[WSI] Kajetan Rożej/Mikołaj Szawerda - Zadanie 5. (Sieci Neuronowe)

1. Implementowane algorytmy

Głównym celem zadania 5. była implementacja perceptronu wielowarstwowego (ang. MLP). Jest to najpopularniejszy typ sztucznych sieci neuronowych. Sieć tego typu składa się zwykle z jednej warstwy wejściowej, kilku warstw ukrytych oraz jednej warstwy wyjściowej. Neurony warstw ukrytych posiadają wiele wejść i jedno wyjście, a jego wartość obliczana jest w następujący sposób:

1. $s = \sum_{i=1}^{n} x_{j,i}^k w_{j,i}^k + b_j^k$, gdzie (j,k) oznacza j. neuron w k. warstwie
2. $s = f(s)$, gdzie s to wybrana funkcja aktywacji

W naszej sieci jako funkcję aktywacji zdecydowaliśmy się przyjąć gaussian ($f(x) = exp(-x^2)$), ze względu na dobre właściwości przy zastosowaniu sieci neuronowej jako aproksymatora uniwersalnego.

Trenowanie sieci polega na poszukiwaniu takiego zestawu wag i biasów, które pozwolą na jak najlepszą estymację zadanej funkcji. W tym celu wykorzystywać można metodę propagacji wstecznej, która to rozkłada uzyskany błąd na poszczególne neurony, wskazując kierunek, w którym w danej iteracji należy zmodyfikować wagi w celu zmniejszenia popełnionego błędu. Tempo modyfikacji wag określone jest natomiast za pomocą współczynnika uczenia. Wykorzystanie tej metody w połączeniu z algorytmem gradientowym jest jednym z najpopularniejszych sposobów na znajdowanie wag sieci.

W związku z wymaganiami zaimplementowaliśmy również poszukiwanie wag przy pomocy algorytmu ewolucyjnego. Po konsultacji z prowadzącym, aby upodobnić sposób ten do metody gradientowej i skrócić czas wykonania zdecydowaliśmy się na wykorzystanie populacji liczącej jednego osobnika.

2. Zaplanowane eksperymenty

Do eksperymentów użyliśmy perceptronu z dwiema warstwami ukrytymi. Głównym zadaniem było zbadanie wpływu ilości neuronów w poszczególnych warstwach i użytego algorytmu (gradientowego lub ewolucyjnego) na przebieg procesu uczenia i jakość estymacji zadanej funkcji:

$$f(x) = x^2 sin(x) + 100\ sin(x) cos(x)$$

Po wstępnym rozpoznaniu zdecydowaliśmy się na następujące ilości neuronów:

- $2, 2$
- $5, 5$
- $10, 10$
- $20, 20$

Oznaczenie (5,5) oznacza, że zarówno pierwsza jak i druga warstawa ukryta mają po 5 neuronów.

Aby uzyskać pełniejszy obraz jakości poszczególnych rozwiązań, zmierzyliśmy również czas potrzebny na wytrenowanie poszczególnych wariantów sieci.

W celu poprawy własności optymalizacyjnych funkcji celu, przeskalowaliśmy x i $f(x)$ do $\langle -1, 1 \rangle \times \langle 0, 1 \rangle$

