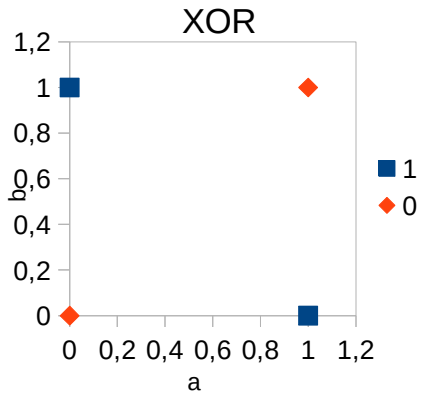
Marcin Nasiłowski, Bartosz Ruszniak
Projekt 2B XOR

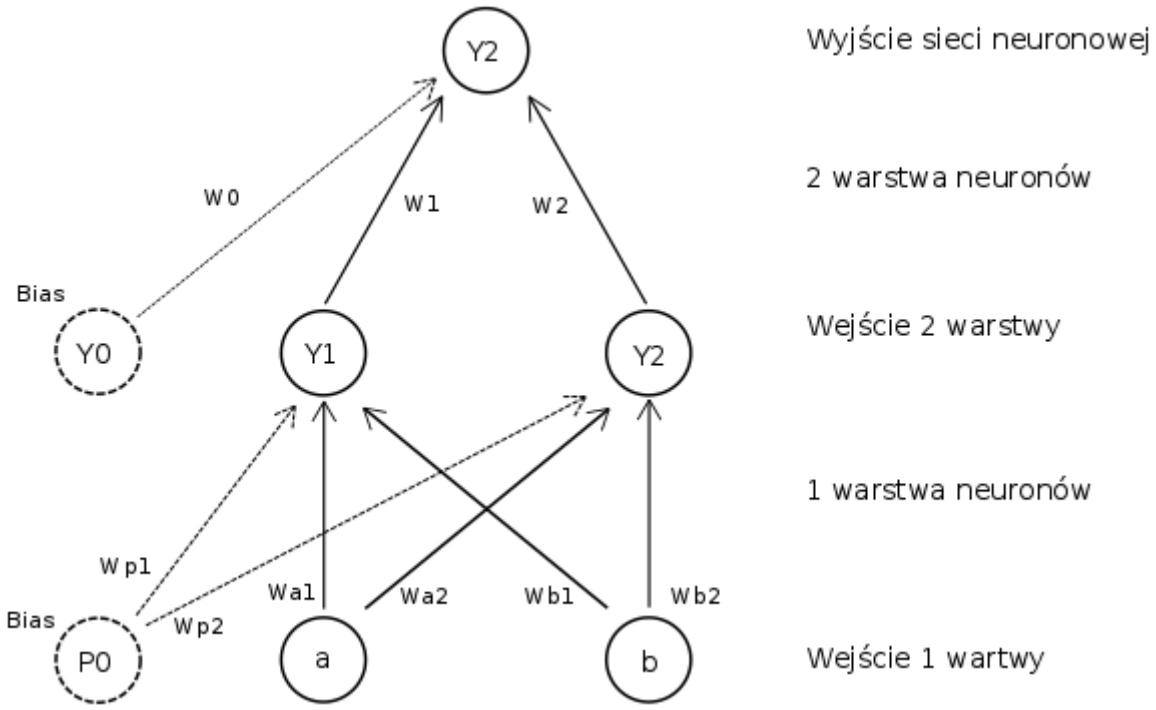
Celem projektu jest zaprojektowanie sieci neuronowej która była by w stanie przypisać punktom (a,b) wartości 1 lub 0 zgodnie z operacją logiczną alternatywa wykluczająca dalej w skrócie nazywaną XOR.

a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0



Jak widać na rysunku 1 nie jest możliwy podział płaszczyzny jedną linią tak by w jednej części znalazły się punkty o wartości 1 a w drugiej o wartości 0. W związku z tym proponowana sieć musi być przynajmniej dwu warstwowa.

