Sprawozdanie z wykonania projektu – „Stworzyć program do estymacji jakości sieci teleinformatycznej za pomocą wybranych modeli uczenia maszynowego” PSZT w semestrze 2020Z

Skład zespołu: Magdalena Majkowska, Bartłomiej Janowski

Prowadzący: Stanislaw Kozdrowski

Problem badawczy

Celem naszego projektu jest wykorzystanie dwóch metod uczenia maszynowego do zbadania jakości ścieżek światłowodowych i porównania ich działania i jakości przewidywań. Jakość danej ścieżki jest oceniana na podstawie czynnika OSNR, którego wartość jest arbitralnie przydzielana i zależy jedynie od sumarycznej długości połączenia.

W naszym projekcie dla łatwości obliczeń zostały przyjęte poniższe wartości OSNR w zależności od długości ścieżki:

Długość ścieżki [km]	Wartość OSNR	Przydzielona klasa
0-200	25	3
200-400	20	2
400-600	15	1
>600	10	0

Badany graf połączeń obejmuje zbiór 50 wierzchołków sieci światłowodowej w Niemczech dostępny do pobrania ze strony - http://sndlib.zib.de/home.action. Niestety na stronie nie jest dostępna lista ścieżek w grafie, dlatego, zostały one wygenerowane w sposób automatyczny w programie (jako 2,5 tyś najkrótszych ścieżek pomiędzy każdą parą miast, oraz kolejne 7,5 tyś losowych) i podzielone na dwa zbiory:

- Zbiór uczący – stanowiący około 80% całości, złożony ze ścieżek, na które nie zostały nałożone żadne ograniczenia, tzn. nie muszą one być najkrótszą ścieżką punktu A do B
- Zbiór walidacyjny – stanowiący około 20% całości, są to ścieżki wyznaczone za pomocą algorytmu Dijkstra.

Opis algorytmów uczenia maszynowego użytych w projekcie

Regresja logistyczna

Opis modelu

Wejściami do modelu regresji logistycznej są kombinacje zerojedynkowe o długości 50 znaków. W czym „0” oznacza brak tego miasta w badanej ścieżce. Oczywiście przewidziane jest również dodatkowe wejście ze stałą jedynką, aby zapewnić dokładność modelu. Funkcja sigmoid użyta w naszym projekcie jest następująca:

$$f(x) = \frac{1}{1 + e^{-z}}$$

Gdzie z oznacza produkt mnożenia wejść do modelu przez poszczególne jego wagi.

$$z = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 \ldots + \beta_{51} * x_{51}$$

Decyzję o przynależności do danej klasy podejmujemy w momencie, gdy wartość predykcji dla danej ścieżki jest powyżej lub poniżej wartości 0.5. Zaś funkcja kosztu wyglądała następująco:

$$J(\theta) = \sum y^{(i)} * \log\left(h\theta\left(x(i)\right)\right) + \left(1 - y^{(i)}\right) * \log\left(1 - h\theta\left(x(i)\right)\right)$$

Gdzie $h\theta\left(x(i)\right)$ oznaczało predykcję dla i-tego zestawu wejść, czyli dla i-tej badanej ścieżki, a $y^{(i)}$ oznaczało prawidłową klasyfikację.

