznego.

W sekcji szóstej zaprezentowano przebieg wykonania algorytmu genetycznego: zmiany wartości funkcji dopasowania dla najgorszego i najlepszego osobnika, średnią oraz medianę. Pokazano również, jak zmieniały się wartości poszczególnych hiperparametrów dla najlepszego osobnika. Następnie przedstawiono wyniki dokładnego uczenia finalnego modelu. Wyniki te zestawiono z rezultatami uzyskanymi po wytrenowaniu innych modeli o podobnej architekturze. Na koniec pokazano przykłady najlepszych i najgorszych predykcji finalnego modelu.

W ostatniej sekcji podsumowano prace. Zaproponowano modyfikacje, które mogłyby przyczynić się do poprawy wyników. Oceniono wyniki. Wypunktowano również czynniki, które przypuszczalnie miały największy wpływ na wielkość błędów.

#### Abstract

The first section describes in details the components shaping musical preferences. The following section covers the meaning and operating principle of music platforms, with an emphasis on Spotify from Sweden. This is also where the dataset, that is the subject of this study’s research, comes from. Next, the most famous experiments with estimating the popularity of songs based on their musical characteristics are mentioned. In the end, the chances of creating a high-class model are assessed, taking into account the available data and tools.

The second section briefly describes the history of the creation of artificial neural networks and the biological inspirations behind it. Then, the structure and operating principle of the perceptron, one of the first implementations of neural networks, is discussed. Finally, the principle of the backpropagation algorithm is presented - the basic method of teaching deep neural networks.

The third section introduces genetic algorithms. It briefly describes the discoveries of Charles Darwin and Gregor Johann Mendel, which then served Alan Turing as the basis for his ‘evolutionary search’ - that is the prototype of the above-mentioned algorithm. The following section discusses the life cycle of the population as part of the algorithm. The key components are then described: population initiation, individual evaluation, parent selection, crossover, genome mutation, offspring selection and the end condition.

The fourth section is a description of the selected dataset. After presenting the highlights, all features are detailed using information from the development documentation provided by Spotify. Additionally, clear histograms are created for all discrete parameters, and pie charts for flags and categories.

The fifth section describes the preparations for the experiment, including the initial data processing, the basic artificial neural network (ANN) architecture used in the algorithm, specific ANN elements, the selection of hyperparameters to be determined, and the most important components of the implementation

of the genetic algorithm.

The sixth section presents the execution phase of the genetic algorithm: changes in the value of the fitness function for the worst and best individual,

the mean, and the average. It also shows how the values of individual hyperparameters changed for the best individual. Then the results of the comprehensive training

of the final model are presented. These results are compared with those obtained after training other models with similar architecture. Finally, examples of the best and worst predictions of the final model are shown.

The work is summarized in the last section. Modifications are proposed that could improve the results. The results are assessed. Factors that probably had the greatest impact on the size of errors are also listed.

## Wpływ cech utworów muzycznych na popularność

Zaczynając rozważania nad przyczynami, które sprawiają, że niektóre utwory muzyczne osiągają ogromną popularność, podczas gdy inne nie zdobywają szerokiego uznania, trzeba najpierw zdefiniować podstawowe czynniki, jakimi kieruje się odbiorca przy dokonywaniu wyborów muzycznych. Magdalena Parus-Jankowska i Szymon Nożyński w swojej pracy[PJN20] wymieniają 3 takie czynniki:

* *upodobania* - Przywiązanie do konkretnych gatunków muzycznych, wraz z otaczającym je kontekstem kulturowym.
* *smak/gust* - Umiejętność dostrzegania i oceny poszczególnych elementów składowych utworu, także znajomość i zrozumienie przesłania utworu.
* *preferencje* - Przywiązanie do konkretnych utworów lub twórców muzycznych.

Czynniki te często kształtują się równolegle i przenikają znaczeniowo, dlatego też dla uproszczenia w dalszej części posłużono się szeroko rozumianym pojęciem preferencji muzycznych.

Preferencje muzyczne człowieka nie są czymś stałym. Zmieniają się one przez całe życie, szczególnie intensywnie w wieku dziecięcym i w okresie dojrzewania[PJN20]. Wielki wpływ na sposób konsumpcji treści muzycznych ma środowisko, z którego człowiek się wywodzi, jego wykształcenie, zamożność, światopogląd i osobowość[PJN20]. Muzyki słuchają najczęściej ludzie młodzi

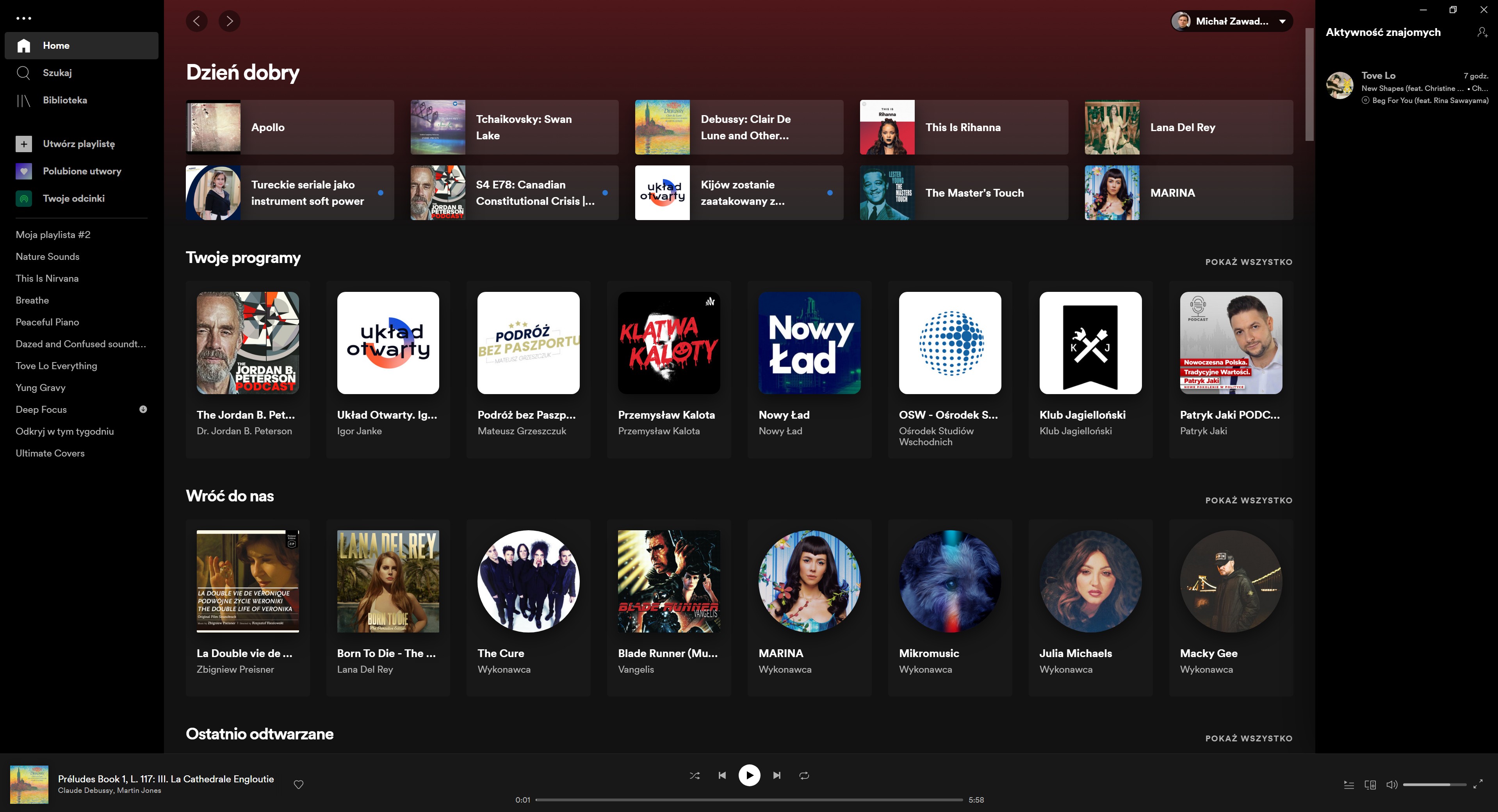
(w grupie wiekowej 18–24 lata 77% deklarowało słuchanie muzyki codziennie[Fel18]) i to oni pośrednio, a bezpośrednio wywodzący się z tej grupy twórcy, kształtują

nowe trendy i czynią rynek muzyczny bardzo dynamicznym.

Wejście na rynek platform streaming-owych, których sztandarowym przykładem jest szwedzki gigant Spotify, dokonało małej rewolucji na rynku muzycznym.

Platforma działa w dwóch modelach płatności: bezpłatnym (z reklamami) i subskrypcyjnym. Można korzystać z niej za pośrednictwem przeglądarki, ale dostępne są także dedykowane aplikacje na praktycznie każdy system, zaczynając od smartcaru na smartfonie kończąc, między którymi można się swobodnie przełączać. Muzyki można słuchać zarówno w trybie online, jak i w offline (oczywiście po wcześniejszym pobraniu na urządzenie). Co ważne zmianie uległ sposób konsumpcji samych treści. Autor niniejszej pracy, jako szczególnie aktywny użytkownik platformy, uznał za stosowne wymienienie następujących alternatywnych kanałów:

* *strona główna (ang. home)* - Najmocniej promowane przez platformę treści, również spersonalizowane sekcje z dedykowanymi playlistami, a także szybki dostęp do treści ostatnio odtwarzanych.
* *radio wykonawcy, radio utworu* - Pseudo-radio, a właściwie strumień muzyczny generowany przez algorytm Spotify dla danego wykonawcy lub konkretnego utworu muzycznego.
* *aktywność znajomych* - Trzeba nadmienić, że Spotify jest też platformą społecznościową. Istnieje możliwość wglądu w bieżącą aktywność znajomych (o ile słuchają muzyki w trybie sesji publicznej).
* *playlisty publiczne* - Playlisty stworzone przez pracowników platformy, algorytmy oraz członków społeczności. Równolegle do klasycznego podziału na gatunki muzyczne, platforma proponuje także bardziej utylitarne kategorie, jak na przykład muzykę dobrą do ćwiczeń, czytania czy wyciszenia się przed snem.
* *playlisty prywatne* - Playlisty stworzone przez użytkownika, nieupublicznione.
* *biblioteka* - Treści pobrane przez użytkownika.



Rysunek 2.1: Interfejs aplikacji desktopowej Spotify na system Windows.

Wewnętrzna polityka platform muzycznych w zakresie promowania i pozycjonowania treści (w którą niestety nie ma wglądu, a która z ogromną dozą prawdopodobieństwa

dyktowana jest przede wszystkim chęcią powiększania zasięgów i poprawy wyników finansowych) ma niebagatelny wpływ na wybory dokonywane przez użytkowników[PJN20]. Zdaniem autora niniejszej pracy szczególnie duże znaczenie w tym aspekcie mają kryteria, na których oparty jest mechanizm wyszukiwania,

a także (a może przede wszystkim) aspekt losowania kolejnych utworów w usłudze *radio*. Z drugiej strony prawdziwe zdaje się być twierdzenie, że dobra muzyka obroni się sama. Bez względu na to jak bardzo konsument będzie zasypywany promowanymi treściami i tak regularnie będzie wracać tylko do tego co najbardziej trafiło w jego gusta.

Christoph Drösser, popularny w Niemczech dziennikarz popularnonaukowy, napisał “Muzyka jest [. . . ] doświadczeniem całościowym, w znacznym stopniu nierozsądnym, którego nie da się ogarnąć za pomocą zimnego, racjonalnego instrumentarium nauki”[Drö21]. Ale czy to aby na pewno prawda? Na początku

20. wieku w mainstreamie zaczęło funkcjonować pojęcie *Hit Song Science* (pl. nauka o przebojach muzycznych), za którego twórcę i głównego promotora uważa się Mike’a McCready[Comb]. McCready wraz ze współpracownikami, działając w ramach firmy Polyphonic HMI, starał się rozwijać i sprzedawać wytwórniom muzycznym narzędzia, oparte o techniki *MIR* (ang. *music information retrieval*, pl. pozyskiwanie informacji muzycznych z utworu) i *data science*, ułatwiające przewidywanie czy dany utwór ma szanse stać się ’hitem’[Comc]. Odnośnie podstaw naukowych *Hit Song Science* od lat, w szczególności w społeczności *MIR*, toczy się burzliwa, nierozstrzygnięta do końca debata[Comb].

François Pachet i Pierre Roy z firmy Sony Computer Science Laboratories, Inc. w artykule “Hit Song Science Is Not Yet a Science” z 2009-go roku[PR09] opisują swoją nieudaną próbę stworzenia klasyfikatora szacującego popularność utworów na podstawie ich cech muzycznych oraz subiektywnych ‘ludzkich’ etykiet przypisanych przez profesjonalnych ankieterów. Danymi, na których pracowali była baza 32978 utworów dostarczona przez firmę HiFind. W swoim eksperymencie, oprócz zaawansowanych technik *MIR*, wykorzystali oni *Support*

*Vector Machine* (pl. maszyna wektorów nośnych). Wyniki uzyskane dla poszczególnych prób tylko nieznacznie różniły się od losowych. Za przyczynę niepowodzenia

uznano niewłaściwy dobór zestawu parametrów użytych do stworzenia modelu.

O wiele bardziej obiecujące wyniki uzyskali Yizhao Ni, Raúl Santos-Rodríguez, Matt Mcvicar i Tijl De Bie z kooperacji uniwersytetów University of Bristol i Universidad Carlos III de Madrid. W swoim artykule “Hit Song Science Once Again a Science?” z 2011-go roku[Ni+11] (już poprzez sam tytuł ustawiając

się w kontrapunkcie do poprzedników) przedstawiają wyniki swojego eksperymentu przeprowadzonego na zbiorze 5947 utworów z brytyjskiej listy przebojów z lat

1960–2010. Do ekstrakcji wartości parametrów muzycznych wykorzystali oni narzędzie EchoNest. Jako model wykorzystali oni prosty ruchomy perceptron (ang. *Shifting perceptron*) będący funkcją czasu. Wyniki eksperymentów tym razem różniły się znacząco na korzyść w porównaniu do odczytów losowych. Badaczom udało się też sformułować na ich podstawie szereg cennych wniosków.

Wychodząc z założenia, że na przestrzeni ostatniej dekady nastąpił ogromny postęp w dziedzinie data science, w szczególności w obszarze badań nad głębokimi sieciami neuronowymi, a także (co nie mniej istotne) dostępne są ogromne bazy zawierające wysokiej jakości, świetnie ustandaryzowane

i udokumentowane dane udostępnione przez platformy muzyczne takie jak Spotify, autor niniejszej pracy z dużym entuzjazmem postanowił podjąć się samodzielnej próby stworzenia wysokiej klasy modelu do predykcji popularności utworów muzycznych.

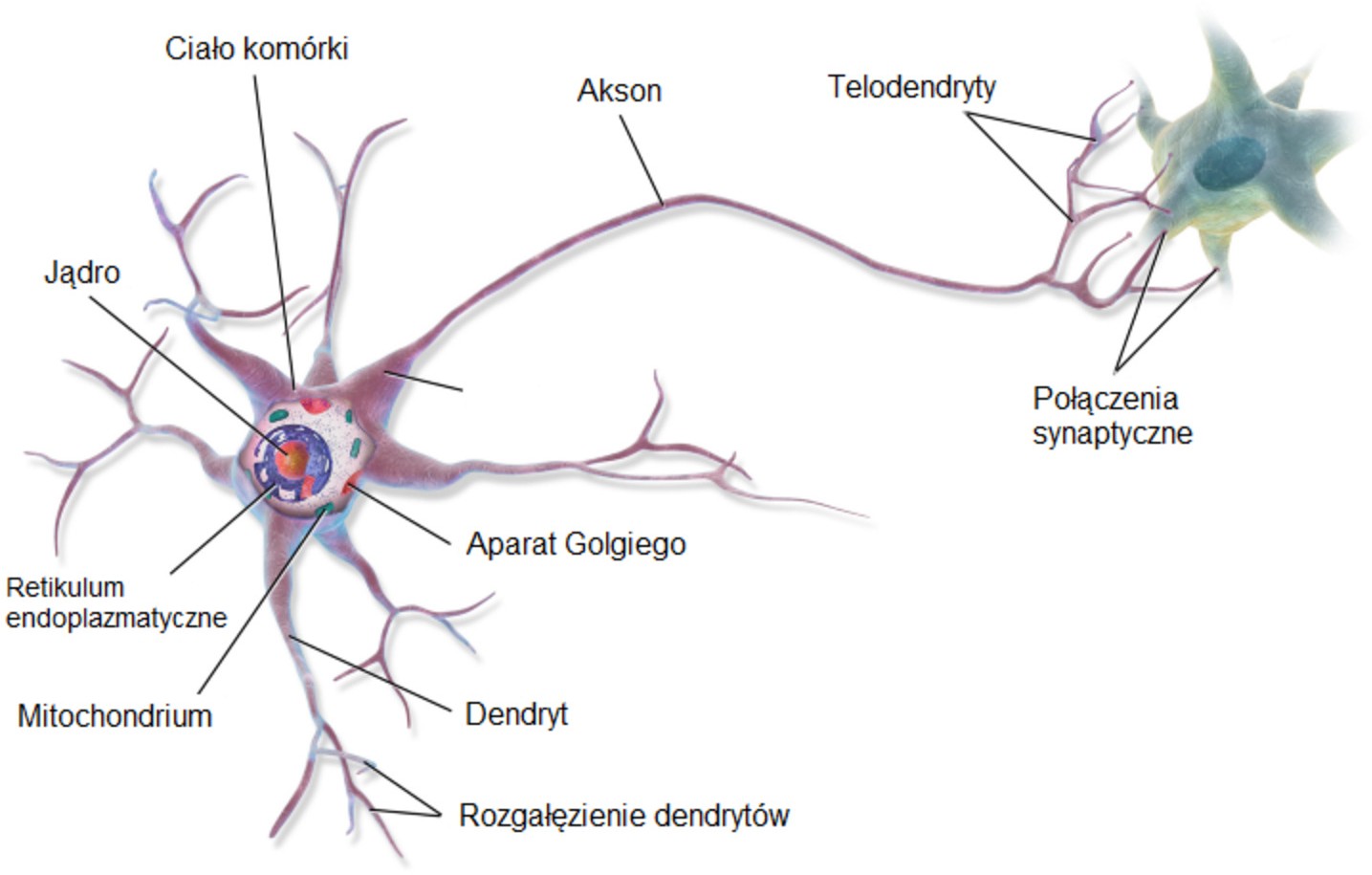
## Wykorzystanie sieci neuronowych do problemu regresji dla tabelarycznego zbioru danych

#### Sztuczne sieci neuronowe a biologia

Inspiracją do stworzenia sztucznych sieci neuronowych była budowa mózgu. Wraz z postępem badań w obszarze AI, sztuczne sieci neuronowe znacznie oddaliły się od swoich biologicznych odpowiedników[Gé18].