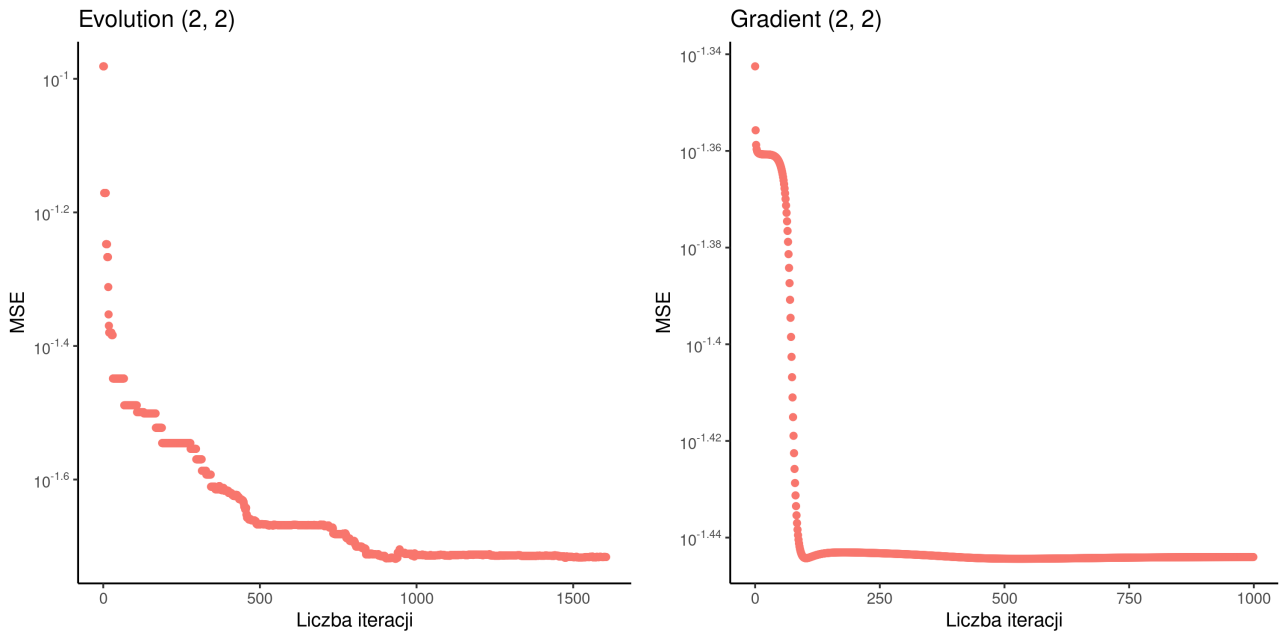
Zbiór danych stanowiło 1000 równomierenie rozłożonych punktów z przedziału $\langle -1, 1 \rangle$ dla których obliczono dokładne wartości zadanej funkcji. Następnie w sposób losowy wybrane zostały próbki do zbiotu uczącego i testującego odpowiednio w stosunku 7:3.

W przypadku algorytmu ewolucyjnego ewaluacja rozwiązania przeprowadzana była dla wszystkich próbek (700 wartości), natomiast na potrzeby algorytmu gradientowego epoch o wielkości 700 został podzielony na taką ilość batchy, dla której uzyskiwano najlepsze rezultaty.

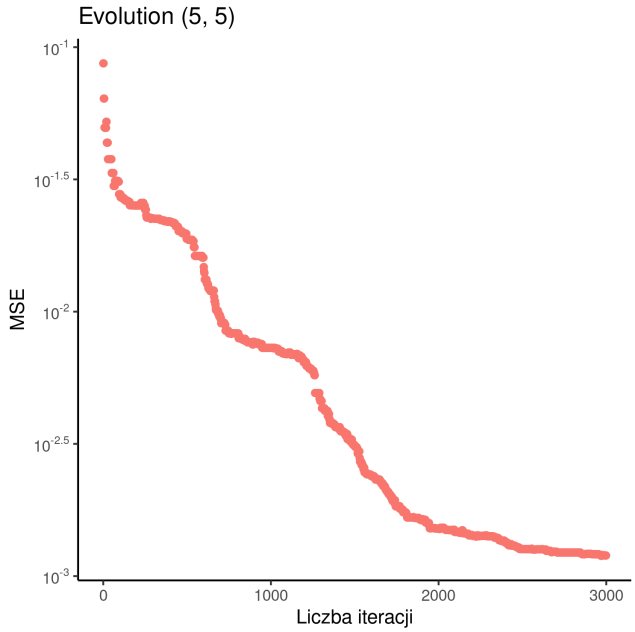
Aby uzyskać miarodajne wyniki związane z wykorzystaniem losowości do inicjalizacji wektora wag (metoda gradientowa), jak i ogólnym działaniem algorytmu (metoda ewolucyjna), każdy eksperyment został uruchomiony kilka razy a do analizy użyto najlepszą z uzyskanych prób.