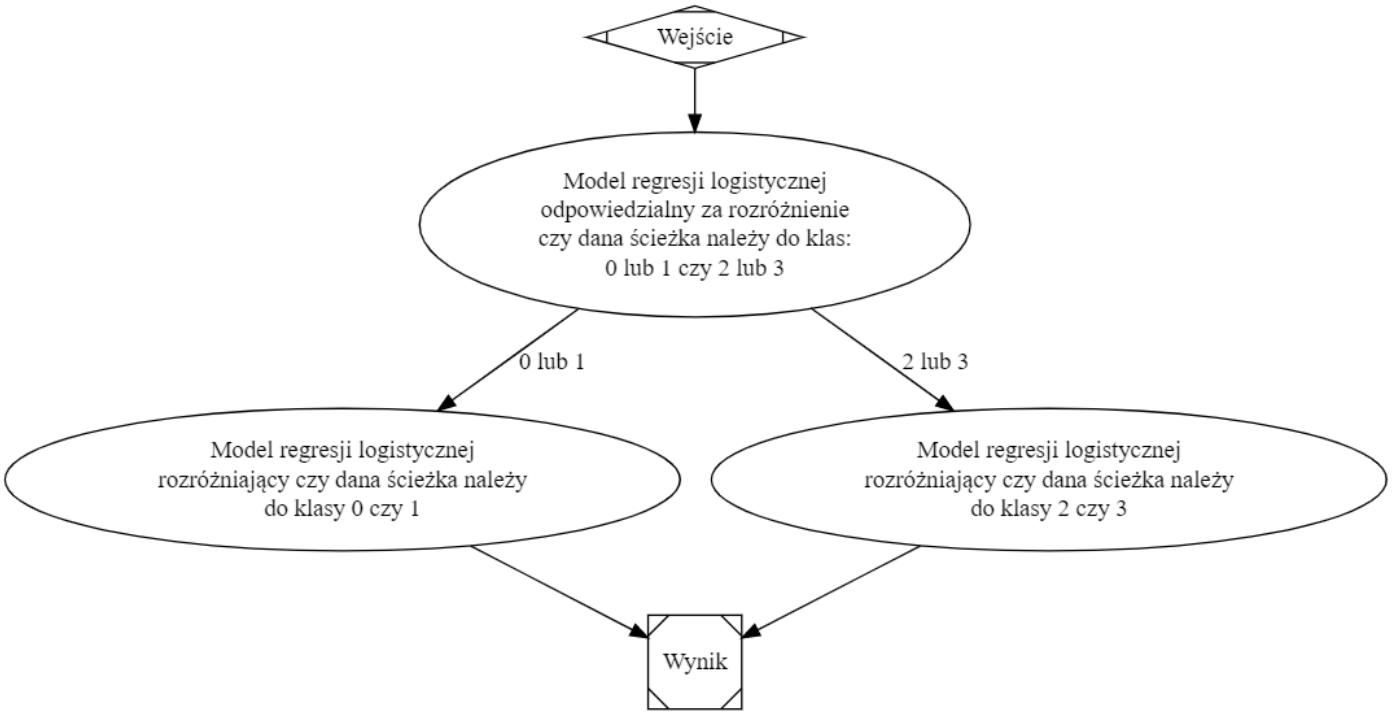
Następnie z funkcji kosztu została wyliczona funkcja gradientu za pomocą której aktualizowane były wagi.

$$\theta j = \theta j - \alpha \sum h\theta\left(x(i)\right) - y^{(i)})xj^{(i)}$$

Gdzie θj oznaczało j-tą wagę modelu, współczynnik α był to krok uczący, a $h\theta\left(x(i)\right) - y^{(i)})xj^{(i)}$ to funkcja gradientu wyliczona na podstawie funkcji kosztu.

Opis algorytmu uczenia

W naszym projekcie użyliśmy w sumie trzech modeli regresji logistycznej.



Klasa	Kodowanie
3	11
2	10
1	01
0	00

Takie rozwiązanie było podyktowane początkowymi próbami z jedynie dwoma modelami regresji logistycznej, które niestety nie dawały zadowalających efektów. W naszym pierwotnym pomyśle chcieliśmy zastosować dwa modele regresji logistycznej, których zadaniem byłoby na podstawie ścieżki ustalenie bitów kodowania danej klasy ścieżki. Modele niestety nie działały dobrze – wielką trudność sprawiało im odróżnienie 3 od 1 i pomimo treningu osiągały bardzo wysokie błędy, w porównaniu do bardzo niskich błędów odróżniania 0 i 2. (mówimy tutaj o różnicach sięgających nawet 8 rzędów wielkości!)