Architektura proponowanej sieci do rozwiązania powyższego problemu ma postać:



Gdzie:

a, b	są wejściem sieci neuronowej i mogą przyjmować wartości 0 lub 1?
Wa1, Wa2 …	są wagami odpowiadającym poszczególnym neuronom . Przed uczeniem wagi przyjmowały losową wartość od -1 do 1
P0 i Y1	są biasem odpowiednio pierwszej i drugiej warstwy

Y1, Y2 są wyjściem pierwszej warstwy i jednocześnie wejściem dla warstwy 2 i wynoszą odpowiednio:

$Y01=\frac{1}{1-\exp(-\beta x_1)}$	gdzie	$x_1=Wp1\cdot P0+Wa1\cdot a+Wb1\cdot b$
$Y02=\frac{1}{1-\exp(-\beta x_2)}$	gdzie	$x_2=Wp2\cdot P0+Wa2\cdot a+Wb2\cdot b$
$Y=\frac{1}{1-\exp(-\beta x)}$	gdzie	$x=W0\cdot Y0+W1\cdot Y1+W2\cdot Y2$

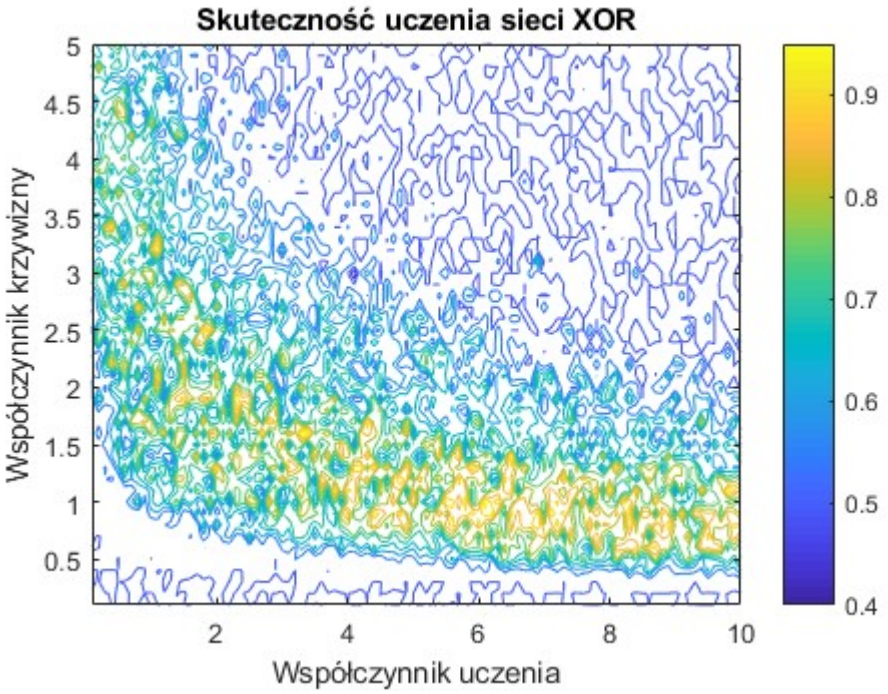
Dane wejściowe do sieci mają następującą współrzędne:

[(1,0);(0,1),;(1,1); (0,0)]

I odpowiadają wartości wyjściowe:

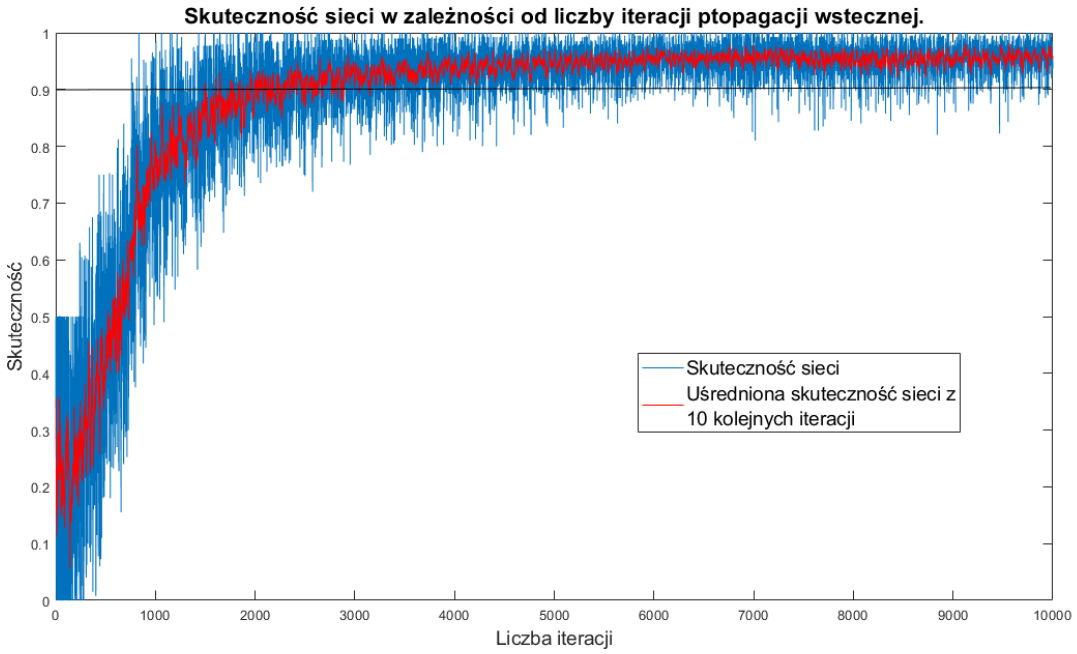
[1;1;0;0]