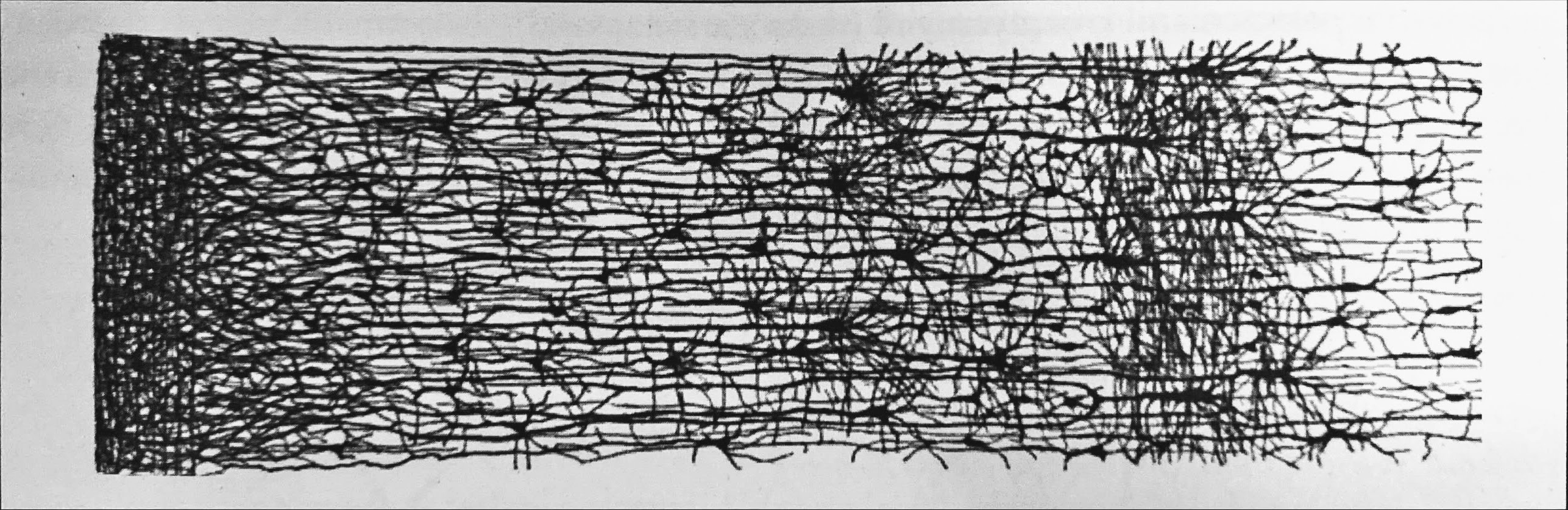
Na rysunku 3.1 przedstawiono budowę komórki nerwowej. Komórki te mieszczą się w zwierzęcej korze mózgowej. Ich główne elementy to: ciałko komórki, liczne rozgałęzione wypustki - dendryty, jedna bardzo długa wypustka

* akson. Rozgałęzienia na końcu aksonu to telodendryty, zakończone synapsami. Służą one do łączenia się z kolejną komórką. Komunikacja odbywa się za pośrednictwem krótkich impulsów elektrycznych. Po otrzymaniu dostatecznej liczby sygnałów, w przeciągu ułamku sekundy komórka zaczyna sama emitować i propagować dalej sygnały[Gé18].



Rysunek 3.1: Neuron biologiczny[Gé18].

Działanie pojedynczej komórki jest bardzo proste, natomiast zbudowana z miliardów takich komórek sieć jest zdolna do wykonywania bardzo skomplikowanych operacji. Sieć (mózg) ma strukturę warstwową, co przedstawiono na rysunku 3.2[Gé18].

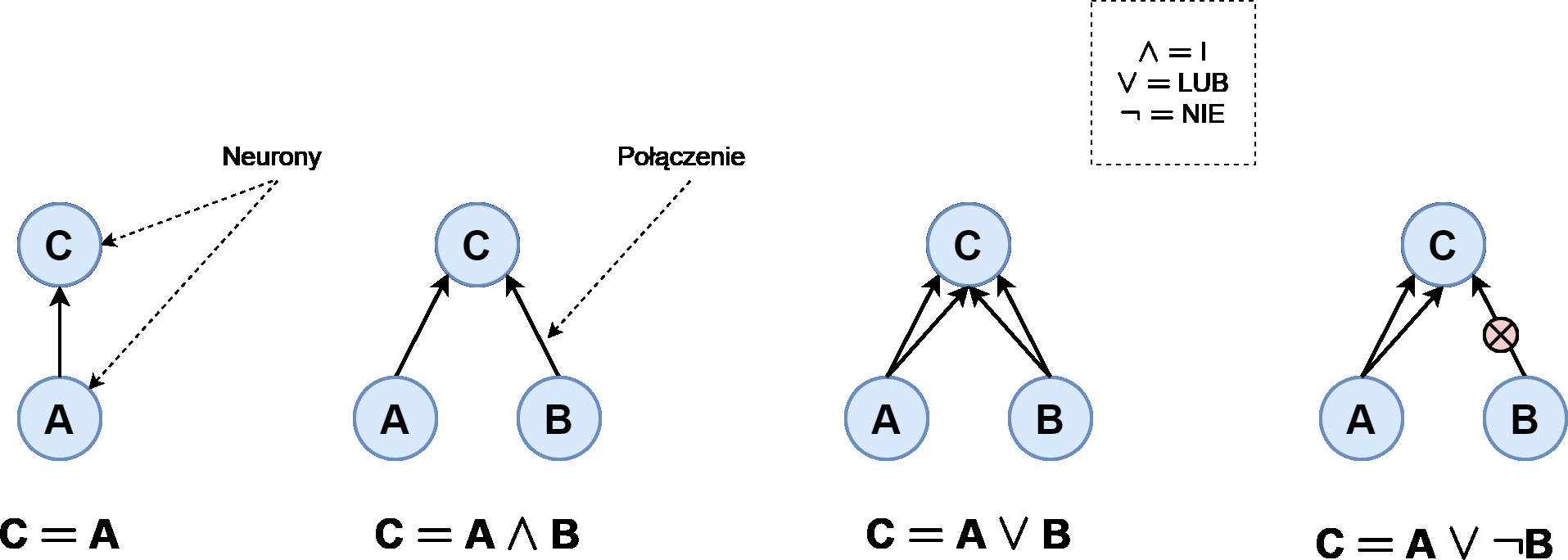


Rysunek 3.2: Wielowarstwowość kory mózgowej człowieka[Gé18].

Koncepcja sztucznych sieci neuronowych po raz pierwszy została zaprezentowana już w 1943 roku przez neurofizjologa Warrena McCullocha i matematyka Waltera Pittsa[MP43]. Był to uproszczony model działania sieci neuronów

w mózgach zwierzęcych, do przeprowadzania obliczeń przy użyciu rachunku zdań (ang. *propositional logic*)[Gé18].

Model neuronu zaproponowany przez McCullocha i Pittsa, znany jako ‘sztuczny neuron’ (ang. *artificial neuron*), ma conajmniej jedno wejście i jedno wyjście binarne (stan 0 lub 1). Wyjście aktywuje się wyłącznie przy określonej kombinacji sygnałów na wejściu. Udowodniono, że jest to wystarczające do stworzenia sieci rozwiązującej dowolne zadanie logiczne[Gé18]. Na rysunku 3.3 przedstawiono realizację prostych operacji logicznych z wykorzystaniem sztucznych neuronów.

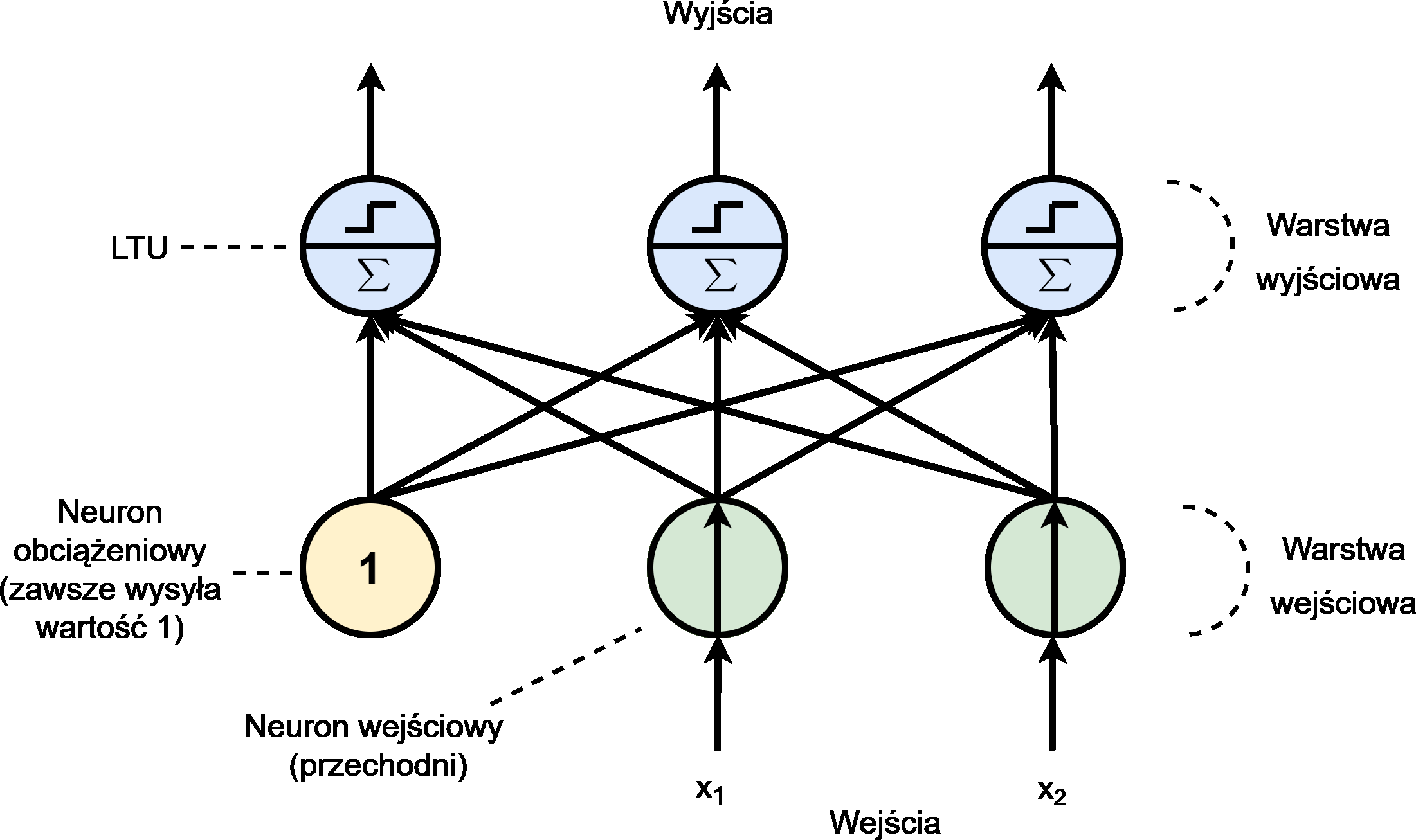


Rysunek 3.3: Realizacja prostych operacji logicznych za pomocą sztucznych neuronów[Gé18].

#### Perceptron

Perceptron był jedną z pierwszych implementacji sztucznej sieci neuronowej.

Został zaprezentowany w 1957 roku przez Franka Rosenblatta. Składa się z warstwy wejściowej i wyjściowej. Warstwa wejściowa zbudowana jest z tzw. neuronów przechodnich - propagujących dalej dostarczone do nich sygnały wejściowe, oraz z neuronu obciążeniowego - emitującego sygnał o wartości 1. Warstwa wyjściowa zbudowania jest z jednostek LTU. Każdy neuron warstwy wejściowej połączony jest ze wszystkimi neuronami wyjściowymi. Perceptron jest klasyfikatorem, pozwalającym na przypisywanie próbek do klas binarnych (liczba klas jest równa liczbie jednostek LTU w warstwie wyjściowej)[Gé18].



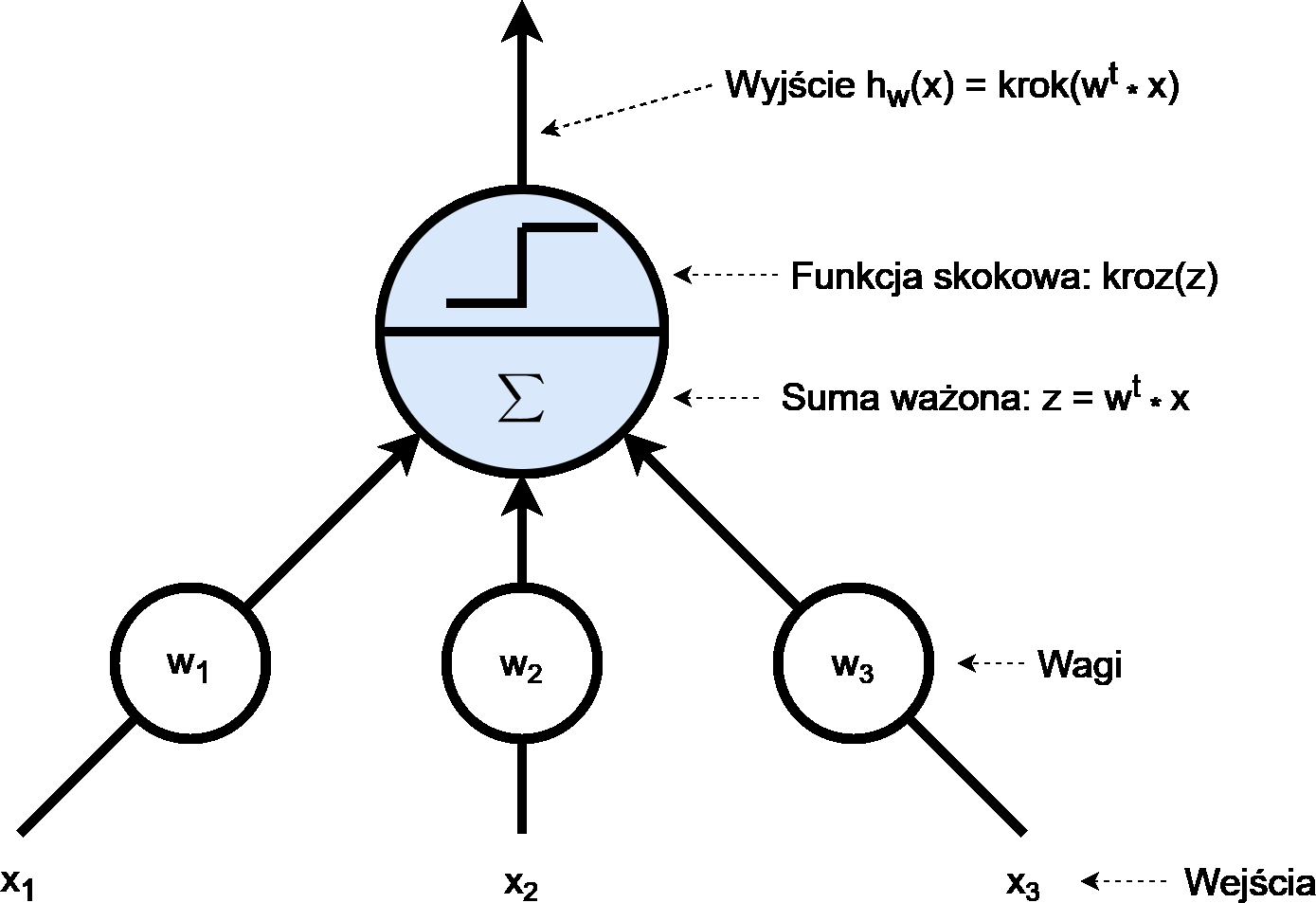
Rysunek 3.4: Diagram perceptronu z dwoma wejściami, obciążeniem i trzema wyjściami. Pozwala na klasyfikację danych do trzech klas binarnych[Gé18].