Sądzimy, że takie zachowanie było spowodowane tym, że modelowi ciężko byłoby odnaleźć zasadnicze podobieństwa pomiędzy ścieżkami klasyfikowanymi do klas 0 i 2, jednocześnie odróżniające je od tych klasyfikowanych jako klasa 1 lub 3. Dlatego, aby uniknąć problemów zastosowaliśmy powyższe rozwiązanie, gdzie już nie zauważyliśmy tego zgubnego wpływu bitu parzystości.

Zbiór 10 000 ścieżek (2500 najkrótszych z algorytmu Dijkstry oraz 7500 generowanych losowo) podzieliliśmy na zbiór uczący i walidacyjny w stosunku $8{:}\,2$. Ścieżki były podawane do modelu pojedynczo, zatem również i wagi były aktualizowane każdorazowo po podaniu danych. Zauważyliśmy, że najlepsze wyniki model osiąga z wartością początkową parametru kroku na poziomie 1.75 Jego zwiększanie i zmniejszanie powodowało, że ostateczna dokładność przewidywań drastycznie malała. W momencie, kiedy został już odnaleziony dobry kierunek, w którym powinny podążać zmiany wag zbyt duży krok powodował, że były one zbyt gwałtowne i model nie potrafił osiągnąć optimum. Zaś zbyt mały krok powodował to, że model nigdy nie mógł dotrzeć w jego okolice. Dlatego zastosowaliśmy metodę, w której przy każdej próbce uczącej parametr kroku został mnożony przez wartość 0.999 dzięki czemu krok dostatecznie wolno malał, aby krok nie był zbyt mały i na koniec treningu, kiedy jego wartość spadła już znacząco, pozwalało to na końcową optymalizację.

Następnie po wytrenowaniu modelu na pozostałym zestawie danych sprawdzaliśmy poprawność jego przewidywań. Wyniki jakie osiągnęliśmy były zaskakująco dobre, nawet w porównaniu z siecią neuronową.

Sieć neuronowa

W przypadku sieci neuronowej zastosowaliśmy klasyczny model sieci z 2 warstwami ukrytymi. Za wejście do modelu służy, podobnie do regresji logistycznej, wektor 50 wartości przybierających wartość 1 – jeżeli ścieżka przebiega przez dane miasto lub 0 jeżeli nie przebiega. Przykładowo dla sieci 5 miast $\{0,1,2,3,4\}$ i ścieżki $2 \to 3 \to 0$ byłby to wektor $(1,0,1,1,0)$. Odrzucamy więc informację o kolejności występowania miast w ścieżce, przewidując, że tak prosta konstrukcja sieci i tak nie byłaby w stanie użyć tych informacji skutecznie.

Konstrukcja sieci neuronowej

Sieć neuronowa, jak wspomniano, ma 50 wejść oraz 2 warstwy ukryte. Przyjmują one odpowiednio szerokości 24 oraz 16 neuronów. Wyjście stanowi pojedyncza liczba stanowiąca przewidywaną wartość OSNR dla danej ścieżki.

Operacje wykonywane na poszczególnych warstwach przedstawiają się następująco:

- Warstwa 1: $h1_i = \sum_{t=1}^{50} W1_{it} * x_t + b1_i$
- Warstwa 2: $h2_i = \sum_{t=1}^{24} W2_{it} * \frac{o_t}{h1_t} + b2_i$ – ze względu na spodziewaną odwrotną proporcjonalność spodziewanego wyniku do krawędzi w ścieżce, na drugiej warstwie dokonujemy dzielenia parametru $o2$ przez wynik 1. warstwy. Zabieg ten znacząco poprawił jakość szacowania
- Warstwa 3 (wyjście): $y = \max\left(\sum_{t=1}^{16} h2_t * V_t + a, 0\right)$ – na wyjściu stosujemy rectifier wyniku, odrzucając wszelkie o wartości poniżej zera.