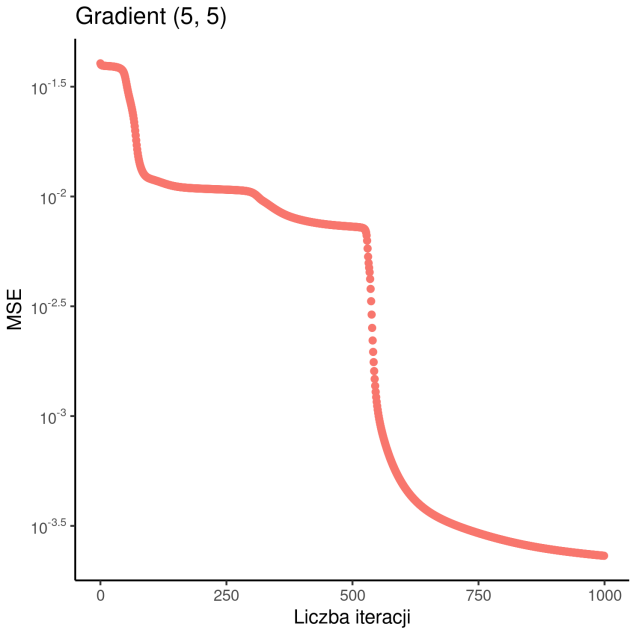
3. Uzyskane rezultaty

Wykres wartości MSE w poszczegolnych przypadkach:

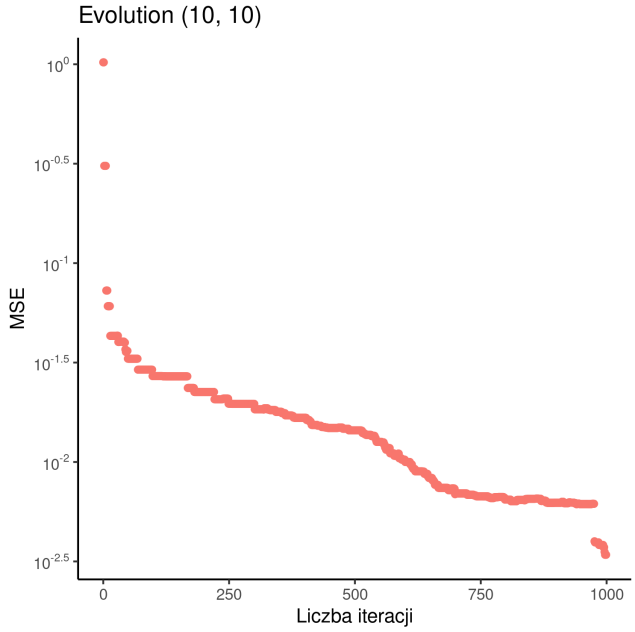


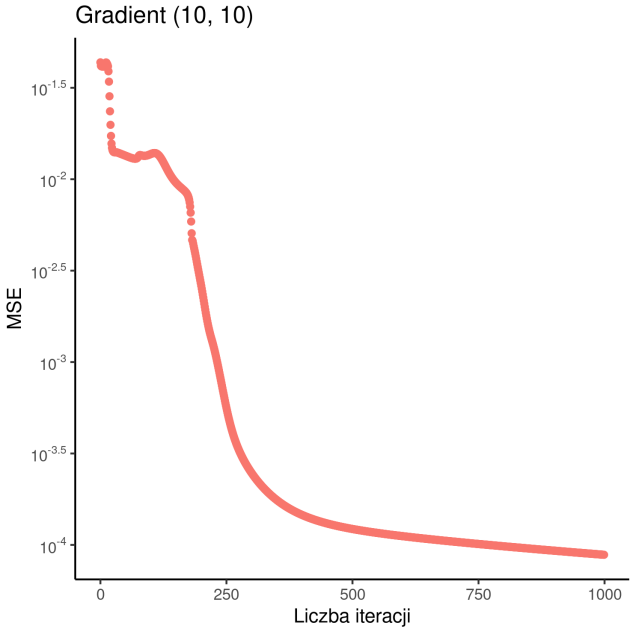
Można zauważyć, że gradient dużo szybciej znalazł optimum, lecz algorytm ewolucyjny lepiej zeksplorował i zeksploatował funckję celu.