LTU - jednostka liniowa z progiem (ang. *linear threshold unit*), zaprezentowana na rysunku 3.5, jest modyfikacją sztucznego neuronu zaproponowanego przez McCullocha i Pittsa. Sygnałami wejściowymi są liczby (a nie stany binarne),

a każde połączenie ma przypisaną wagę. Jednostka wylicza sumę ważoną dla sygnałów wejścia ( = 11+22+ + = 𝑇 ), a wynik przekazuje do funkcji skokowej (ℎ() = () = (𝑇 )). Najczęściej stosuje się funkcje Heaviside’a, rzadziej funkcję sigma. Przekroczenie progu (1) oznacza klasyfikację pozytywną, wartość poniżej (0 lub -1) - negatywną[Gé18].

·

· · · ·



Rysunek 3.5: Jednostka liniowa z progiem[Gé18].

Najpopularniejsze funkcje skokowe wykorzystywane w perceptronach: Funkcja Heaviside’a:

ℎ𝑒𝑎𝑑𝑒() = ⎧⎨0 jeśli < 0

⎩1 jeśli ≥ 0

(1)

Funkcja signum:

𝑔()=

−1 jeśli < 0

0 jeśli = 0

⎧⎪⎨

⎪⎩1 jeśli > 0

(2)

Marvin Minsky i Seymour Papert w pracy “Perceptrons” z 1969 roku wskazali wiele poważnych wad perceptronu, takich jak np. brak możliwości rozwiązywania prostych problemów logicznych (alternatywa rozłączna - XOR). Rezultatem było spowolnienie dalszych prac nad sztucznymi sieciami neuronowymi. W przyszłości okaże się, że wspomniane ograniczenia można pokonać stosując modyfikację - perceptron wielowarstwowy[Gé18].

#### Uczenie wielowarstwowych sieci przy użyciu propagacji wstecznej

W 1985 roku D.E. Rumelhart i jego zespół opublikowali artykuł[RHW85],

w którym przedstawili koncepcję algorytmu propagacji wstecznej (ang. *backpropagation*). Łączy ona operację odwrotnego różniczkowania i metodę gradientu prostego.

W pierwszej fazie algorytm pobiera przykład uczący i przechodzi przez

całą sieć od wejścia do wyjścia (przebieg do przodu), dla każdej pary wejścia-wyjścia (⃗ , ) oblicza i zapamiętuje wyniki , 𝑎 i dla każdego węzła w warstwie

𝑑 𝑑

.

̂︁𝑑

Faza druga to przebieg odwrotny. Dla każdej pary wejścia-wyjścia (⃗𝑑, 𝑑)

obliczane są i zapamiętywane wyniki 𝜕𝐸𝑑 dla każdej wagi połączenia

𝜕

węzła w warstwie − 1 z z węzłem w warstwie :

1. zostaje zmierzony błąd 𝛿 na wyjściu z sieci z wykorzystaniem wzoru 4

1

1. błąd propagowany jest wstecznie do kolejnych warstw ukrytych 𝛿 z

wykorzystaniem wzoru 5

1. wyliczane są pochodne cząstkowe funkcji błędu 𝐸𝑑 po wadze połączenia

między neuronami z wykorzystaniem wzoru 6

W fazie trzeciej gradienty cząstkowe 𝜕𝐸𝑑 są sumowane dla każdej pary

𝜕

wejścia-wyjścia, tak by uzyskać gradient całkowity 𝜕𝐸(𝑋,𝜃) dla całego zbioru

𝜕

par wejścia-wyjścia 𝑋 = (⃗1, 1), . . . , (⃗𝑁 , 𝑁 ) z wykorzystaniem równania 6 (jest to po prostu średnia gradientów cząstkowych).

{ }

Ostatni krok to aktualizacja wag połączeń proporcjonalnie do przyjętego współczynnika uczenia 𝛼, odwrotnie do kierunku gradientu całkowitego 𝜕𝐸(𝑋,𝜃)

𝜕

z wykorzystaniem równania 7 [Bri].

Wzory:

Pochodna częściowa:

𝜕𝐸𝑑

= 𝛿*−*1

(3)

𝜕

Wartości błędu dla warstwy wyjściowej:

𝛿 = 𝑔*′* (𝑎)(𝑑 − 𝑑) (4)

1 0 1 ̂︁

Wartości błędu dla warstw ukrytych:

+1

𝛿 = 𝑔*′*(𝑎) ∑︁ +1𝛿+1

=1

(5)

Suma pochodnych częściowych dla każdej pary wejście-wyjście:

𝜕𝐸(𝑋, 𝜃) = 1 ∑︁

𝑁

𝜕 ( 1 ( − )2) = 1

∑︁ 𝜕𝐸𝑑

(6)

𝜕

𝑁 𝑑=1

𝜕 2 𝑑 𝑑

𝑁 𝑑=1

𝑁

𝜕

Aktualizacja wag dla połączeń:

= 𝛼 𝜕𝐸(𝑋, 𝜃)

̂︁

Δ

−

𝜕

(7)

## Algorytmy Genetyczne

#### Algorytm genetyczny a teoria ewolucji

Algorytm genetyczny powstał głównie jako odpowiedź na wyzwania obliczeniowe, do których trudno było podejść, korzystając z tradycyjnych narzędzi matematycznych. Czerpie on garściami inspiracje z teorii ewolucji i ogólnej wiedzy o genetyce.

Już w 1948 Alan Turing zaproponował koncepcję ‘ewolucyjnego poszukiwania’ (ang. *evolutionary search*). Od lat 60-tych XX wieku podobne prace prowadzone były już w wielu ośrodkach[ES15].

Teoria ewolucji Darwina wyjaśnia mechanizmy leżące u podstaw biologicznej różnorodności. To właśnie Darwin, badając fenotypy (cechy zewnętrzne) fauny

i flory na wyspie Galapagos, dostrzegł, że gatunki ewoluują, to jest, dostosowują się do otoczenia, a głównej mierze opiera się to na selekcji naturalnej, czyli zdolności przetrwania w danym środowisku i znalezieniu partnera do prokreacji. Selekcja naturalna jest brutalna - wygrywa przeważnie najlepiej dopasowany. Innym ważnym czynnikiem odkrytym przez Darwina (oczywiście jedynie pośrednio, poprzez obserwacje różnic w fenotypie) są mutacje, czyli zmiany genetyczne przekładające się na różnice w fenotypie, wprowadzające element

losowości. Ewolucja nie jest równocześnie jednokierunkowym procesem optymalizacyjnym prowadzącym do ‘globalnego optimum’. W środowisku naturalnym oddziałuje

na siebie ogromna liczba czynników i może się zdarzyć, że najlepiej dopasowany osobnik zginie albo, że globalny, wysoki wskaźnik dopasowania dla całej populacji stanie się powodem jej upadku (na przykład poprzez wyparcie z ekosystemu gatunku zależnego, położonego niżej w łańcuchu pokarmowym)[ES15].





Rysunek 4.1: Darwin pracując na Galapagos prowadził między innymi badania nad różnicami w fenotypie zięb. Autor ilustracji: John Gould (‘Voyage of the Beagle’).

Za ojca genetyki uważany jest Gregor Johann Mendel - czeski zakonnik niemieckiego pochodzenia, działający w drugiej połowie XIX wieku. To on, prowadząc eksperymenty z krzyżowaniem roślin, odkrył zjawisko cech dominujących i recesywnych (w dzisiejszym spojrzeniu alleli dominujących i regresywnych).

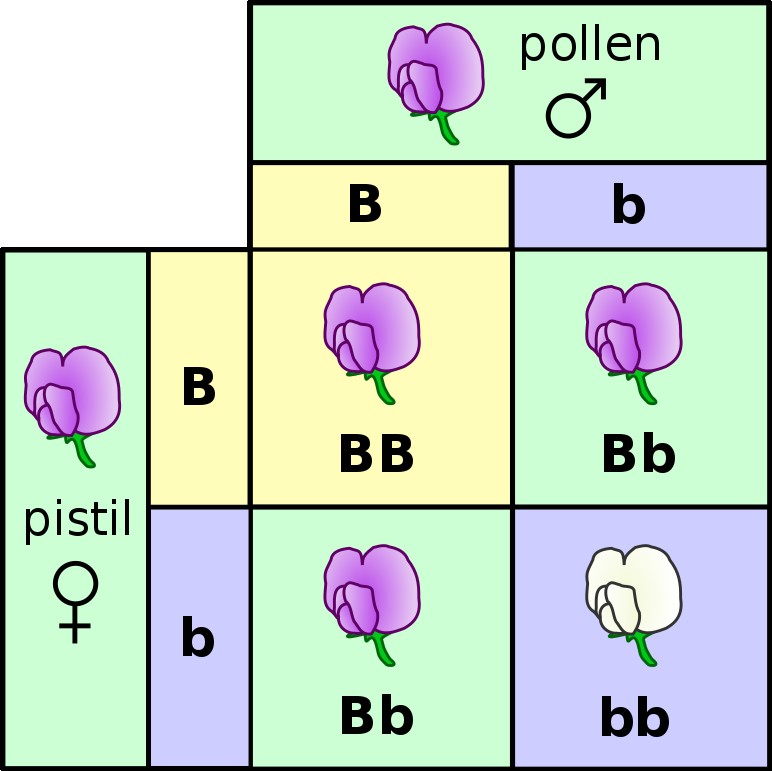
Podstawową jednostką przechowującą informację genetyczną jest gen. Konkretna wartość genu to allel. Jeden gen może mieć wpływ na wiele cech fenotypowych, jak i jedna cecha fenotypowa może być zdefiniowana przez wiele genów.

Zmiany w fenotypie są zawsze uzależnione od zmian genetycznych, które z kolei są wynikiem mutacji i rekombinacji (krzyżówki materiału genetycznego rodziców). Genom jest kompletną informacją genetyczną osobnika. Genom jest przechowywany w zestawie chromosomów (materiał genetyczny człowieka jest zapisany w 46 chromosomach). Istoty żywe, takie jak zwierzęta i bakterie, przechowują podwójną kopię informacji genetycznej w większości komórek - nazywanych diploidami. Gamety, czyli komórki rozrodcze, zawierają jedynie pojedynczy zestaw chromosomów. Połączenie gamety męskiej z żeńską prowadzi do powstania zygoty. Proces ten określa się terminem ontogenezy. Podczas

ontogenezy materiał genetyczny nie jest zmieniany, dlatego nie można przyrównywać tego procesu do krzyżówki znanej z algorytmu genetycznego. Podobny do krzyżówki, jest z kolei proces formowania się gamet - mejoza - szczególny rodzaj podziału komórkowego, który gwarantuje, że w wynikowym produkcie

* gamecie - znajdzie się tylko jedna kopia każdego z chromosomów. Na jednym z etapów mejozy ma również miejsce rekombinacja, a mówiąc precyzyjniej - krzyżówka w losowym punkcie przecięcia. Każda z czterech gamet powstałych w wyniku mejozy ma inną informacje genetyczną od oryginalnego genomu

męskiego i żeńskiego, co potem przekłada się na różnice genetyczne u potomków[ES15].



Rysunek 4.2: Szachownica Punnetta ilustruje dziedziczenie w przypadku dominacji zupełnej cechy jednogenowej. Autor: Madprime (Wikipedia Community).

Niezmiernie ważne jest, by uświadomić sobie, że wszystkie zmiany (to jest mutacje i rekombinacje) zachodzą na poziomie genetycznym, natomiast selekcja naturalna odbywa się na podstawie fenotypu[ES15].

Komputerowe algorytmy genetyczne nie są odwzorowaniem ‘jeden do jednego’ procesów biologicznych, a jedynie się nimi inspirują. Terminologia używana w obu tych domenach jest często podobna, ale znaczenia poszczególnych terminów mogą być całkowicie odmienne. Zasadniczo, najważniejszym wspólnym mianownikiem jest koncepcja selekcji naturalnej, warunkująca dobieranie osobników do rekombinacji i wyznaczanie ocalałych (ang. *survivors*).

#### Cykl życia populacji

Algorytm genetyczny jest z założenia generyczny - trzeba go traktować jako szablon, w którym, w zależności od potrzeb można wymieniać komponenty składowe. Te komponenty to:

* + - inicjacja populacji
    - ewaluacja osobników
    - selekcja rodziców
    - rekombinacja/krzyżowanie
    - mutacja
    - wybór ocalałych
    - warunek końcowy

Wspólnym mianownikiem wszystkich algorytmów genetycznych jest populacja osobników (potencjalnych rozwiązań) funkcjonująca w środowisku o ograniczonych zasobach (ograniczenia dla poszczególnych parametrów), gdzie rządzi prawo selekcji naturalnej - przeżywają osobniki najlepiej dopasowane (z najlepszą wartością metryki dopasowania)[ES15]. W przypadku prawidłowo działającego algorytmu zaobserwować można ciągłą poprawę średniej wartości metryki dopasowania w ujęciu pokoleniowym. W głównej mierze do polepszania wyników przyczynia się mechanizm selekcji. Mutacja i rekombinacja (krzyżówka) wprowadzają z kolei element losowości, zapewniając lepsze pokrycie przestrzeni potencjalnych rozwiązań.

Poniżej przedstawiono pseudokod dla algorytmu genetycznego oraz

diagram ilustrujący cykl życia populacji.

**Algorithm 1** Generyczny szablon dla algorytmu genetycznego[ES15].

INICJACJA POPULACJI LOSOWYMI OSOBNIKAMI EWALUACJA OSOBNIKÓW

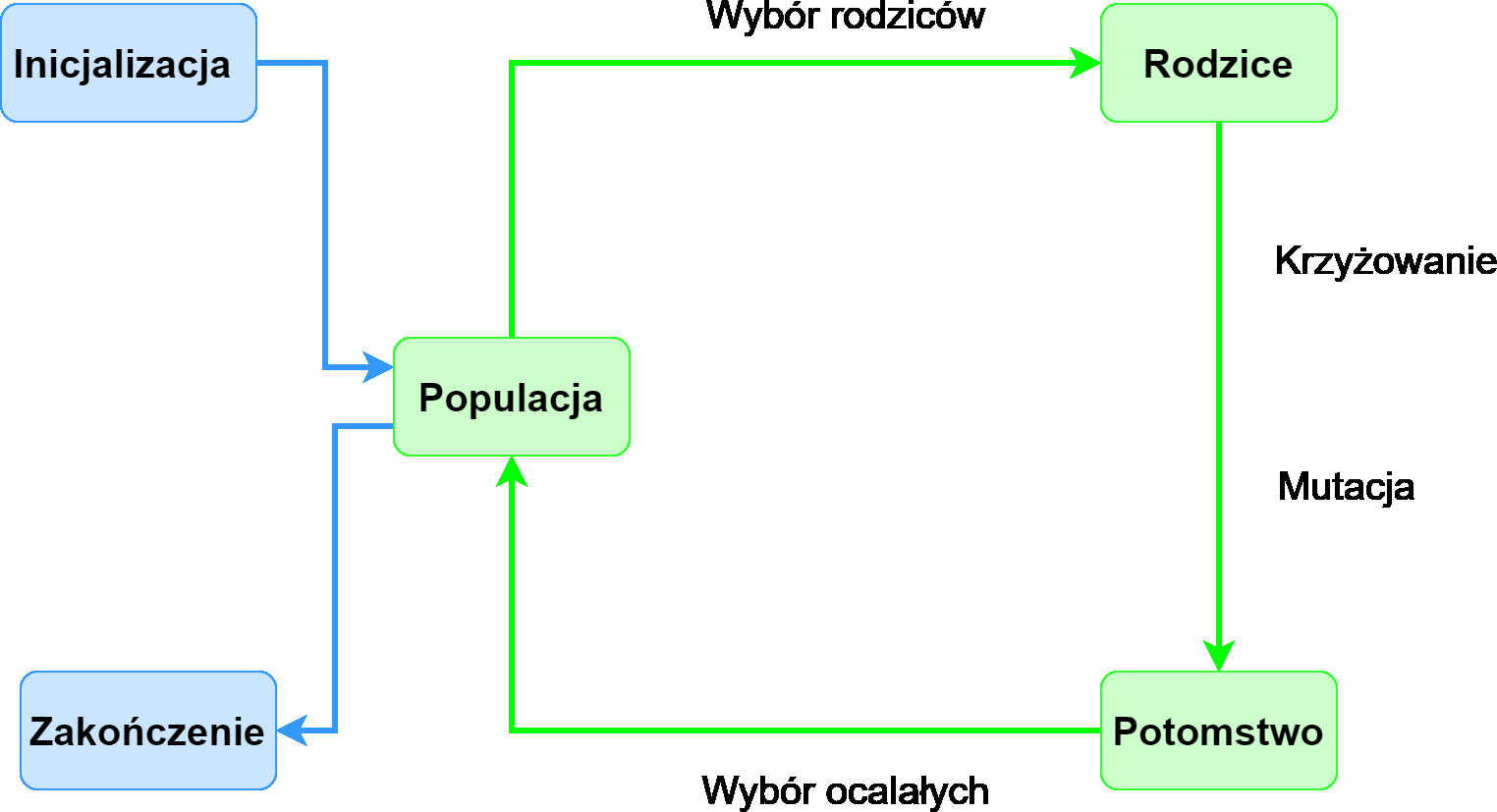
**while** NIE SPEŁNIENIE WARUNKU KOŃCOWEGO **do**

WYBÓR RODZICÓW KRZYŻOWANIE RODZICÓW MUTACJA POTOMKÓW EWALUACJA POTOMSTWA

WYBÓR OSOBNIKÓW DO NASTĘPNEGO POKOLENIA

##### end while

**return** NAJLEPSZY OSOBNIK



Rysunek 4.3: Cykl życia populacji w algorytmie genetycznym[ES15].