Eksperymenty dawały stosunkowo wysoką jakość wyników nawet przy ograniczaniu wielkości sieci do pojedynczego neuronu na każdej z warstw, co sprowadza problem do quasi-regresji, ale chcąc pozostać przy idei realizacji faktycznej sieci neuronowej pozostaliśmy przy parametrach opisanych powyżej.

Realizacja sieci

Sieć neuronową zrealizowaliśmy z użyciem biblioteki Dynet dla języka C++.

Do uczenia sieci wykorzystaliśmy metodę AdaGrad, będącą odmianą metody stochastycznego gradientu o adaptacyjnej prędkości uczenia/learning rate dla poszczególnych parametrów. Była ona w naszym testowaniu najskuteczniejszą z udostępnianych przez bibliotekę metod, co może wynikać z niskiej gęstości danych w sieci, a co za tym idzie z dużej korzyści z adaptacyjnej prędkości uczenia.

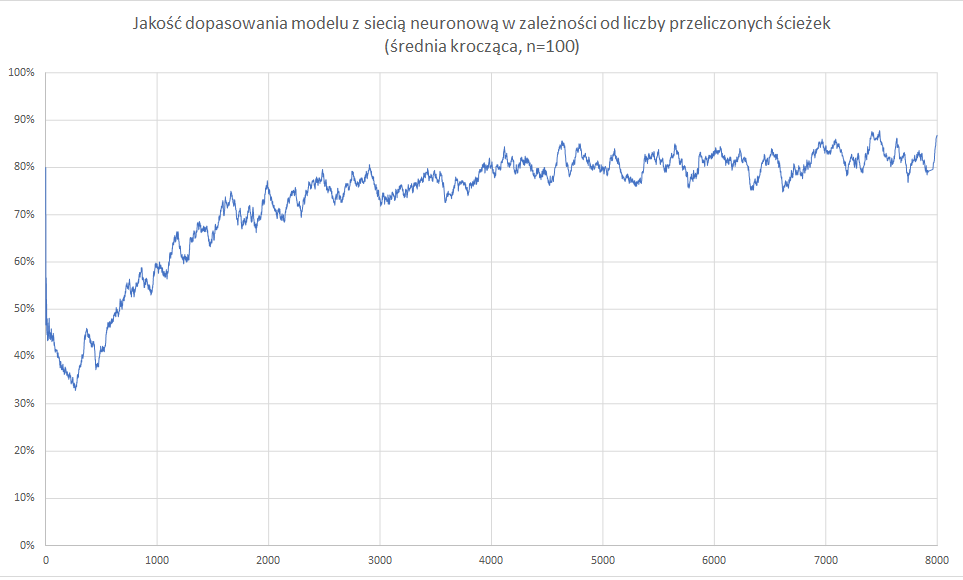
Wartością oczekiwaną na wyjściu z sieci jest wielkość OSNR (wg tabeli ze strony 2.).