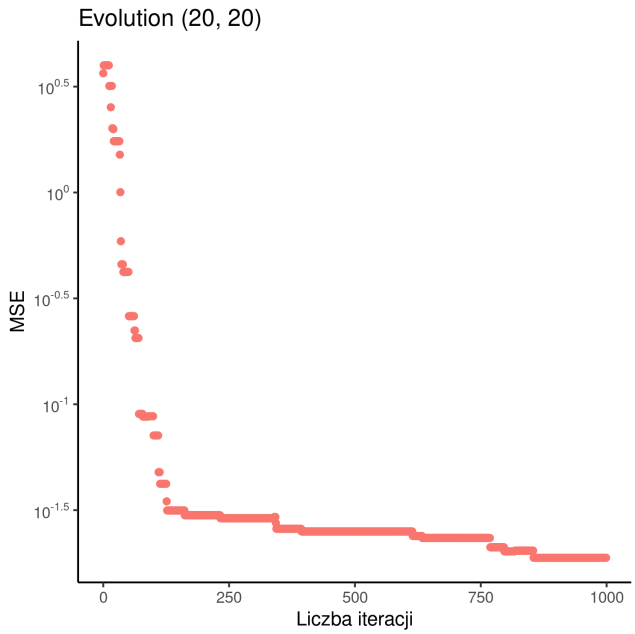


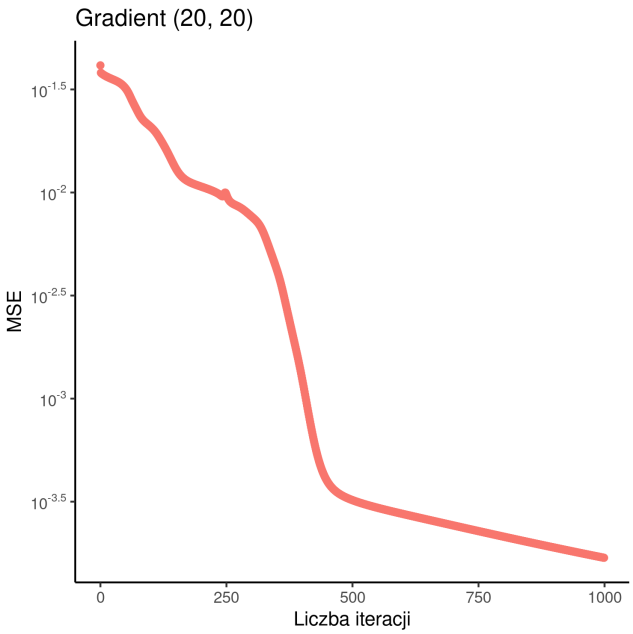
MSE dla algorytmu ewolucyjnego spadało bardziej równomiernie, niż dla algorytmu z użyciem gradientu. W tym przypadku gradient osiągnął rezultaty lepsze o rząd wielkości.





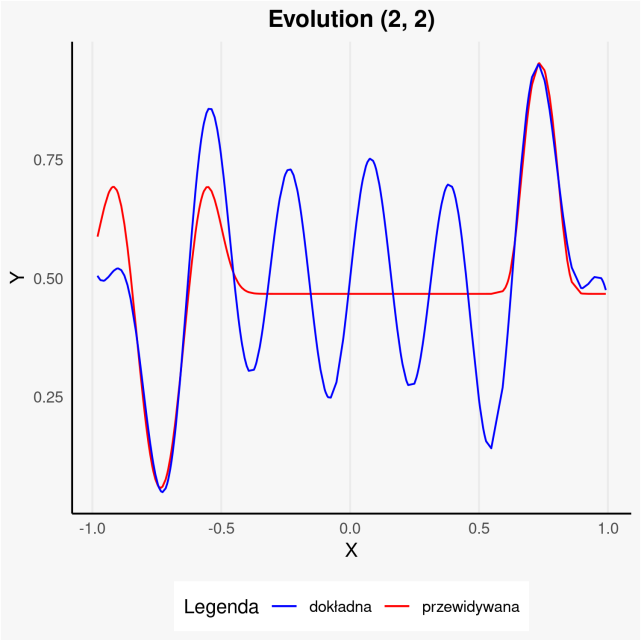
Algorytm gradientu w tym przypadku szybko znalazł otoczenie minimum i osiągnął rezultaty lepsze o rząd wielkości.