#### Inicjacja populacji

W większości implementacji inicjacja populacji odbywa się w możliwie prosty sposób: tworzy się osobniki o w pełni losowych wartościach parametrów, gdzie jedynym ograniczeniem są wyznaczone arbitralnie (w oparciu o doświadczenie i literaturę przedmiotu) minimalne i maksymalne wartość dla każdego z parametrów. W specyficznych przypadkach stosuje się heurystyki w celu umieszczenia w zerowym pokoleniu rozwiązań lepszych od losowych. Trzeba

mieć jednak w pamięci, że wiąże się to z dodatkowym kosztem obliczeniowym i nie zawsze przyczynia się do poprawy finalnych wyników[ES15].

#### Ewaluacja osobników

Funkcja ewaluacyjna (funkcja dopasowania) reprezentuje wymagania, do których powinna się adaptować populacja. Jest bazą dla mechanizmu selekcji i tym samym przyczynia się do ogólnej poprawy wyników osobników. Mówiąc precyzyjniej - definiuje ona co oznacza poprawa w kontekście poszukiwanego rozwiązania dla zadanego problemu. Jej wartość jest metryką dopasowania

osobnika[ES15]. Dobrymi funkcjami ewaluacyjnymi w przypadku sieci neuronowych są: RMSE (ang. *root mean square error* ), MSE (ang. *mean square error*)

lub MAE (ang. *mean absolute error* ) wyliczane dla zbioru walidacyjnego. Ze względu na to, że w ortodoksyjnym podejściu dąży się do maksymalizacji

funkcji dopasowania, często tam gdzie jest to konieczne stosuje się odwrotność oryginalnej wartości funkcji[ES15].

#### Wybór rodziców

Selekcja rodziców (ang. *parents*) ma na celu dobór najodpowiedniejszych osobników do rekombinacji (krzyżówki). Tak samo jak wybór ocalałych (ang. *survivors*) wymusza ogólną poprawę jakości populacji. W algorytmach genetycznych selekcja rodziców oparta jest najczęściej na prawdopodobieństwie, gdzie osobniki

z wysoką wartością metryki dopasowania mają największe szanse na wybór. Niemniej, osobniki gorsze również mają niewielkie szanse na bycie wyznaczonym. Powodem jest chęć zmniejszenia ‘zachłanności’ algorytmu (ang. *greedy algorithm*) i tym samym minimalizacja ryzyka wpadnięcia w optimum lokalne[ES15].

#### Krzyżówki

Krzyżówka jest operacją pozwalającą na rekombinacje materiału genetycznego dwóch lub więcej osobników (rodziców, ang. *parents*) w celu powołania do życia potomstwa (ang. *offsprings*), które odziedziczy unikalną kombinację ich cech. Główną motywacją operacji krzyżowania jest chęć uzyskania przynajmniej jednego potomka, który osiągnie lepszą wartość metryki dopasowania od swoich rodziców[Kra17].

W praktyce stosowane są różne rodzaje rekombinacji: warianty z dwoma (najczęściej) lub z większą liczbą rodziców, krzyżówki jedno i wielopunktowe, dolosowywanie kolejnych cech od poszczególnych rodziców z określonym współczynnikiem prawdopodobieństwa. Trzeba pamiętać, że zastosowanie każdego z wymienionych wariantów niesie za sobą określone konsekwencje. Gdy pewne cechy są ze sobą

powiązane warto rozważyć wariant z punktem przecięcia. W przypadku gdy konieczne jest zagwarantowanie określonych proporcji udziału rodziców w wynikowym materiale genetycznym potomka, zasadne jest rozważanie wariantu opartego na dolosowywaniu kolejnych genów (parametrów) z pewnym współczynnikiem prawdopodobieństwa[ES15].

#### Mutacje

Mutacja ma na celu dostarczenie do populacji mutantów - osobników, których materiał genetyczny nie jest wyłącznie kombinacją genów rodziców. Jest to swoisty dopływ ‘świeżej krwi’, gwarantujący pełniejsze przeszukanie przestrzeni rozwiązań. Operator mutacji jest z założenia stochastyczny (losowy).

Należy wystrzegać się przy jego implementacji jakichkolwiek elementów systemowych

czy obciążeń (ang. *bias*). Czysto teoretycznie prawidłowo zaimplementowana operacja mutacji gwarantuje osiągnięcie optimum globalnego w skończonym czasie[ES15].

#### Wybór potomstwa

Wybór potomstwa lub wyrażając się bardziej precyzyjnie wybór osobników do następnego pokolenia, w przeciwieństwie do wyboru rodziców jest operacją w pełni deterministyczną. Najczęściej przy wyborze brane pod uwagę są dwa kryteria: wartość metryki dopasowania lub wiek osobnika. W pierwszym przypadku z populacji usuwa się najgorsze osobniki, w drugim te, które są w populacji najdłużej, a w przypadkach spornych można odwołać się do wartości metryki dopasowania. W zależności od ‘dynamiki’ algorytmu, rozumianej jako stosunku generowanych potomków do wielkości populacji, operacja może mieć bardziej charakter wymiany - stabilna populacja, mała

ilość potomstwa w pokoleniu, lub mocnej selekcji - bardzo duża liczba potomstwa

i konieczność ograniczenia się do wyboru jedynie najlepszych nowych osobników[ES15].

#### Warunek końcowy

Warunek końcowy, kończy cykl życia populacji i tym samym cały algorytm.

Wyróżnia się dwa główne przypadki[ES15]:

* + - znany jest optymalny poziom dopasowania
    - docelowy poziom dopasowania oparty jest na założeniach

W pierwszym przypadku, dla warunków idealnych, zakończenie działania algorytmu następuje w chwili osiągnięcia optimum globalnego. W praktyce, trzeba wziąć pod uwagę uproszczenia wprowadzone do modelu, ‘szumy’ w danych i inne czynniki wpływające na ostateczny wynik. Z tego też powodu, wyznacza się arbitralnie niewielką wartość 𝜖 > 0, która określa precyzję algorytmu[ES15].

Niestety algorytm genetyczny jest z natury stochastyczny (ogromną rolę odgrywa czynnik losowy) i nie ma żadnej gwarancji osiągnięcia (czy choćby zbliżenia się na akceptowalną odległość) do optimum globalnego. Z tego też powodu konieczne jest wprowadzenie dodatkowego warunku. Najczęściej stosowane ograniczenia to[ES15]:

* + - odgórnie przyjęty limit liczby pokoleń
    - stagnacja w poprawie wyniku w czasie cyklu życia kilku pokoleń
    - różnice w wynikach poszczególnych osobników w populacji (mierzone na przykład odchyleniem standardowym) spadną poniżej określonego progu

W przypadku, kiedy optimum globalne nie jest znane, do zakończenia wystarczający jest jeden z warunków z powyższej listy[ES15].

Poważnym zagrożeniem w przypadku algorytmów genetycznych jest zbieganie i ‘utknięcie’ w optimum lokalnym. Zwłaszcza w sytuacji kiedy optimum globalne nie jest znane, warto rozważyć dodanie do implementacji strategi restartu algorytmu[Kra17].

## Opis zbioru danych

Zbiór danych zawiera informacje o ponad 170 tysiącach utworów muzycznych, opublikowanych w latach 1921–2020 i dostępnych na szwedzkiej platformie streamingowej Spotify. Dataset został stworzony przy wykorzystaniu oficjalnego, publicznego API deweloperskiego Spotify i udostępniony na platformie Kaggle

w formacie CSV przez użytkownika Yamaç Eren Ay[Ay].

Oryginalny dataset to zbiór tabelaryczny zawierający 170653 rekordów podzielonych na 19 kolumn. Kolumny te to kolejno:

* *valence* (pl.ozdobnik) - Miara w skali od 0.0 do 1.0 określająca stopień ‘pozytywności’ przekazywany przez utwór. Utwory z wysoką wartością współczynnika *valence* brzmią bardziej pozytywnie (szczęśliwie, radośnie lub euforycznie), podczas gdy niska wartość tego współczynnika objawia się brzmieniem negatywnym (smutnym, depresyjnym, zdenerwowanym lub wściekłym).
* *year* (pl. rok) - Rok publikacji utwory. Wartości między 1921 a 2020 włącznie.
* *acousticness* (pl. akustyczność) - Miara w skali od 0.0 do 1.0 określająca pewność, z jaką dany utwór można zakwalifikować jako akustyczny. 1.0
  + wskazuje na wysoką pewność, 0.0 na bardzo niską.
* *artists* (pl. artyści) - Lista imion i nazwisk lub pseudonimów artystycznych artystów wykonujących dany utwór.
* *danceability* (pl. taneczność) - Miara w skali od 0.0 do 1.0 określająca, w jakim stopniu dany utwór jest odpowiedni do tańca, wyliczana jako kombinacja takich parametrów muzycznych ja tempo, stabilność rytmu, moc taktu i ogólna regularność. 1.0 oznacza wysoką taneczność, a 0.0
  + bardzo niską.
* *duration\_ms* (pl. czas trwania) - Czas trwania utworu w milisekundach.
* *energy* (pl. energia) - Miara w skali od 0.0 do 1.0 reprezentująca odczuwalny stopień intensywności i aktywności utworu. Typowe utwory ‘energetyczne’ są odbierane jako szybkie, głośne czy wręcz hałaśliwe. Przykładem wysokiej energetyczności mogą byś utwory death-metalowe, podczas gdy preludia Bacha będą cechowały się niską energetycznością. Postrzegalne czynniki wpływające na tę cechę to: dynamiczny zakres, odbierana głośność, tembr, *onset rate* (pl. współczynnik rozpoczęć ??) i ogólna entropia.
* *explicit* (pl. odważny/śmiały) - Wartość *true* (pl. prawda)/*false* (pl. fałsz). Flaga określająca czy dany utwór zawiera wulgaryzmy, treści erotyczne, treści nacechowane przemocą, wzmianki o nielegalny używkach i tym podobne; w ogólności treści skierowane do odbiorców pełnoletnich.
* *id* (pl. identyfikator) - Unikalne, alfanumeryczne Spotify ID dla utworu.
* *instrumentalness* (pl. instrumentalność) - Miara prawdopodobieństwa określająca, czy dany utwór nie zawiera wstawek wokalnych (fragmentów śpiewanych). Wyrazy dźwiękonaśladowcze takie jak ‘ooh’ czy ‘aah’ przez algorytm/klasyfikator są traktowane jako instrumentalne. Rap czy zwykła mowa traktowana jako ‘czysty wokal’. Im bliżej wartości 1.0, tym wyższe prawdopodobieństwo braku wokali. Wartości powyżej 0.5 w założeniu mają reprezentować utwory instrumentalne, natomiast prawdopodobieństwo prawidłowej klasyfikacji rośnie wraz ze zbliżaniem się wskaźnika do wartości 1.0.
* *key* (pl. tonacja). Tonacja utworu przedstawiona jako liczba całkowita nieujemna: 0,1,2. . . . Liczby zmapowane są na standardową notację muzyczną: 0 = C, 1 = G, 2 = D i tak dalej.
* *liveness* - (pl. żywość/żywiołowość) - Miara żywiołowości określa prawdopodobieństwo obecność publiczności podczas nagrania. Wartości *liveness* powyżej 0.8

wskazują na wysokie prawdopodobieństwo, że utwór wykonywany był ‘na żywo’.

* *loudness* (pl. głośność) - Ogólna głośność utworu mierzona w decybelach. Wartości głośności są wyliczane poprzez uśrednienie głośności całego utworu, co sprawia, że metryka staje się użyteczna do porównywania utworów. Głośność jest cechą dźwięku ściśle skorelowaną z amplitudą fali dźwiękowej. Wartości w większości przypadków wahają się między

-60 a 0 db.

* *mode* (pl. tryb/dominanta) - Parametr ’mode’ oznacza modalność utworu: *major* (pl. główna, pierwszorzędna) lub *minor* (pl. pomniejsza, drugorzędna), w znaczeniu typu skali, w której znajduje się warstwa melodyczna. 1 oznacza *major*, 0 to *minor*.
* *name* (pl. nazwa) - Tytuł piosenki, utworu lub nagrania.
* *popularity* (pl. popularność) - Popularność utworu w skali od 0 do 100, gdzie 100 oznacza najwyższą popularność. Popularność wyliczana jest przez algorytm, bazujący przede wszystkim na liczbie odtworzeń

utworu, ale także na tym jak świeże (odległe od chwili obecnej) są te odtworzenia.

* *release\_date* (pl. data publikacji) - Data publikacji utworu lub albumu, na którym znalazł się utwór.
* *speechiness* (pl. wypełnienie mową) - Parametr *speechiness* określa stopień obecności mowy w utworze. Nagrania wypełnione mową takie jak programy ‘talk show’, audiobooki czy poematy będą uzyskiwały wartości tego parametru zbliżone do 1.0. Wartości powyżej 0.66 wskazują na wysokie prawdopodobieństwo wypełnienia nagrania w całości mową. Wartości

od 0.33 do 0.66 wskazują na występowanie zarówno mowy jak i muzyki, zarówno w następujących po sobie sekwencjach, jak i w nakładających się na siebie warstwach dźwiękowych. Wartości poniżej 0.33 oznaczają duże prawdopodobieństwo nie występowania mowy w utworze.

* *tempo* (pl. tempo) - Ogólne, szacowane tempo utworu w taktach na minutę (ang. *BPM* ). W terminologi muzycznej, tempo jest szybkością danego fragmentu/kawałka utworu wyliczaną bezpośrednio w oparciu o średnią długość taktu.

Opisy parametrów są luźnym tłumaczeniem oficjalnej dokumentacji API deweloperskiego Spotify[Tea]. **Metody wyznaczania wartości metryk subiektywnych są własnością intelektualną firmy Spotify i nie są dostępne publicznie.**

valence

4000

3500

3000

liczba utworów

2500

2000

1500

1000

500

0

0.0

0.2

0.4

wartośś

0.6

0.8

1.0

Rysunek 5.1: Histogram dla cechy *valence* (pl. ozdobnik).

acousticness

17500

15000

12500

liczba utworów

10000

7500

5000

2500

0

0.0

0.2 0.4 0.6

wartośś

0.8

1.0

Rysunek 5.2: Histogram dla cechy *acousticness* (pl. akustyczność).

danceability

3500

3000

2500

liczba utworów

2000

1500

1000

500

0

0.0 0.2 0.4 0.6 0.8 1.0

wartośś

Rysunek 5.3: Histogram dla cechy *danceability* (pl. taneczność).

duration\_ms

10000

8000

liczba utworów

6000

4000

2000

0

104 105 106

milisekundy

Rysunek 5.4: Histogram dla cechy *duration\_ms* (pl. czas trwania w milisekundach).

energy

2000

1500

liczba utworów

1000

500

0

0.0 0.2 0.4 0.6 0.8 1.0

wartośś

Rysunek 5.5: Histogram dla cechy *energy* (pl. energia/energetyczność).