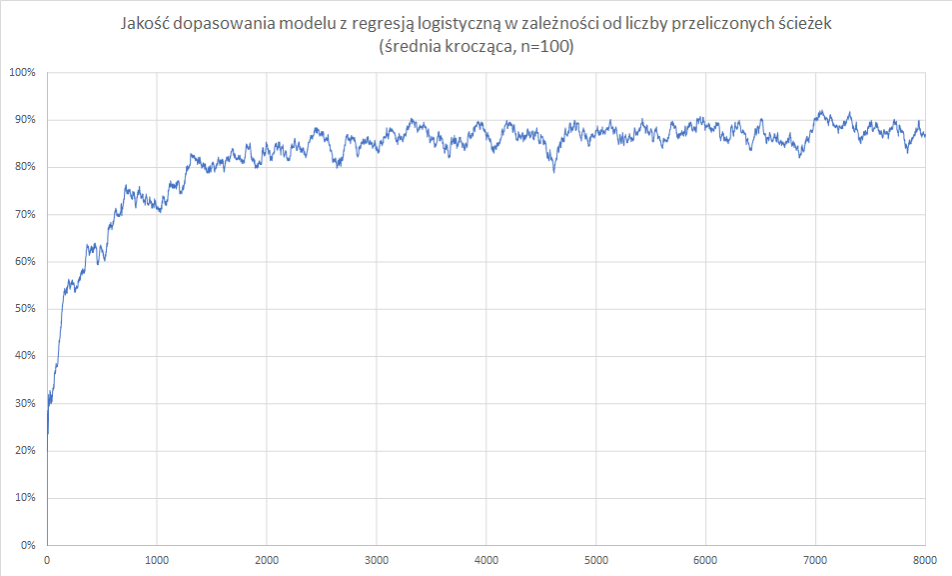
Eksperyment i analiza wyników

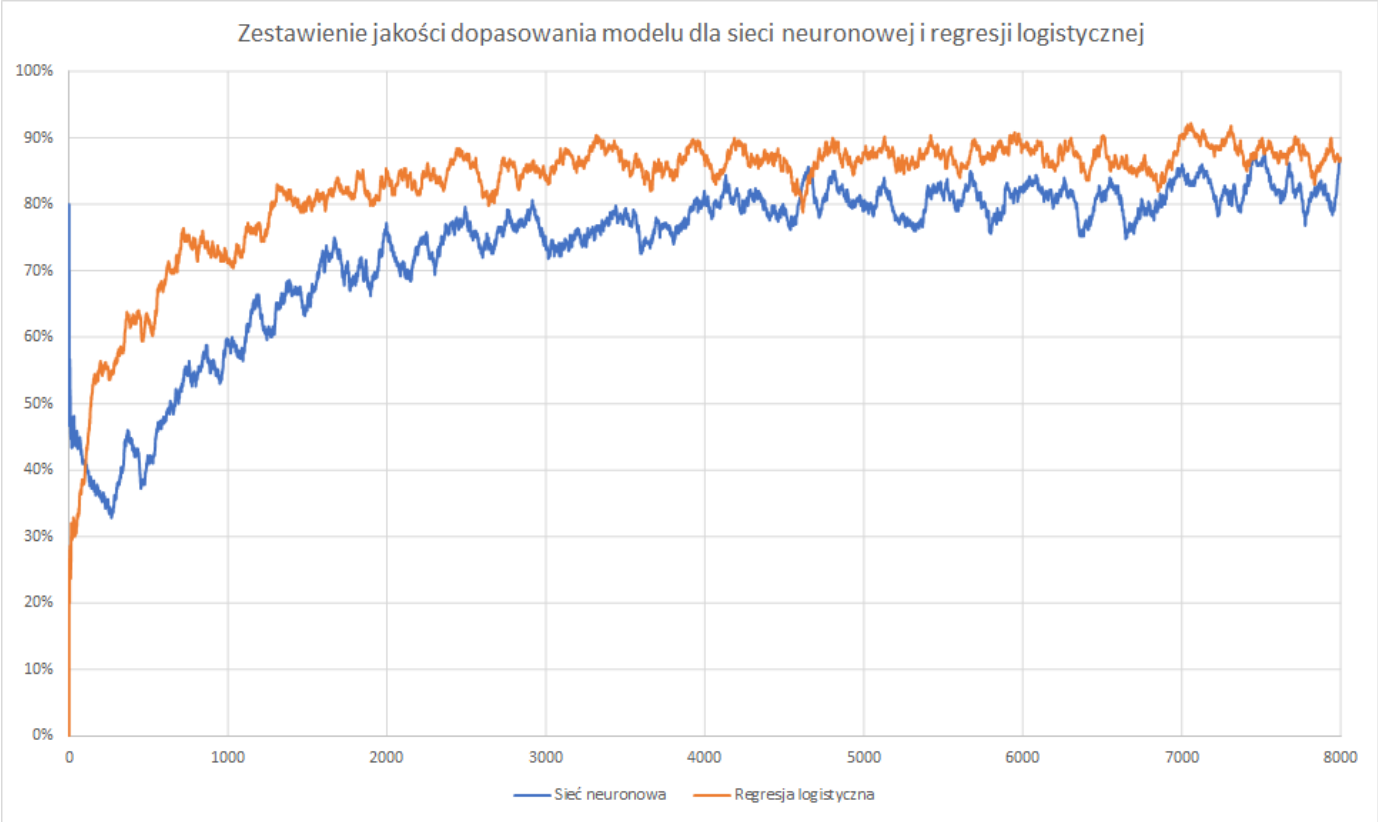
W ramach badania podawaliśmy modelowi regresji i sieci neuronowej te same zestawy różnych ścieżek w tej samej kolejności (wylosowane spośród wszystkich wygenerowanych ścieżek). W ramach pojedynczego eksperymentu podawaliśmy niewytrenowanym modelom po 8 0000 losowych ścieżek, trenując modele po każdej iteracji, a rezultaty na wykresach zostały zagregowane z 5 prób i uśrednione dla ograniczenia wpływu siłą rzeczy losowego charakteru modeli.