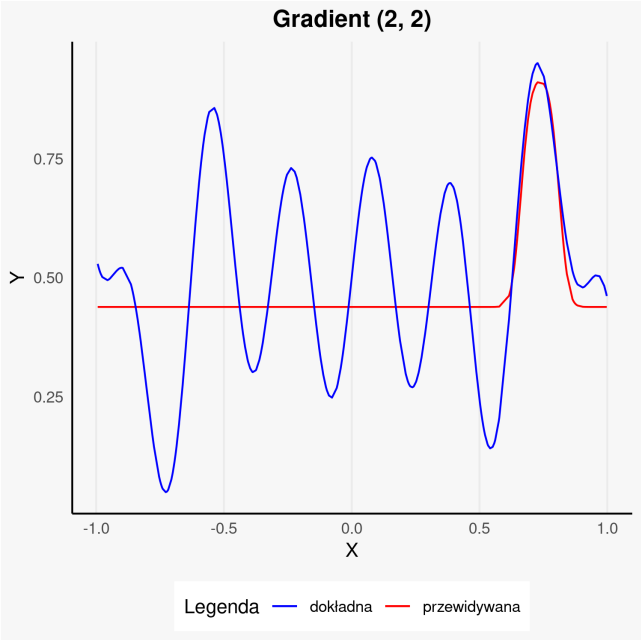




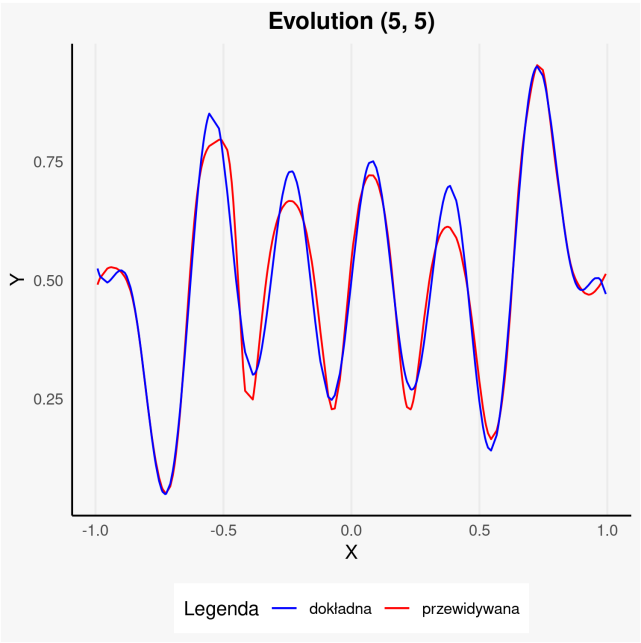
W przypadku wielu wymiarów algorytm ewolucyjny nie był w stanie wyjść z płaskowyżu nie poprawiał swojego rezultatu w kolejnych iteracjach. Algorytm z użyciem gradientu osiągnał rezulaty lepsze o dwa rzędy wielkości.