Dobór współczynnika uczenia i krzywizny.

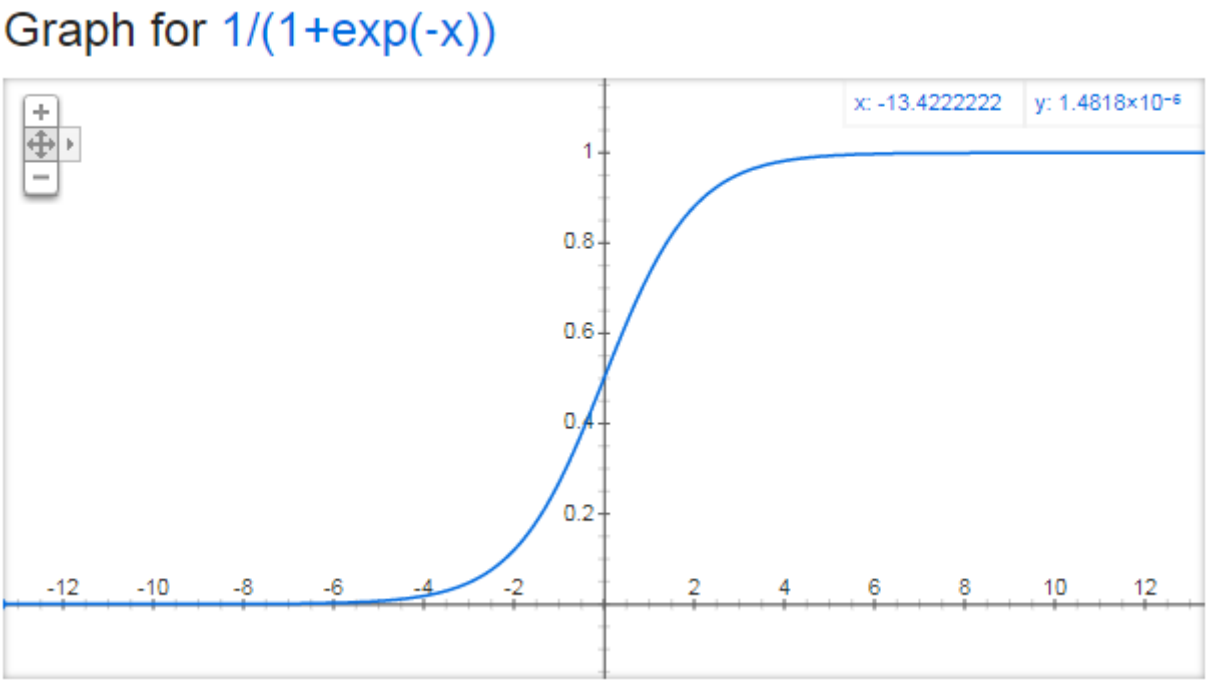


Współczynnik krzywizny wstępnie przyjęto równy 1 natomiast współczynnik uczenia równy 6 Ze względu na dużą liczbę obliczeń, współczynnik uczenia i współczynnik krzywizny badane były przy liczbie iteracji wynoszącej 3000.