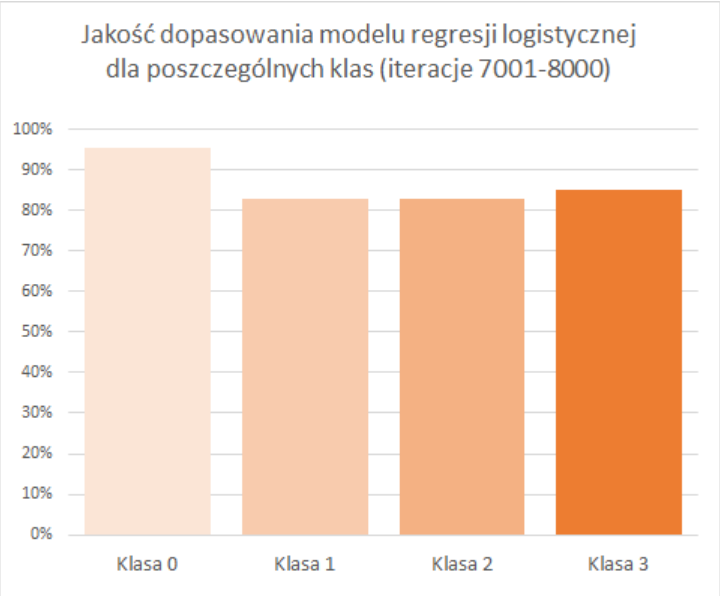
Jako jakość dopasowania rozumiemy odsetek prób w których model zwraca poprawną klasyfikację, tj. w przypadku regresji logistycznej dwubitowa wartość musi odpowiadać oczekiwanej klasie, a w sieci neuronowej zastosowaliśmy następujące kryteria sukcesu:

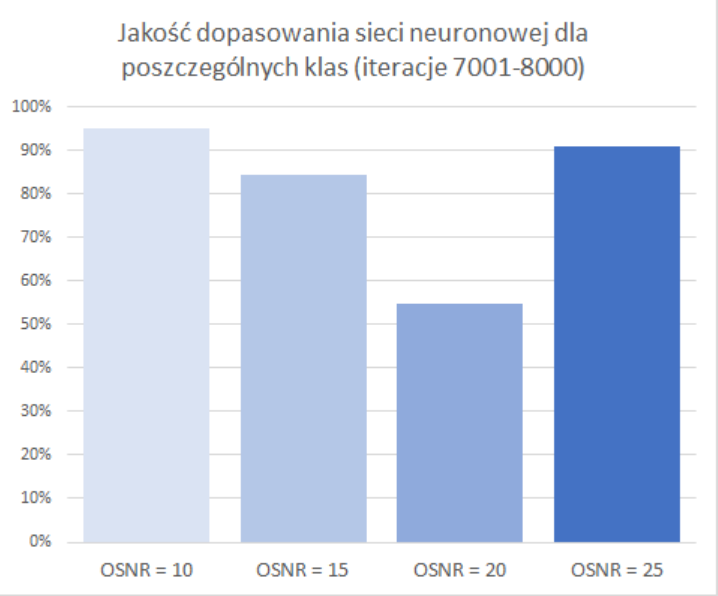
$$\begin{cases} x \leq 12.5 \rightarrow klasa\ 0 \\ 12.5 < x \leq 17.5 \rightarrow klasa\ 1 \\ 17.5 < x \leq 22.5 \rightarrow klasa\ 2 \\ 22.5 < x \rightarrow klasa\ 3 \end{cases}$$