14000

12000

10000

liczba utworów

8000

6000

4000

2000

0

instrumentalness

10−6 10−5 10−4 10−3 10−2 10−1 100

wartośś

Rysunek 5.6: Histogram dla cechy *instrumentalness* (pl. instrumentalność).

liveness

14000

12000

10000

liczba utworów

8000

6000

4000

2000

0

0.0

0.2

0.4 0.6 0.8 1.0

wartośś

Rysunek 5.7: Histogram dla cechy *liveness* (pl. żywość/żywiołowość).

loudness

8000

7000

6000

liczba utworów

5000

4000

3000

2000

1000

0

−60 −50 −40 −30

dB

−20

−10 0

Rysunek 5.8: Histogram dla cechy *loudness* (pl. głośność).

year

2000

1750

1500

liczba utworów

1250

1000

750

500

250

0

1920 1940 1960 1980 2000 2020

rok publikacji

Rysunek 5.9: Histogram dla cechy *year* (pl. rok publikacji).

speechiness

8000

6000

liczba utworów

4000

2000

0 10−2 10−1 100

wartośś

Rysunek 5.10: Histogram dla cechy *speechiness* (pl. wypełnienie mową).

6000

tempo

5000

4000

lic ba utworów

3000

2000

1000

0

0 50 100 150 200 250

BPM (beats per minute pl. uder enia na minutę)

Rysunek 5.11: Histogram dla cechy *tempo* (pl. tempo).

popularity

25000

20000

liczba utworów

15000

10000

5000

0

0 20 40 60 80 100

wartośś

Rysunek 5.12: Histogram dla cechy *popularity* (pl. popularność).

mode

minor

70.7%

29.3%

major

Rysunek 5.13: Wykres kołowy pokazujący procentowy rozkład dominant dla badanych utworów.

explicit

nie

91.5%

8.5%

tak

Rysunek 5.14: Wykres kołowy pokazujący procentowy udział utworów zawierających treści przeznaczone dla odbiorców pełnoletnich.

key

D G

A 11.0%

12.2%

C

10.3%

12.7%

9.6%

E

4.3%

Eis

5.1%

7.6%

H

7.6%

7.1%

6.3%

6.3%

Ais

Dis

Fis

Cis

Gis

Rysunek 5.15: Wykres kołowy pokazujący procentowy rozkład tonacji dla badanych utworów.

## Zastosowanie dwustopniowego algorytmu uczenia maszynowego do predykcji popularności utworu

#### Selekcja i wstępna obróbka danych

Do stworzenia modeli wykorzystano następujące parametry: *valence*, *acousticness*, *danceability*, *duration\_ms*, *energy*, *explicit*, *instrumentalness*, *key*, *liveness*, *loudness*, *mode*, *release\_date*, *speechiness* i *tempo*. Na potrzeby dalszych

działań wartości parametru *key* zostały zamienione na binarną postać wektorową (ang. *one-hot encoding*), a wartości *release\_date* na milisekundy w standardzie UNIX. Na podstawie wartości wcześniej wymienionych parametrów modele szacowały wartość popularności (ang. *popularity*) utworu.

#### Ogólny zarys architektury zastosowanej sieci neuronowej

Przed ostatecznym wyborem modelu bazowego, do późniejszego wyliczenia

przez algorytm genetyczny, dokonano szeregu prób, eksperymentując z zastosowaniem różnych optymalizatorów, regularyzatorów, inicjatorów wag i funkcji aktywacji,

a także z ilością warstw ukrytych i liczbą neuronów w każdej z nich. Ostatecznie zdecydowano się na następującą bazę:

* + - warstwa wejściowa z 25 neuronami i normalizacją wsadową (ang. *Batch Normalization*) (𝑒 = 0.99)
    - trzy warstwy ukryte posiadające między 10 a 200 neuronów; wagi i obciążenia inicjowane funkcją He z rozkładem normalnym; normalizacja wsadowa (𝑒 = 0.99); aktywacja funkcją ELU; regularyzacja

przez porzucenie (ang. *Dropout*), ze współczynnikiem porzucania wynoszącym między 0.01 a 0.5

* + - warstwa wyjściowa z 1 neuronem i aktywacją liniową
    - optymalizator Adam (𝜂 = 0.001)
    - jako funkcję straty (ang. *loss*) do minimalizacji wybrano MSE (ang.

*mean square error*)

* + - jako metryki pomocnicze dodano RMSE (ang. *root mean squared error* ) i MAE (ang. *mean absolute error* )
    - liczbę próbek w minigrupie (ang. *batch*) z przedziału 100–400

Decyzja o zastosowaniu tylko trzech warstw ukrytych podyktowana była specyfiką działania algorytmu genetycznego oraz ograniczeniami środowiska wykonawczego (maksymalny czas trwania sesji w Google Colab Pro Plus to 24 godziny), i wynikającą z powyższych koniecznością skrócenia średniego czasu uczenia pojedynczego modelu.

#### Inicjacja wag funkcją HE

Podczas badań nad procesem uczenia głębokich sieci neuronowych naukowcy natrafili na tak zwany ‘problem zanikających gradientów’ (malenie wartości gradientów funkcji kosztu wraz z propagacją wsteczną na coraz niższe warstwy sieci) oraz będący jego przeciwieństwem ‘problem wybuchających gradientów’ (stały wzrost gradientów powodujący rozbieżność algorytmu (problem dotyczący głównie rekurencyjnych sieci neuronowych)[Gé18]. Była to przeszkoda na tyle poważna, że na długie lata wstrzymała rozwój prac nad głębokimi sieciami neuronowymi. Dopiero w 2010 roku Xavierowi Glorot i Yoshue Benglo udało się zdiagnozować przyczynę problemu i zaproponować skuteczne rozwiązanie.

Przyczyną okazało się powszechne wykorzystywanie kombinacji logistycznej i sigmoidalnej funkcji aktywacji, oraz inicjacji wag losowymi wartościami o rozkładzie normalnym i średniej wynoszącej 0, oraz odchyleniu standardowym równym 1. Badacze doszli do wniosku, że prawidłowy przepływ informacji między warstwami sieci jest możliwy jedynie, gdy wariancja dla wyjść warstwy jest równa wariancji wejść tejże, a dodatkowo gradienty funkcji kosztu muszą mieć taką samą wariancję przed i po przejściu przez daną warstwę w procesie propagacji wstecznej[GB10].

Wzory dla inicjacji Xaviera (Glorot), dla logistycznej funkcji aktywacji przedstawiają się następująco:

Odchylenie standardowe dla rozkładu normalnego o średniej 0:

2

+

√︃𝜎 =

(8)

Promień dla rozkładu jednorodnego (gdzie wartości mieszczą się w przedziale ⟨-r,r⟩):

√︃ =

Gdzie:

6

+

(9)

liczba połączeń wejściowych w warstwie

liczba połączeń wyjściowych w warstwie

Zaproponowana przez Glorot i Benglo metoda inicjacji wag sprawdzała się świetnie w połączeniu z logistyczną funkcją aktywacji. W przypadku zastosowania jej w połączeniu z funkcją aktywacji ReLU (a także jej odmianami takimi jak ELU) wyniki były niezadowalające. Kolejnego przełomu udało się dokonać w 2015 roku Kaimingowi He wraz z zespołem współpracowników z

Microsoft Research, w wyniku prac nad klasyfikatorem dla zbioru ImageNet[He+15]. Inicjacja He w odróżnieniu od inicjacji Xaviera (Glorot) uwzględnia jedynie obciążalność wejściową warstwy (a nie jak u poprzednika średnią obciążalności wejściowej i wyjściowej)[Gé18].

Wzór na odchylenie standardowe w inicjacji He dla rozkładu normalnego o średniej 0:

Gdzie:

𝜎 = 2

√︃

(10)

liczba połączeń wejściowych w warstwie

#### Funkcja aktywacji ELU

Początkowo uważano, że najlepsze jako funkcje aktywacji będą funkcje sigmoidalne. Przekonanie to brało się stąd, że podobnymi funkcjami można opisać przetwarzanie sygnału przez neurony biologiczne. To intuicyjne skojarzenie okazało się jednak mylne. Po pewnym czasie odkryto, że o wiele lepiej do tego celu nadaje się funkcja ReLU (ang. *Rectified Linear Unit* pl. prosta liniowa funkcja aktywacji)[Gé18]. Głównymi zaletami funkcji ReLU są:

* + - nie uleganie nasyceniu dla wartości dodatnich
    - ma bardzo niski koszt obliczeniowy

Wzór dla funkcji aktywacji ReLU:

( ) = ⎧0 jeśli < 0

⎨𝑅𝑒𝐿𝑈

⎩ jeśli ≥ 0

(11)

Funkcja aktywacji ReLU

5

4

3

2

1

0

−4

−2

0

2

4

Rysunek 6.1: Wykres funkcji ReLU.

Funkcja ReLU ma jednak ogromną wadę, potocznie nazywaną ‘śmiercią ReLU’ - polega to na tym, że w procesie uczenia niektóre neurony trwale ‘giną’, to znaczy zaczynają przesyłać wyłącznie sygnał 0 i nie ma możliwości, żeby ‘zmartwychwstały’[Gé18]. Z tego też powodu w praktyce stosowane są obecnie modyfikacje funkcji ReLU, z pośród który najpopularniejsze to:

* + - ‘przeciekająca’ funkcja ReLU
    - losowa ‘przeciekająca’ funkcja ReLU
    - parametryczna ‘przeciekająca’ funkcja ReLU
    - ELU (ang. *exponential linear unit* pl. jednostka wykładniczo liniowa)

W 2015 roku Djork-Arné Clevert wraz z zespołem odkrył nową funkcję aktywacji - ELU[CUH16]. Po przeprowadzonych eksperymentach okazało się, że góruje ona zarówno nad tradycyjną funkcją ReLU, jak i jej modyfikacjami. Umożliwiała skrócenie procesu uczenia, a także przyczyniła się wydatnie

do poprawy wyników dla zbioru testowego[Gé18]. Główne zalety ELU w zestawieniu z ReLU to:

* + - rozwiązanie ‘problemu zanikających gradientów’ dzięki przyjmowaniu ujemnych wartości dla x < 0
    - rozwiązanie problemu ‘umierających neuronów’ dzięki niezerowemu gradientowi dla x < 0
    - przyspieszenie procesu uczenia metodą gradientu prostego dzięki gładkości funkcji na cały przebiegu

Wadą jest natomiast niewątpliwie większy koszt obliczeniowy w porównaniu z ReLU spowodowany obecnością składowej wykładniczej.

Wzór dla funkcji aktywacji ELU:

Gdzie:

𝐸𝐿𝑈𝛼

( ) = ⎧𝛼(exp() − 1) jeśli < 0

⎩ jeśli ≥ 0

⎨

(12)

𝛼 hiperparametr definiujący wartość do jakiej ma się zbliżać funkcja dla ujemnych wartości x

Funkcja aktywacji ELU (α = 1)

5

4

3

2

1

0

−

2

0

2

4

−1

−

4

Rysunek 6.2: Wykres funkcji ELU dla 𝛼=1.

* 1. **Normalizacja wsadowa (ang. *Batch Normalization*)**

Zastosowanie kombinacji inicjacji He w połączeniu z funkcją aktywacji

ELU nie rozwiązało całkowicie problemu zanikających i eksplodujących gradientów. Problem nadal występował na dalszych etapach procesu uczenia. Dopiero zaproponowane w 2015 roku przez Sergeya Ioffe i Christiana Szegedy rozwiązanie

* normalizacja wsadowa (ang. *Batch Normalization*), w połączeniu z wcześniej wymienionymi pozwoliło ostatecznie pokonać trudność[IS15].

Operację normalizacji wsadowej wykonuje się na każdej warstwie bezpośrednio przed funkcją aktywacji. Najpierw wyśrodkowuje się i normalizuje się dane wyjściowe. Następnie przeskalowuje się i przesuwa się wyniki za pomocą dwóch parametrów wyliczanych dla każdej warstwy (𝛾 - parametr skalowania,

𝛽 - parametr przesunięcia)[Gé18].

Normalizacja wsadowa danych wejściowych dla pojedynczej warstwy: Wyliczenie średniej dla minigrupy:

𝜇𝐵 =

1

𝐵

() (13)

∑︁

𝐵 =1

Wyliczenie odchylenia standardowego dla minigrupy:

2 = 1

𝜎

𝐵

𝐵

(() − 𝜇𝐵

∑︁

)2 (14)

𝐵 =1

Wyśrodkowanie i znormalizowanie danych wejściowych:

() =

() − 𝜇𝐵

(15)

̂︀ √︁𝜎2

𝐵

+ 𝜖

Przeskalowanie i przesunięcie wyniku:

() = 𝛾() + 𝛽 (16)

Gdzie: ̂︀

𝜇𝐵 średnia empiryczna wyliczona dla całej minigrupy B

𝜎𝐵 empiryczne odchylenie standardowe wyliczone dla całej minigrupy B

𝐵 liczba przykładów w minigrupie

() wyśrodkowane i znormalizowane dane wejściowe

𝛾̂︀

parametr skalowania w danej warstwie

𝛽 parametr przesunięcia w danej warstwie

𝜖 niewielka liczba służąca do uniknięcia operacji dzielenia przez 0 (zazwyczaj ma wartość 10*−*5; jest to tzw, człon wygładzający (ang. *smoothing term*)

() wynik operacji normalizacji wsadowej: przeskalowana i przesunięta wersja danych wejściowych

Główne zalety normalizacji wsadowej to[Gé18]:

* + ograniczenie ‘problemu zanikających gradientów’ w stopniu pozwalającym na używanie nasycających funkcji aktywacji takich jak tangens hiperboliczny
  + zmniejszenie wrażliwości sieci na inicjację wag
  + umożliwienie wykorzystywania większej wartości współczynnika uczenia, a co za tym idzie znaczne przyspieszenie całego procesu uczenia
  + poprawa wyników dla zagadnień klasyfikacji i regresji
  + częściowe spełnianie roli regularyzatora

Największe wady to[Gé18]:

* + zwiększenie skomplikowania modelu
  + spowolnienie działania modelu, ze względu na konieczność przeprowadzenia dodatkowych obliczeń dla każdej warstwy (spowalnia to w szczególności początek procesu uczenia, gdyż algorytm gradientu prostego wyszukuje optymalne skale i przesunięcia dla każdej warstwy; następnie proces przyspiesza)

#### Regularyzacja przez Porzucanie (ang. *Dropout*)

Pierwotny pomysł zastosowania techniki ‘porzucania’ jako sposobu na regularyzację głębokiej sieci neuronowej powstał w 2012 roku. Jego autorem

był pracujący na Uniwersytecie Toronto G.E. Hinton (wraz ze współpracownikami).

Wyniki swoich badań zaprezentował w artykule “Improving neural networks by preventing co-adaptation of feature detectors”[Hin+12]. Jego dzieło rozwinął działający na tej samej uczelni Nitish Srivastava (ze swoim zespołem), a rezultaty opublikował w 2014 roku w artykule “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”[Sri+14]. Osiągane rezultaty były imponujące - zwiększenie dokładności klasyfikatorów o 1–2% co przekładało się na nawet kilkudziesięciu procentowe zmniejszenie współczynnika błędu.

Algorytm jest bardzo prosty. Podczas każdego przebiegu procesu uczenia dowolny neuron, za wyjątkiem neuronów warstwy wyjściowej, może zostać ‘porzucony’, to znaczy całkowicie pominięty w procesie uczenia, z ustalonym z góry prawdopodobieństwem *p*. Następnie, już po zakończeniu treningu, konieczne jest przemnożenie każdej wagi połączenia wejściowego przez tak zwane ‘prawdopodobieństwo utrzymania’ (ang. *keep probability*) równe 1 - *p*.

Alternatywą jest podzielenie wartości wejścia każdego neuronu przez ‘prawdopodobieństwo utrzymania’ w trakcie nauki[Gé18].

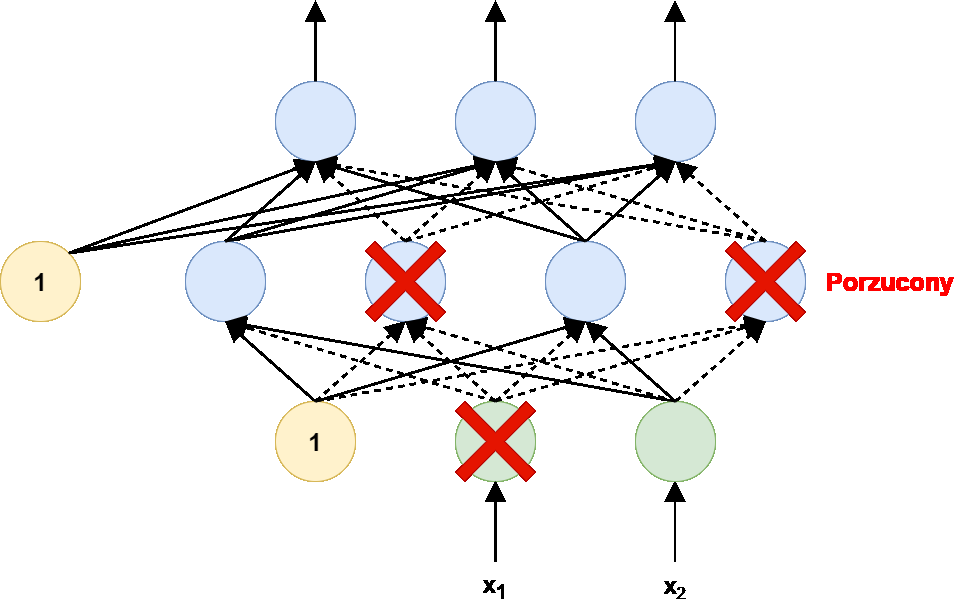
W praktyce dla każdej ‘epoki’ mamy do czynienia z unikalną siecią. Istnieje 2𝑁 możliwych kombinacji gdzie N oznacza całkowitą liczbę neuronów, które mogą zostać porzucone. Te sztucznie pomniejszone sieci są wobec siebie całkowicie niezależne. Finalna sieć jest pewnym rodzajem uśrednienia[Gé18].