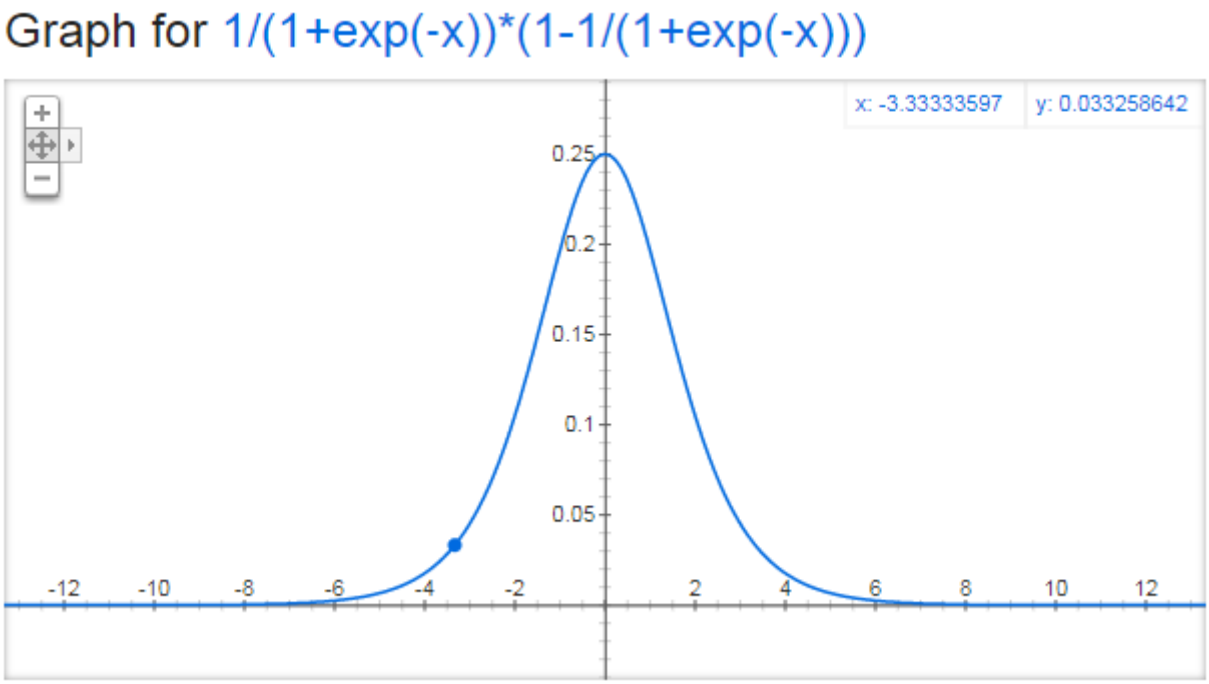
Liczbę iteracji propagacji wstecznej ustalono na 10000 liczba ta jest na tyle duża by skuteczność uczenia nie spadała poniżej 80% i na tyle mała by obliczenia numeryczne nie stanowiły problemu dla komputera.