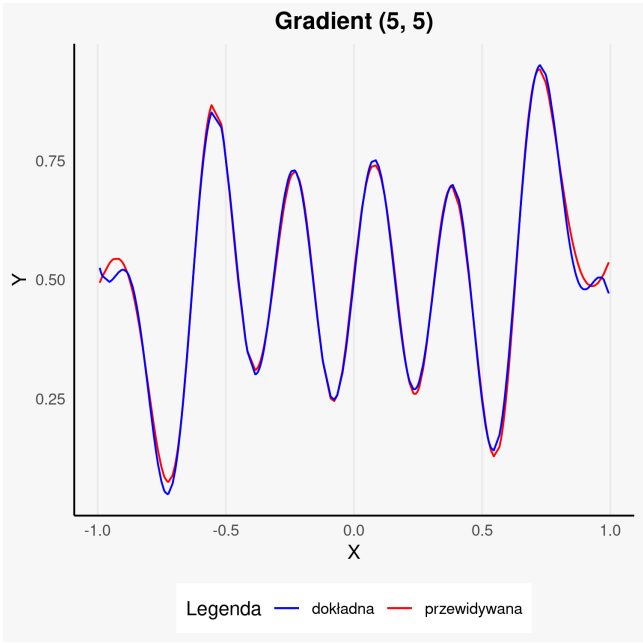
Wykres porównujacy jakość estymacji w porównaniu z funkcją wejściową:



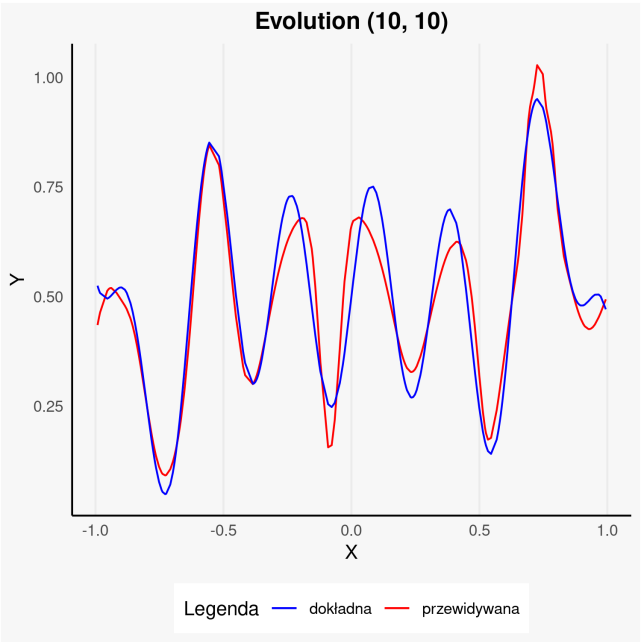


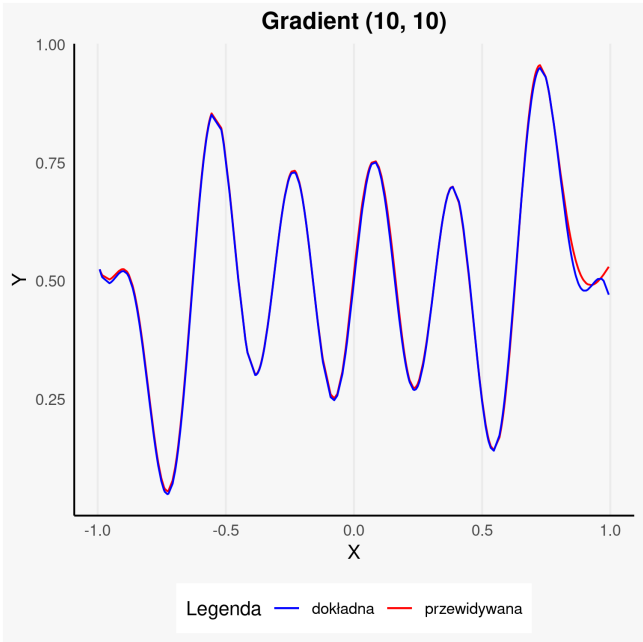
Oba algorytmy nie były w stanie dopasować się do kształtu funkcji - ilość wymiarów była niewystarczająca.