Podsumowując, główne korzyści płynące z zastosowania techniki ‘porzucania’ to[Gé18]:

* + - uczone neurony nie mogą uzależniać się od swoich sąsiadów; same muszą wnosić jak najwięcej ‘wartości dodanej’ przy przetwarzaniu sygnału; muszą skupiać się na poszczególnych wejściach, a nie traktować ich kombinacji całościowo
    - wynikające z powyższego zmniejszenie wrażliwości na drobne zmiany na wejściach
    - patrząc całościowo uzyskany w wyniku ‘porzucania’ model jest lepszą generalizacją

Główną wadą jest spowolnienie procesu konwergencji modelu[Gé18].

Bardzo ważny jest prawidłowy dobór ‘współczynnika porzucania’ tak, żeby uniknąć zarówno niedotrenowania (zbyt duża wartość) jak i przetrenowania (zbyt mała wartość). Cenną wskazówką jest zastosowanie większej wartości współczynnika dla rozbudowanych warstw, a mniejszej dla niewielkich.



Rysunek 6.3: Regularyzacja przez ‘porzucanie’[Gé18].

* 1. **Optymalizator Adam (ang. *Adaptive Moment Estimation*)**

Optymalizatory są kolejnym elementem wprowadzonym w celu przyspieszenia procesu uczenia głębokich sieci neuronowych. Są one bardziej efektywne

od standardowego algorytmu gradientu prostego. W praktyce, ze względu na koszty pamięciowe, wykorzystuje się jedynie optymalizatory bazujące na

pochodnych cząstkowych pierwszego rzędu (jakobinach)[Gé18]. Najpopularniejsze z nich to:

* + - optymalizacja momentum (ang. *Momentum Optimization*)
    - przyspieszony spadek wzdłuż gradientu (algorytm Nesterova)
    - AdaGrad (ang. *Adaptive Subgradient Method*)
    - RMSProp
    - Adam (ang. *Adaptive Moment Estimation*)

Algorytm Adam został opracowany w 2015 roku przez Diederika P. Kingma

i Jimma Lei Ba[KB15]. Jest połączeniem pomysłów wykorzystanych w optymalizacji momentum i optymalizatorze RMSProp. Z optymalizacji momentum zaczerpnięte jest śledzenie rozkładu wykładniczego średniej wcześniejszych gradientów,

a z RMSProp - śledzenie rozkładu wykładniczego średniej wcześniejszych kwadratów momentów[Gé18]. Poniżej przedstawiono pseudokod z oryginalnej publikacji Kingma i Ba.

**Algorithm 2** Algorytm Adam[KB15].

**Require:** 𝛼 ◁ współczynnik uczenia, który zazwyczaj inicjowany jest wartością 𝛼 = 0.001

**Require:** 𝛽1, 𝛽2 (0, 1] ◁ 𝛽1 to hiperparametr rozkładu momentu, który zazwyczaj inicjowany wartością 𝛽1 = 0.9; 𝛽2 to hiperparametr rozkładu skalowania, który zazwyczaj inicjowany wartością

∈

𝛽2 = 0.999

**Require:** 𝜖 ◁ człon wygładzający, który zazwyczaj przyjmuje wartość

𝜖 = 10*−*8

**Require:** 𝑓 (𝜃) ◁ funkcja stochastyczna 𝑓 z wektorem parametrów 𝜃

**Require:** 𝜃0 ◁ postać początkowa wektora parametrów 𝜃

0 0 ◁ inicjacja pierwszego wektora momentu

←

0 0 ◁ inicjacja drugiego wektora momentu

←

0 ◁ inicjacja licznika przebiegów

←

**while** 𝜃 nie zbiegnie **do**

+ 1

←

𝑔 𝜃𝑓(𝜃 1) ◁ uzyskanie gradientów dla stochastycznej funkcji kosztu w przebiegu

← ∇ *−*

𝛽1 1 + (1 𝛽1) 𝑔 ◁ aktualizacja szacunkowej wartości pierwszego wektora momentu

← · − ·*−*

← 𝛽2 · *−*1 + (1 − 𝛽2) · 𝑔2 ◁ aktualizacja szacunkowej wartości

drugiego wektora momentu

obliczenie szacunkowej wartości pierwszego wektora

̂︁ ← *−*𝛽1

1 ◁

momentu

obliczenie szacunkowej wartości drugiego wektora

̂︀ ← 1*−*𝛽2 ◁

momentu

𝜃 ← 𝜃*−*1 − √𝛼*·*̂︁

+𝜖

◁ aktualizacja wektora parametrów

##### end while

̂︀

**return** 𝜃 ◁ wynikowy wektor parametrów

Optymalizator Adam, podobnie jak RMSProp, wykorzystuje adaptacyjny współczynnik uczenia. Z tego też powodu nie ma konieczności wyznaczania jego optymalnej wartości, wystarczy jedynie skorzystać z domyślnej wartości

𝜂 = 0.001[Gé18]. To z kolei sprawia, że Adam jest w praktyce prostszy w użyciu od metody gradientu prostego.

#### Wybór hiperparametrów procesu uczenia sieci neuronowej

Do estymacji za pomocą algorytmu genetycznego wybrano następujący zestaw hiperparametrów:

* + - liczba jednostek (neuronów) w warstwie ukrytej ‘1’ (min.10, max.200)
    - współczynnik porzucenia (ang. *dropout*) dla warstwy ukrytej ‘1’ (min. 0.01, max. 0.5)
    - liczba jednostek dla warstwie ukrytej ‘2’ (min.10, max.200)
    - współczynnik porzucenia (ang. *dropout*) dla warstwy ukrytej ‘2’ (min. 0.01, max. 0.5)
    - liczba jednostek dla warstwie ukrytej ‘3’ (min.10, max.200)
    - współczynnik porzucenia (ang. *dropout*) dla warstwy ukrytej ‘3’ (min. 0.01, max. 0.5)
    - rozmiar minigrupy (ang. *batch size*); (min.100, max.400)

W przypadku pozostały hiperparametrów zdecydowano się na przyjęcie zalecanych wartości domyślnych (optymalizator Adam: 𝜂 = 0.001, 𝛽1 = 0.9,

𝛽2 = 0.999, 𝜖 = 10*−*7; normalizacja wsadowa: 𝑒 = 0.99).

* 1. **Wybór rodziców z wykorzystaniem selekcji proporcjonalnej (ang. *Fitness Proportionate Selection / FPS*)**

Selekcja proporcjonalna (ang. *Fitness Proportionate Selection / FPS* ) lub inaczej selekcja ruletkowa (ang. *Roulette Wheel Selection*) jest popularną metodą wykorzystywaną do wyboru odpowiednich osobników (rodziców) do rekombinacji (krzyżowania). Autorem oryginalnego konceptu był J.H Holland z Massachusetts Institute of Technology. Selekcja polega na losowaniu osobnika z prawdopodobieństwem równym stosunkowi absolutnej wartości jego metryki dopasowania (ang. *fitness*) 𝑓 do sumy wartości absolutnych metryk dopasowań

dla całej populacji ∑︀𝜇 𝑓[ES15].

=1

Wzór na prawdopodobieństwo wylosowania pojedynczego osobnika prezentuje się następująco:

𝑃 () = |𝑓|

∑︀

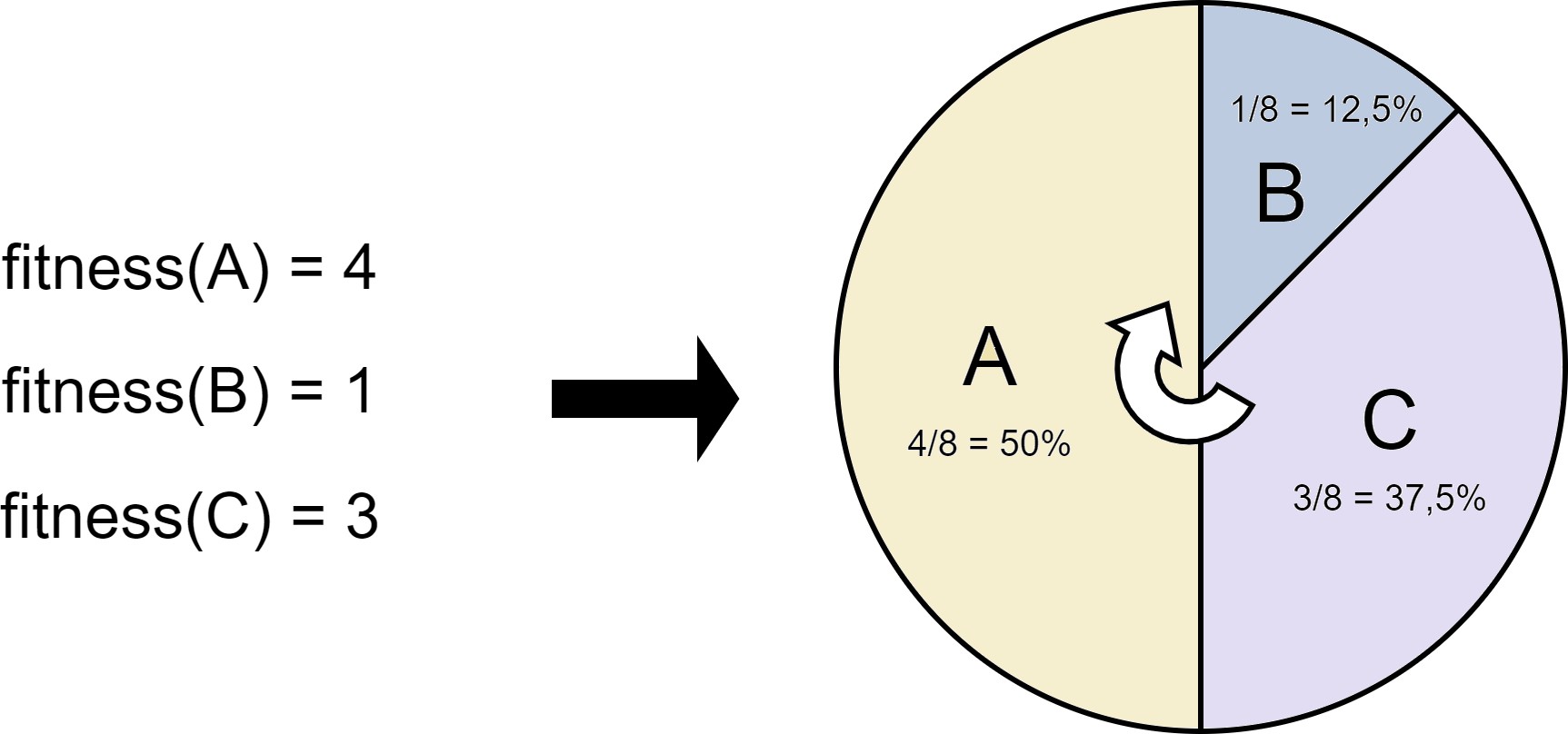
(17)

𝐹 𝑃 𝑆

𝜇

=1

|𝑓|



Rysunek 6.4: Obrazowy przykład działania selekcji proporcjonalnej.

Okazało się, że selekcja proporcjonalna wyjątkowo dobrze nadaje się do

analizy teoretycznej. Niestety bardzo szybko odkryto też jej główne mankamenty[ES15]:

* Najlepsze osobniki bardzo szybko przejmują ‘kontrolę’ nad populacją. To powoduje, że zostaje bardzo zawężony obszar poszukiwań najlepszego rozwiązania, a prawdopodobieństwo uzyskania dobrego wyniku jest nikłe. Problem uwidacznia się zwłaszcza we wczesnych generacjach, kiedy wiele losowo utworzonych osobników ma niską wartość metryki dopasowania (ang. *fitness*). Fachowo określa się to terminem przedwczesnej konwergencji (zbiegnięcia) (ang. *premature convergence*).
* Kiedy wartości metryki dopasowania są zbliżone do siebie zanika tak zwana ‘presja selekcyjna’ (ang. *selection pressure*). Rezultatem jest niemal losowa selekcja w kolejnych pokoleniach (generacjach) i bardzo powolna konwergencja (zbieganie) do optymalnego rozwiązania.

Na szczęście opracowane zostały skuteczne techniki zaradcze. Najpopularniejsze

to:

* ‘okienkowanie’ (ang. *windowing*)
* skalowanie sigma (ang. *sigma scaling*)

‘Okienkowanie’ (ang. *windowing*), z którego skorzystano w ramach niniejszej pracy, polega na odejmowaniu od pierwotnej wartości metryki dopasowania

𝑓 wartości 𝛽 zależnej od przebiegu działania algorytmu i zmiennej w czasie.

Przeważnie jest to najmniejsza (lub najgorsza) wartość metryki dopasowania dla całej populacji z kilku wcześniejszych pokoleń[ES15].

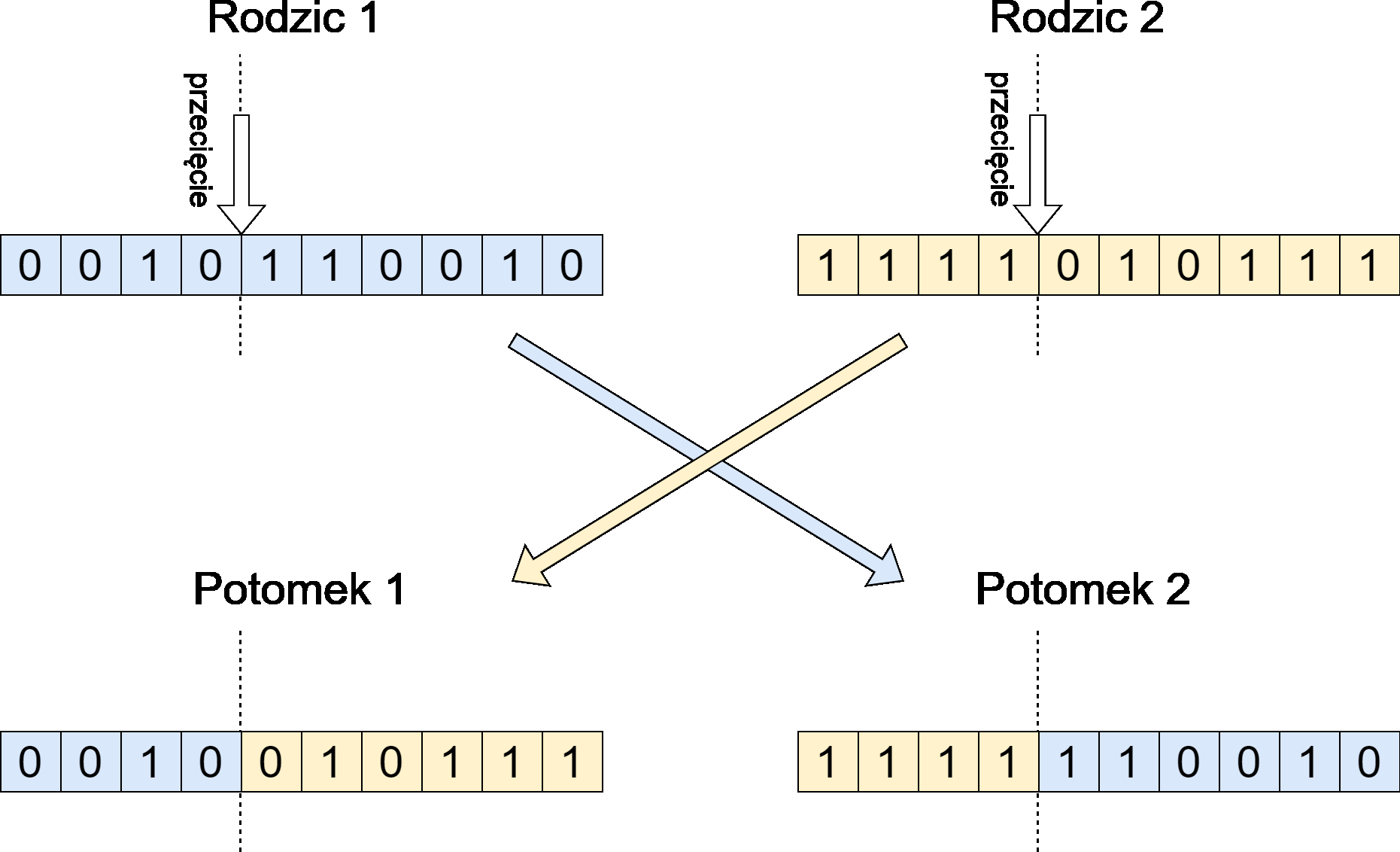
𝑓 *′*() = 𝑓 () − 𝛽 (18)

#### Krzyżówka z jednym punktem przecięcia (ang.

***One Point Crossover* )**

Krzyżówka z jednym punktem przecięcia (ang. *One Point Crossover* ) polega na wyznaczeniu losowego punktu, przez który dzieli się następnie chromosomy rodziców, a z otrzymanych części buduje się finalnie chromosomy potomków. Przykładowo, jeśli chromosomy mają postać ciągów bitowych, pierwszy rodzic to 0010110010, drugi rodzic to 1111010111, a punkt przecięcia

został wyznaczony na 4, to w wyniku operacji krzyżowania otrzymamy następujących potomków: 0010010111 i 1111110010[Kra17].