Jakość dopasowania modelu z regresją logistyczną w zależności od liczby przeliczonych ścieżek
(średnia krocząca, n=100)
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
0
1000
2000
3000
4000
5000
6000
7000
8000


Zestawienie jakości dopasowania modelu dla sieci neuronowej i regresji logistycznej
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
0
1000
2000
3000
4000
5000
6000
7000
8000
Sieć neuronowa
Regresja logistyczna


Jakość dopasowania modelu regresji logistycznej
dla poszczególnych klas (iteracje 7001-8000)
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Klasa 0
Klasa 1
Klasa 2
Klasa 3


Jakość dopasowania sieci neuronowej dla
poszczególnych klas (iteracje 7001-8000)
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
OSNR = 10
OSNR = 15
OSNR = 20
OSNR = 25

Analiza wyników

Obserwujemy, że w przypadku naszej definicji i realizacji problemu lepiej spisującą się metodą jest regresja logistyczna. Oferuje ona zarówno bardziej stromą krzywą uczenia, jak i finalnie lepsze wyniki w zakresie skuteczności klasyfikacji.

Podejrzewamy, że może wynikać to ze znaczących uproszczeń, które poczyniliśmy, tj. z nieprzekazywania do sieci neuronowej informacji o kolejności wierzchołków w ścieżce – przez to sieć neuronowa nie ma możliwości „wyuczenia" się długości poszczególnych krawędzi pomiędzy miastami, a zatem realizuje analogiczny dla regresji logistycznej model *regresji* względem listy odwiedzonych miast, minimalizując zasób informacji do, co najwyżej, wiedzy o spodziewanej długości ścieżki wychodzącej z danego miasta. Sieć neuronowa jest zatem wykorzystana do problemu, do którego wystarcza użycie pojedynczej warstwy (a *de facto* regresja logistyczna jest właśnie w ogólności „siecią neuronową" o pojedynczej warstwie). Ze względu na ogrom liczby parametrów podejrzewamy, że wyniki sieci neuronowej są więc zwyczajnie zaszumione.

Porównując jakość dopasowania modeli dla poszczególnych klas obserwujemy, że na krawędziach zbioru sieć neuronowa radzi sobie lepiej, tj. skuteczniej klasyfikuje ścieżki jako przynależne do klas 0 lub 3, ale regresja logistyczna w ogólności zwraca więcej poprawnych wyników, ze względu na wysoką jakość klasyfikacji szczególnie w klasie 2.

Podsumowanie projektu

Realizacja projektu pozwoliła nam zapoznać się w praktyce z metodami uczenia maszynowego – regresję logistyczną implementując samodzielnie, a w przypadku sieci neuronowej korzystając z biblioteki Dynet. Projekt bardzo nam się podobał, bo pozwalał na bezpośrednie obserwowanie poprawy wyników wraz z poprawkami w algorytmach, a także ze względu na bycie opartym na rzeczywistym problemie, operując na danych przynajmniej częściowo mających odzwierciedlenie w rzeczywistości.

Wyniki eksperymentu, tj. lepsze wyniki regresji logistycznej niż sieci neuronowej trochę nas zaskoczyły, szczególnie mając w pamięci artykuły, w których sieci neuronowe w analogicznych problemach spisywały się lepiej, ale wydaje nam się, że ustaliliśmy z czego wynikają te rozbieżności.