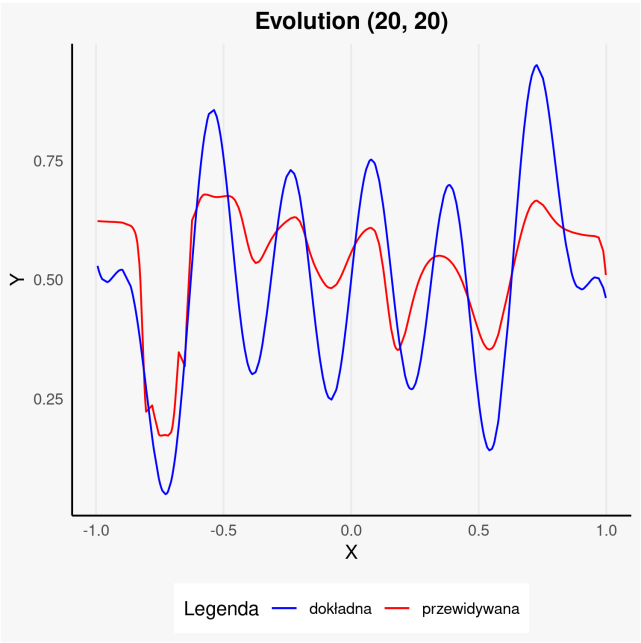


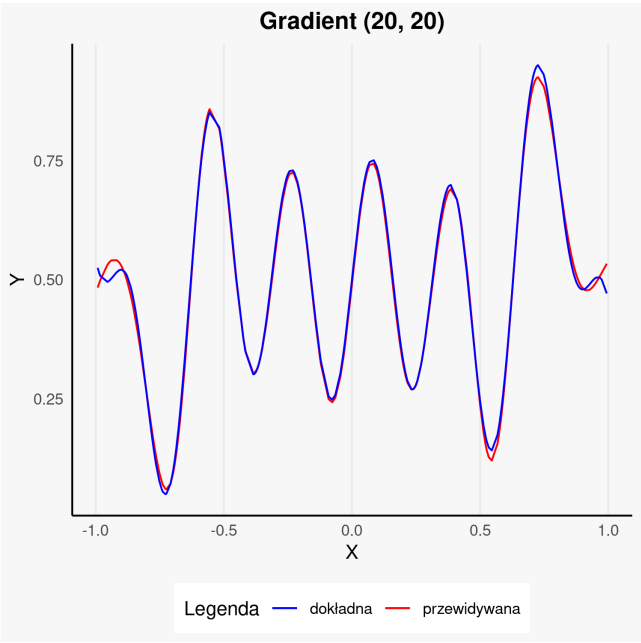
W przypadku warstw (5, 5) każdy z algorytmów przyjął kształt zbliżony do funkcji celu - algorytm z gradientem zrobił to lepiej