Rysunek 6.5: Obrazowy przykład działania krzyżowania z jednym punktem przecięcia.

W implementacji algorytmu genetycznego, będącej przedmiotem niniejszej pracy, autor wykorzystał krzyżówkę z dwoma punktami przecięcia. Działa ona na podobnej zasadzie, ale rezultatem jest sześciu potomków o genotypie różnym od rodziców.

* 1. **Mutacja pełzająca (ang. *Creep Mutation*), niejednorodna (ang. *nonuniform*)**

Mutacja pełzająca (ang. *Creep Mutation*) jest często stosowanym rozwiązaniem dla alleli (parametrów) typu *integer* (całkowitych) lub *float* (zmiennoprzecinkowych). Polega ona na translacji pierwotnej wartości allelu poprzez dodanie niewielkiej (dodatniej lub ujemnej) liczby losowej, tak by rezultat mieścił się w granicach

*′* ∈ [𝐿, 𝑈][ES15].

Transformacja wartości parametrów osobnika w ramach mutacji:

⟨1, . . . , >⟩ → ⟨*′* , . . . , *′* ⟩ gdzie , *′* ∈ [𝐿, 𝑈] (19)

Gdzie:

1

𝐿 dolna granica dla transformacji

𝑈 górna granica dla transformacji

Stosować można zarówno mutację jednorodną (ang. *Uniform Mutation*)

* opartą o liczby losowe wygenerowane według rozkładu jednorodnego, jak i mutację niejednorodną (ang. *Nonuniform Mutation*) - opartą o rozkład Gaussa (rzadziej o rozkład Cauchego)[ES15]. W przypadku niniejszej pracy zdecydowano się na generowanie przesunięć (Δ) w oparciu o rozkład Gaussa, a następnie na ograniczanie wyliczonych nowych wartości parametrów do z góry wyznaczonych granic (𝐿, 𝑈). Do sterowania losowością przesunięć stosuje się odchylenie standardowe 𝜎, tu określane jako ‘miara kroku mutacji’ (ang. *mutation step size*). Dla niniejszej pracy przyjęto, ze względu na przebieg funkcji gęstości prawdopodobieństwa dla rozkładu Gaussa: 𝜎 = (𝑈 − 𝐿)/6.

Funkcja gęstości prawdopodobieństwa dla rozkładu Gaussa:

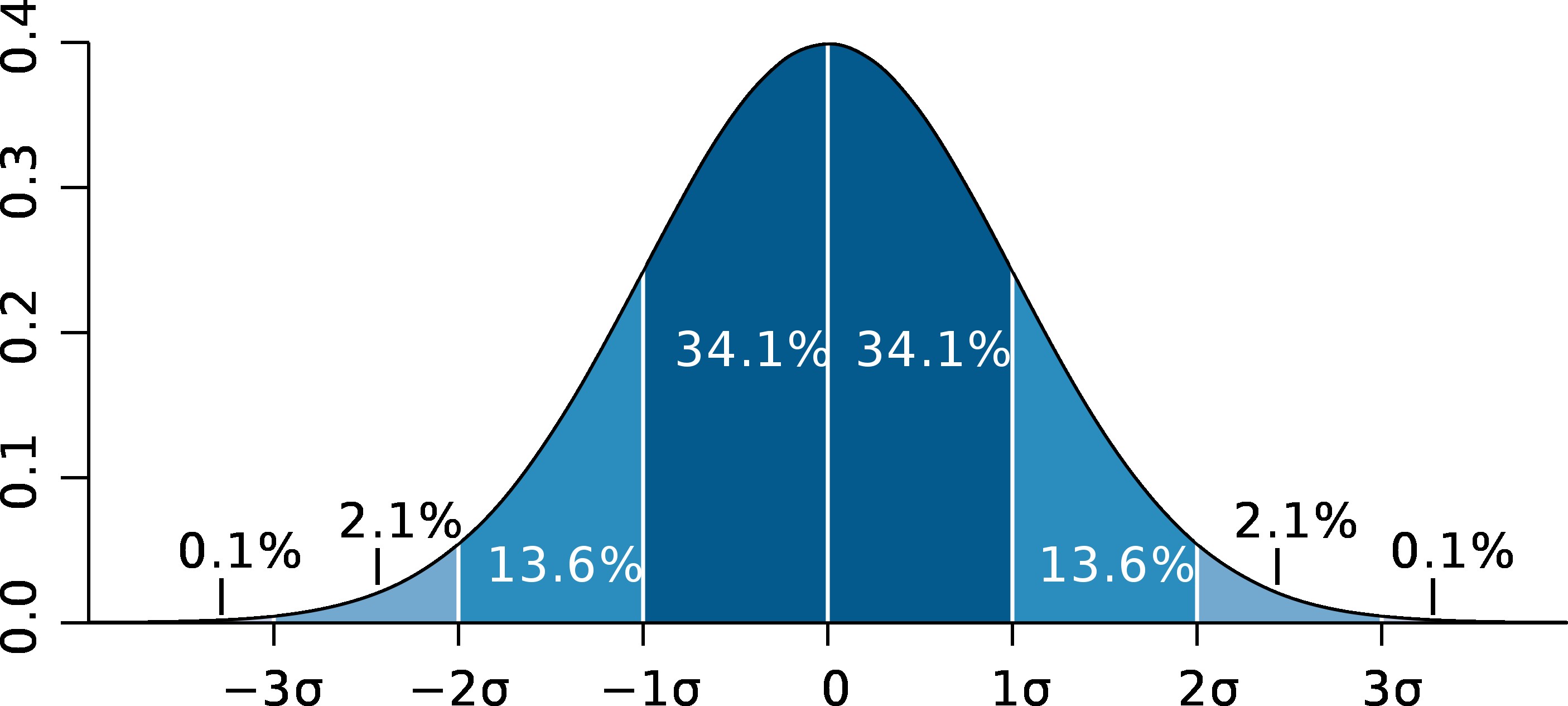
(Δ) = 𝜎√2𝜋 · 𝑒

1

(Δ − 𝜉)2 (20)

𝜎

2 2



Rysunek 6.6: Wykres dla rozkładu Gaussa. Jednostka dla osi *x* to odchylenie standardowe (𝜎). W odległości 1 𝜎 od średniej znajduje się 68.27% elementów; w odległości 2 𝜎 - 95.45% elementów; w odległości 3 𝜎 - 99.73% elementów. Autor: M.W. Toews (Wikipedia Community).

· ·

·

W implementacji algorytmu genetycznego, będącej przedmiotem niniejszej

pracy, autor poddawał mutacji 1 wybranych losowo cech osobnika.

3

## Rezultaty eksperymentów

Wszystkie eksperymenty przeprowadzono w środowisku cloudowym Colab Pro Plus (Google). Wykorzystano biblioteki Pandas i Numpy (wczytanie i

przygotowanie danych), Sklearn (przygotowanie zbioru treningowego, walidacyjnego i testowego), wszystkie modele sieci neuronowych zostały stworzone i wytrenowane w TensorFlow Keras (Google).

Właściwy przebieg algorytmu ograniczono do 120 pokoleń. Liczbę osobników w populacji ustalono na 20. W każdym pokoleniu 6 najgorszych osobników wymieniano na 6 potomków (powstałych po krzyżówce 2-punktowej i mutacjach). Zbiór treningowy miał 95565 przykładów, a walidacyjny - 23891 (podział w proporcjach 0.8-0.2). Zbiór testowy zawierał 51196 przykładów (podział w proporcjach 0.7-0.3).

Funkcją dopasowania była minimalna wartość metryki MSE dla osobnika, wyznaczona dla zbioru walidacyjnego podczas uczenia.

Maksymalna liczba epok (pełnych przebiegów po zbiorze treningowym) została wyznaczona na 20. Dla skrócenia czasu nauki pojedynczego modelu skorzystano z mechanizmu Keras Early Stopping (tzw. ‘wcześniejsze zatrzymanie’). Ustawiono następujące wartości parametrów:

* = 𝑎\_ - monitorowana metryka to strata na zbiorze walidacyjnym (MSE walidacyjne)
* \_𝑑𝑒𝑎 = 0.1 - minimalna wartość poprawy metryki
* 𝑎𝑒𝑐𝑒 = 3 - cierpliwość w oczekiwaniu na poprawę wyniku (w epokach)
* 𝑑𝑒 = - tryb minimalizacji monitorowanej metryki

Dodatkowo do mechanizmu uczącego podpięto Keras CSVLogger - mechanizm logujący, zbierający wartości metryk ze wszystkich epok i zapisujący je do pliku CSV, a także customowy logger stworzony przez autora i podpięty bezpośrednio pod kod dla algorytmu genetycznego, który zapisywał do pliku CSV następujące dane: numer pokolenia, wartość funkcji dopasowania (FF)

dla najlepszego osobnika w populacji, wartości poszczególnych parametrów dla najlepszego osobnika, FF dla najgorszego osobnika, wartości poszczególnych parametrów dla najgorszego osobnika, średnia FF, mediana FF, odchylenie standardowe FF.

Poniżej przedstawiono wykresy ilustrujące przebieg algorytmu genetycznego:

98.75

98.50

Wartośś funkcji dopasowania

(MSE walidacyjne)

98.25

98.00

97.75

97.50

97.25

97.00

Dopasowanie najlepszego osobnika w populacji

0 20 40 60 80 100 120

Numer pokolenia

Rysunek 7.1: Wykres zmiany wartości metryki MSE dla najlepszego osobnika w trakcie życia populacji.

Do asowanie najgorszego osobnika w o ulacji

108

106

Wartośś funkcji do asowania ćMSE walidacyjne)

104

102

100

98

0 20 40 60 80 100 120

Numer okolenia

Rysunek 7.2: Wykres zmiany wartości metryki MSE dla najgorszego osobnika w trakcie życia populacji.

Mediana dopa owania o obnika w populacji

101

Wartośś funkcji dopa owania ćMSE walidacyjne)

100

99

98

0 20 40 60 80 100 120

Numer pokolenia

Rysunek 7.3: Wykres zmiany mediany metryki MSE w trakcie życia populacji.

### Ś ednie dopasowanie osobnika w populacji

101

Wa tośś funkcji dopasowania

100

ćMSE walidacyjne)

99

98

0 20 40 60 80 100 120

Numer pokolenia

Rysunek 7.4: Wykres zmiany średniej metryki MSE w trakcie życia populacji.

Odchylenie standardowe dopasowanie osobnika

2.5

2.0

Odchylenie standardowe funkcji

dopasowania

1.5

1.0

0.5

0 20 40 60 80 100 120

Numer pokolenia

Rysunek 7.5: Wykres zmiany odchylenia standardowego metryki MSE w trakcie życia populacji.

Wartość FF dla najlepszego osobnika w ostatnim pokoleniu wyniosła:

𝑀𝑆𝐸𝑎𝑑 **= 97**.**01777648925781**. Oznaczało to poprawę o 1.7% w stosunku do najlepszego osobnika z pokolenia ‘zero’ i aż 10.11% (!) licząc w odniesieniu do najgorszego osobnika z pierwotnej populacji.

Szczegółowe wartości wyliczonych parametrów dla najlepszego osobnika prezentują się następująco:

* ℎ𝑑𝑑𝑒\_1\_ = 181
* ℎ𝑑𝑑𝑒\_1\_𝑑\_𝑎𝑒 = 0.10379824743424901
* ℎ𝑑𝑑𝑒\_2\_ = 112
* ℎ𝑑𝑑𝑒\_2\_𝑑\_𝑎𝑒 = 0.09567409338419407
* ℎ𝑑𝑑𝑒\_3\_ = 55
* ℎ𝑑𝑑𝑒\_3\_𝑑\_𝑎𝑒 = 0.01
* 𝑏𝑎𝑐ℎ\_𝑒 = 100

Poniżej przedstawiono wykresy zmiany wartości parametrów dla najlepszego osobnika:

Liczba jednostek w warstwie ukrytej 1 dla najlepszego osobnika

180

160

Liczba neuronów

140

120

100

0 20 40 60 80 100 120

Numer pokolenia

Rysunek 7.6: Wykres zmiany wartości liczby jednostek w warstwie ukrytej 1 dla najlepszego osobnika w populacji.

0.225

0.200

Współczynnik porzucenia

0.175

0.150

0.125

0.100

0.075

0.050

### Współczynnik porzucenia dla warstwy ukrytej 1 dla najlepszego osobnika

0 20 40 60 80 100 120

Numer pokolenia

Rysunek 7.7: Wykres zmiany wartości współczynnika porzucenia w warstwie ukrytej 1 dla najlepszego osobnika w populacji.

Liczba jednostek w warstwie ukrytej 2 dla najlepszego osobnika

150

140

130

Liczba neuronów

120

110

100

90

80

70

0 20 40 60 80 100 120

Numer pokolenia

Rysunek 7.8: Wykres zmiany wartości liczby jednostek w warstwie ukrytej 2 dla najlepszego osobnika w populacji.

### Współczynnik porzucenia dla warstwy ukrytej 2 dla najlepszego osobnika

0.175

0.150

Współczynnik porzucenia

0.125

0.100

0.075

0.050

0.025

0 20 40 60 80 100 120

Numer pokolenia

Rysunek 7.9: Wykres zmiany wartości współczynnika porzucenia w warstwie ukrytej 2 dla najlepszego osobnika w populacji.

Liczba jednostek w warstwie ukrytej 3 dla najlepszego osobnika

160

140

Liczba neuronów

120

100

80

60

0 20 40 60 80 100 120

Numer pokolenia

Rysunek 7.10: Wykres zmiany wartości liczby jednostek w warstwie ukrytej 3 dla najlepszego osobnika w populacji.

0.35

0.30

Współczynnik porzucenia

0.25

0.20

0.15

0.10

0.05

0.00

### Współczynnik porzucenia dla warstwy ukrytej 3 dla najlepszego osobnika

0 20 40 60 80 100 120

Numer pokolenia

Rysunek 7.11: Wykres zmiany wartości współczynnika porzucenia w warstwie ukrytej 3 dla najlepszego osobnika w populacji.

Rozmiar minigrupy procesu uczenia dla najlepszego osobnika

300

275

250

Rozmiar minigrupy

225

200

175

150

125

100

0 20 40 60 80 100 120

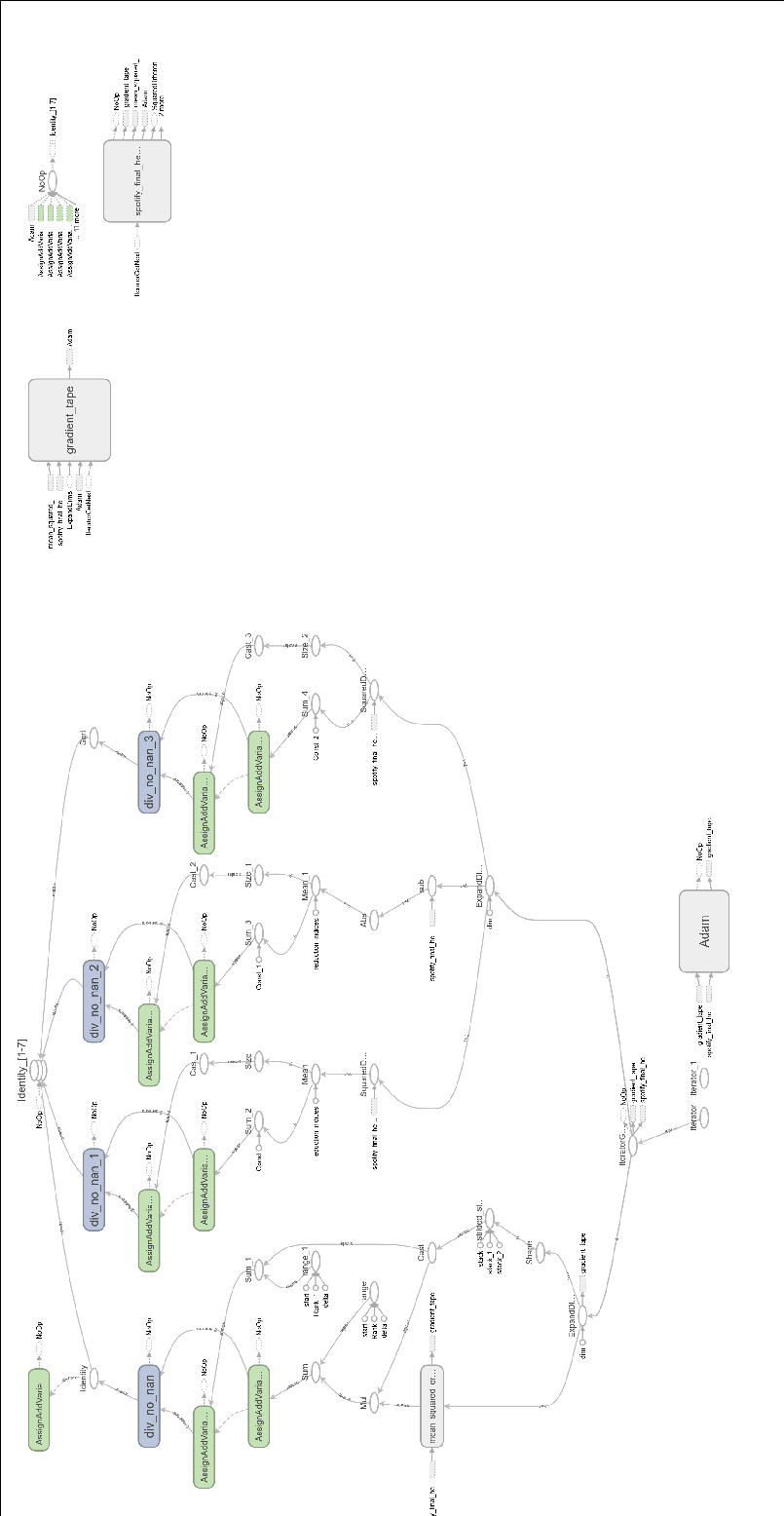
Numer pokolenia

Rysunek 7.12: Wykres zmiany wielkości minigrupy dla najlepszego osobnika w populacji.

Podczas uczenia dokładnego wyliczonego modelu zwiększono maksymalną liczbę epok do 100. Zmieniono też parametry mechanizmu Keras EarlyStopping:

* \_𝑑𝑒𝑎 = 0.01
* 𝑎𝑒𝑐𝑒 = 10
* 𝑒𝑒\_𝑏𝑒\_𝑒𝑔ℎ = 𝑇 𝑒 - przywrócenie wartości wag dla połączeń z najlepszego przebiegu

Dodatkowo do poprawy monitoringu i wizualizacji procesu uczenia skorzystano z narzędzia Tensorboard.



Rysunek 7.13: Graf z TensorBoard przedstawiający architekturę wybranej sieci neuronowej.

Uzyskano wynik: 𝑀𝑆𝐸𝑎𝑑 **= 96**, **329475402832** - poprawa o kolejne 0.71% (poprawa względem najlepszego z pokolenia ‘zero’ o 2,4%, a względem najgorszego o 10,75%).

Poniżej przedstawiono wykresy obrazujące przebieg procesu uczenia modelu stworzonego w oparciu o hiperparametry wyznaczone przez algorytm genetyczny:

Porównanie MSE dla zbioru treningowego i walidacyjnego

MSE dla zbioru treningowego MSE dla zbioru walidacyjnego

500

400

300

MSE

200

100

0 10 20 30 40 50 60

Numer epoki

Rysunek 7.14: Wykres zestawiający wartości metryk MSE dla zbioru treningowego i walidacyjnego podczas procesu uczenia sieci neuronowej.

Porównanie MAE dla zbioru treningowego i walidacyjnego

MAE dla zbioru treningowego MEA dla zbioru walidacyjnego

18

16

14

12

MAE

10

8

0 10 20 30 40 50 60

Numer epoki

Rysunek 7.15: Wykres zestawiający wartości metryk MAE dla zbioru treningowego i walidacyjnego podczas procesu uczenia sieci neuronowej.

Porównanie RMSE dla zbioru treningowego i walidacyjnego

RMSE dla zbioru treningowego RMSE dla zbioru walidacyjnego

22

20

18

16

RMSE

14

12

10

0 10 20 30 40 50 60

Numer epoki

Rysunek 7.16: Wykres zestawiający wartości metryk RMSE dla zbioru treningowego i walidacyjnego podczas procesu uczenia sieci neuronowej.

Otrzymane wyniki zestawiono z rezultatami otrzymanymi dla mniejszych modeli zaprojektowanych w oparciu o metodę ‘prób i błędów’ i wskazówki z

literatury przedmiotu, a także z siecią 2-warstwową zaprojektowaną z wykorzystaniem algorytmu genetycznego. Porównanie przykładowych architektur sieci neuronowych

z wybranym modelem:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Metryka | ŚN1 | ŚN2 | ŚN3 | ŚN4 |
| MSE | 96Ś705 | 94Ś746 | 94Ś356 | 93Ś144 |
| MAE | 6Ś939 | 6Ś803 | 6Ś892 | 6Ś737 |
| RMSE | 9Ś834 | 9Ś734 | 9Ś714 | 9Ś651 |

Rysunek 7.17: Tabelaryczne zestawienie wartości metryk otrzymanych dla różnych architektur sieci neuronowych (sieć neuronowa 1 (ŚN1): 30 jednostek

(u) w warstwie ukrytej 1 (h1), 15 u w h2; ŚN2: h1=40u, h2=20u, h3=40u; ŚN3: h1=201u, h2=177u; ŚN4 (wybrana!): h1=181u, h2=112u, h3=55u), uzyskanych na zbiorze testowym.

100

# Po ównanie wa tości MSE dla zbio u testowego

98

96

MSE

94

92

90

h1=30u;

h2=15u

h1=40u; h2=20u;

h3=10u

h1=201u; h2=177u

h1=181; h2=112; h3=55;

(wybrana)

Sieć neu onowa

Rysunek 7.18: Zestawienie wartości metryk MSE otrzymanych dla różnych architektur sieci neuronowych (nazwy odnoszą się do liczby neuronów w poszczególnych warstwach ukrytych), uzyskanych na zbiorze testowym. **Widać poprawę wyniku o 1.69% w porównaniu do mniejszego modelu o podobnej architekturze.**

P równanie wart ści MAE dla (bi ru test weg

7.4

7.2

7.0

6.8

MAE

6.6

6.4

6.2

6.0

h1=30u;

h2=15u

h1=40u; h2=20u;

h3=10u

h1=201u; h2=177u

h1=181; h2=112; h3=55;

(wybrana)

Sieć neur n wa

Rysunek 7.19: Zestawienie wartości metryk MAE otrzymanych dla różnych architektur sieci neuronowych (nazwy odnoszą się do liczby neuronów w poszczególnych warstwach ukrytych), uzyskanych na zbiorze testowym. **Widać poprawę wyniku o 0.85% w porównaniu do mniejszego modelu o podobnej architekturze.**

10.0

# Porów a ie wartości RMSE dla zbioru testowego

9.8

9.6

RMSE

9.4

9.2

9.0

h1=30u;

h2=15u

h1=40u; h2=20u;

h3=10u

h1=201u; h2=177u

h1=181; h2=112; h3=55;

(w(brana)

Sieć euro owa

Rysunek 7.20: Zestawienie wartości metryk RMSE otrzymanych dla różnych architektur sieci neuronowych (nazwy odnoszą się do liczby neuronów w poszczególnych warstwach ukrytych), uzyskanych na zbiorze testowym.

Najlepsza i najgorsza predykcja:

|  |  |
| --- | --- |
| Najlepsza | predykcja |
| Nazwa utworu: | Basie English - Take 2 |
| Artysta: | Lester Young |
| Gatunek: | Jazz |
| Rok publikacji: | 1944 |
| Wartość rzecz. popularności: | 4 |
| Wartość predykcji: | 4.0000167 |
| Błąd absolutny: | 1.66893005e-05 |

|  |  |
| --- | --- |
| Najgorsza | predykcja |
| Nazwa utworu: | Chika Dance |
| Artysta: | AnimeHub |
| Gatunek: | Anime |
| Rok publikacji: | 2019 |
| Wartość rzecz. popularności: | 69 |
| Wartość predykcji: | -7.619051 |
| Błąd absolutny: | 76.61905098 |

Losowe przykłady:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| W.R | ŚN1 | ŚN2 | ŚN3 | ŚN4 |
| 34.0 | 37.221 | 36.033 | 37.21 | 36.793 |
| 26.0 | 33.727 | 33.Ś75 | 34.772 | 33.602 |
| 3Ś.0 | 41.499 | 41.721 | 42.015 | 43.57 |
| 13.0 | 11.5Ś6 | 11.296 | 15.029 | 10.6Ś4 |
| 0.0 | 1.103 | 0.704 | -0.5Ś3 | 0.69 |
| 62.0 | 62.6Ś3 | 64.264 | 63.234 | 63.975 |
| 23.0 | 32.053 | 29.663 | 2Ś.Ś5Ś | 30.036 |
| 0.0 | 4.Ś2Ś | 3.969 | 4.145 | 1.409 |
| 25.0 | 30.417 | 29.5Ś4 | 30.0Ś4 | 29.605 |
| 25.0 | 36.496 | 35.416 | 35.304 | 34.122 |

Rysunek 7.21: Tabelaryczne zestawienie wartości rzeczywistych dla wybranych przykładów z predykcjami otrzymanymi dla różnych architektur sieci neuronowych (sieć neuronowa 1 (ŚN1): 30 jednostek (u) w warstwie ukrytej 1 (h1), 15 u w h2; ŚN2: h1=40u, h2=20u, h3=40u; ŚN3: h1=201u, h2=177u; ŚN4 (wybrana!): h1=181u, h2=112u, h3=55u), uzyskanymi na zbiorze testowym.

100

(artośś rzecz)(ista h1=30u; hć=15u

spotif)\_he\_elu\_adam\_40\_ć0\_10 h1=ć01u; hć=177u

h1=181; hć=11ć; h3=55; (()bra a)

Poró( a ie (artości rzecz)(ist)ch i ()bra )ch pred)kcji

80

60

40

Popularnośś (0-100)

20

0

0 2 4 6 8 10

Przykład

Rysunek 7.22: Wizualizacja zestawienia wartości rzeczywistych z predykcjami wykonana na podstawie tabeli 7.21.

## Wnioski

Dzięki zastosowaniu algorytmu genetycznego udało się poprawić wynik dla modelu 3-warstwowego o 1,69% licząc w oparciu o metrykę MSE. Wiązało się to ze skonstruowaniem dużo większej sieci od tej stworzonej pierwotnie na etapie *Proof of concept* (pl. sprawdzenie pomysłu).

Przy tworzeniu modelu wzięto pod uwagę wyłącznie cechy muzyczne. Na etapie *Proof of concept* podjęto próby włączenia do modelu tytułów utworów poprzez ich transformację z wykorzystaniem techniki *word embedding* (przeliczenie na wartości numeryczne odległości semantycznych między frazami). Dało to poprawę wyniku o około 0.5% dla metryki MSE, ale z pomysłu wycofano

się ze względu na konieczność uproszczenia modelu, a także fakt że użycie wspomnianej techniki wymagałoby znacznego rozbudowania niniejszego opracowania.

Kolejnym pomysłem, który rozważano, było wprowadzenie do modelu 4tej i 5tej warstwy ukrytej. Pomysł zarzucono ze względu na brak dostępu do odpowiedniego środowiska wykonawczego. Szacunkowe próby na etapie *Proof of concept* wskazywały jednak, że dodawanie kolejnych warstw ukrytych po 3ciej nie przynosi już znaczącej poprawy wyniku, a przy niekorzystnym układzie liczby jednostek w poszczególnych warstwach może wręcz spowodować jego pogorszenie.

Autor niniejszego opracowania uznaje finalny wynik: 𝑀𝑆𝐸 **= 93**.**144**,

𝑀𝐴𝐸 **= 6**.**737**, 𝑅𝑀𝑆𝐸 **= 9**.**651** za satysfakcjonujący (biorąc uwagę, że cecha szacowana *popularity* była w skali 0–100, to błąd wyniósł dla RMSE 9.651%, a dla MAE jedynie 6.737%). Na skalę błędu miały niewątpliwie wpływ następujące czynniki: kampanie promocyjne i sposoby ekspozycji treści przez Spotify, szeroko rozumiane mody, ogólna renoma artysty, jakość warstwy tekstowej utworu. Widać natomiast, że same dane odnośnie warstwy muzycznej są niezwykle cennym nośnikiem informacji, pozwalającym na tworzenie wysokiej jakości predykcji.

Podsumowując, algorytm genetyczny okazał się być dobrym narzędziem do optymalizacji architektury i procesu uczenia sieci neuronowych. Trzeba jednak wyraźnie stwierdzić, że ze względu na fakt iż jest to szablon, a nie gotowe narzędzie jego implementacja i dostrojenie wymaga ogromnych nakładów pracy.

## Bibliografia

[Ay] Yamac Eren Ay. *Spotify Dataset 1921-2020 160K+ Tracks*. url: https :// www . kaggle . com / yamaerenay / spotify - dataset - 19212020-160k-tracks (term. wiz. 25. 11. 2020).

[Bri] Brilliant.org. *Backpropagation*. url: https://brilliant.org/ wiki/backpropagation/ (term. wiz. 11. 01. 2022).

[Coma] Wikipedia Community. *Gene*. url: https :// en . wikipedia . org/wiki/Gene (term. wiz. 23. 01. 2022).

[Comb] Wikipedia Community. *Hit Song Science*. url: https :// en . wikipedia.org/wiki/Hit\_Song\_Science (term. wiz. 26. 10. 2021).

[Comc] Wikipedia Community. *Polyphonic HMI*. url: https : / / en . wikipedia.org/wiki/Polyphonic\_HMI (term. wiz. 26. 10. 2021).

[CUH16] Djork-Arné Clevert, Thomas Unterthiner i Sepp Hochreiter. “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)”. W: *ICLR conference paper* (2016).

[Drö21] Christoph Drösser. *Muzyka Daj się uwieść!* Wydawnictwo Naukowe PWN, 2021.

[ES15] A.E. Eiben i J.E. Smith. *Introduction to Evolutionary Computing*.

Springer Science+Business Media, 2015.

[Fel18] Michał Feliksiak. *Słuchanie muzyki*. Komunikat z badań. Fundacja Centrum Badania Opinii Społecznej, 2018.

[GB10] Xavier Glorot i Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. W: *DIRO, Université de Montréal, Montréal, Québec, Canada* (2010).

[Gé18] Aurélien Géron. *Uczenie maszynowe z użyciem Scikit-Learn i TensorFlow*.

Helion S.A., 2018.

[He+15] Kaiming He i in. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”. W: *Microsoft Research* (2015).

[Hin+12] G. E. Hinton i in. “Improving neural networks by preventing co-adaptation of feature detectors”. W: *Department of Computer Science, University of Toronto* (2012).

[IS15] Sergey Ioffe i Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”.

W: *Google Inc.* (2015).

[KB15] Diederik P. Kingma i Jimmy Lei Ba. “Adam: a Method for Stochastic Optimization”. W: *ICLR conference paper* (2015).

[Kra17] Oliver Kramer. *Genetic Algorithm Essentials*. Springer International Publishing AG, 2017.

[Mit96] Melanie Mitchell. *An Introduction to Genetic Algorithms*. A Bradford Book The MIT Press, 1996.

[MP43] Warren S. McCulloch i Walter Pitts. “A Logical Calculus of the Ideas Immanent in Nervous Activity.” W: *Bulletin of Mathematical Biophysics Volume 5* (1943).

[Ni+11] Yizhao Ni i in. “Hit Song Science Once Again a Science?” W:

*unknown* (2011).

[PJN20] Magdalena Parus-Jankowska i Szymon Nożyński. “Preferencje Muzyczne w Dobie Strumieniowego Słuchania Muzyki”. W: *Studia Humanistyczne AGH* (2020).

[PR09] François Pachet i Pierre Roy. “Hit Song Science is Not yet a Science”. W: *Proc. of Ismir ‘09* (2009).

[RHW85] David E. Rumelhart, Geoffrey E. Hinton i Ronald J. Williams. “Learning Internal Representations by Error Propagation”. W: *ICS Report 8506* (1985).

[Rot19] Stephanie Clare Roth. “What is genomic medicine?” W: *Journal of the Medical Library Association. University Library System, University of Pittsburgh.* (2019).

[Sri+14] Nitish Srivastava i in. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. W: *Journal of Machine Learning Research* (2014).

[Tea] Spotify Development Team. *Web Api Reference*. url: https:// developer.spotify.com/documentation/web-api/reference/ (term. wiz. 18. 10. 2021).

[TS15] Ryszard Tadeusiewicz i Maciej Szaleniec. *Leksykon Sieci Neuronowych*.

Wydawnictwo Fundacji „Projekt Nauka”, 2015.