Dla warstw (10, 10) algorytm gradientu osiągnął niemalże idealne dopasowanie

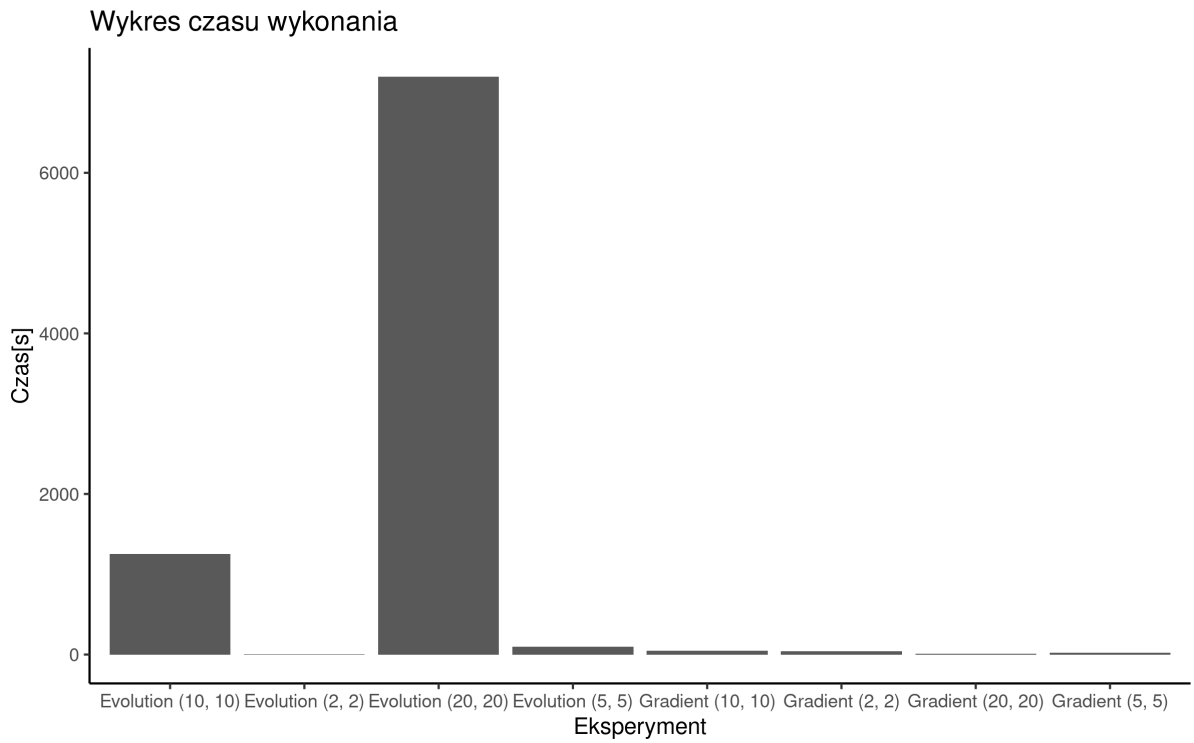




Oba algorytmy poradziły sobie gorzej dla większej ilości neuronów - ewolucyjny w szczególności - pojawiło się zjawisko zwiększonej wymiarowości zadania

Porównując rezultaty pomiędzy ilością neuronów w warstwie, można zauważyć zjawisko nasycenia, gdzie niewystarczająca ilość neuronów prowadzi do wolnego zmniejszania funkcji celu, natomiast zbyt duża ilość znacząco zwiększa wymiarowość zadania, przez co algorytmy są podatne na pozostanie w niezadowalających optimach lokalnych.

Wykres porównujący czas wykonania poszczególnych warintów:



Algorytmy sieci neuronowej z użyciem algorytmu ewolucyjnego wykonywały się o rzędzy wielkości dłużej od z użyciem gradientu. Jest to spowodowane koniecznością reprezentacji sieci neuronowej w postaci wektora(w celu skorzystania z solvera) i jej późniejszym transformacjom do postaci macierzowej

Podsumowanie wyników		
Nazwa	Czas wykonania	MSE
Gradient (2, 2)	38.11	0.03597
Evolution (2, 2)	4.69	0.01923
Gradient (5, 5)	21.26	0.00023
Evolution (5, 5)	94.50	0.00120
Gradient (10, 10)	47.19	0.00009
Evolution (10, 10)	1.25K	0.00342
Gradient (20, 20)	8.82	0.00017
Evolution (20, 20)	7.20K	0.01884

4. Wnioski

Porównując otrzymane rezultaty pod względem ilości neuronów, można wysnuć następujące wnioski:

- ilość neuronów musi być adekwatna do problemu - ich niewystarczająca ilość nie pozwala na właściwe oddanie kształtu funkcji, natomiast zbyt duża ilość zwiększa wymiarowość zadania optymalizacyjnego, utrudniając znaleznie optimum globalnego
- czas trenowania sieci rośnie wraz z ilością neuronów w warstwie

Porównując otrzymane rezultaty pod względem użytego algorytmy optymalizacyjnego, można wysnuć następujące wnioski:

- algorytm gradientu wraz z zastosowaniem liczenia gradientu błędu przez propagację wstęczną pozwala na osiągnięcie lepszych rezultatów od algorytmu ewolucyjnego
- ponieważ alogrytm ewolucyjny wymaga dużej ilości ewaluacji funkcji celu wykonuje się dłużej
- implementacja sieci neuronowej przy użyciu algorytmu ewolucyjnego jest prostsza - nie wymaga stosowania zaawansowanych zależności analitycznych