Funkcja aktywacji

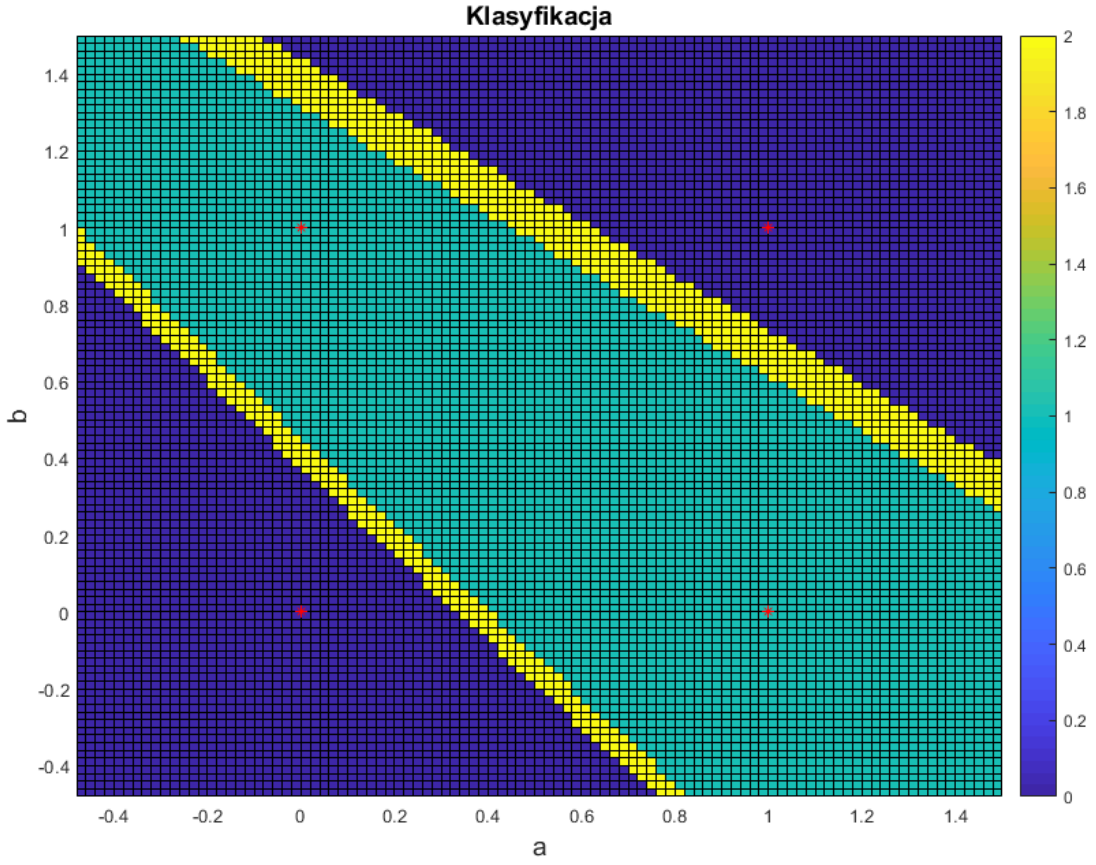


Pochodna funkcji aktywacji:

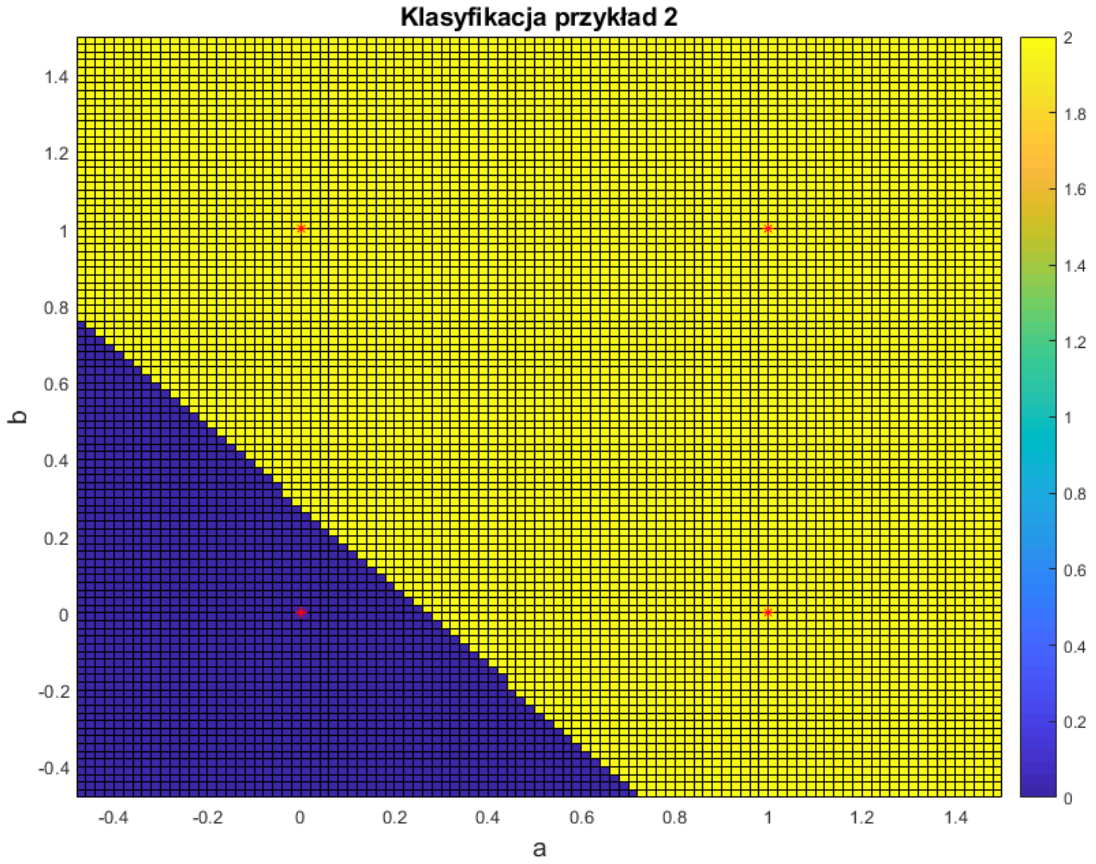


Podział na klasy sporządzono następujący:
0 dla wyjścia z sieci mniejszego niż 0.25
1 dla wyjścia z sieci większego niż 0.75
2 dla pozostałych wartości.

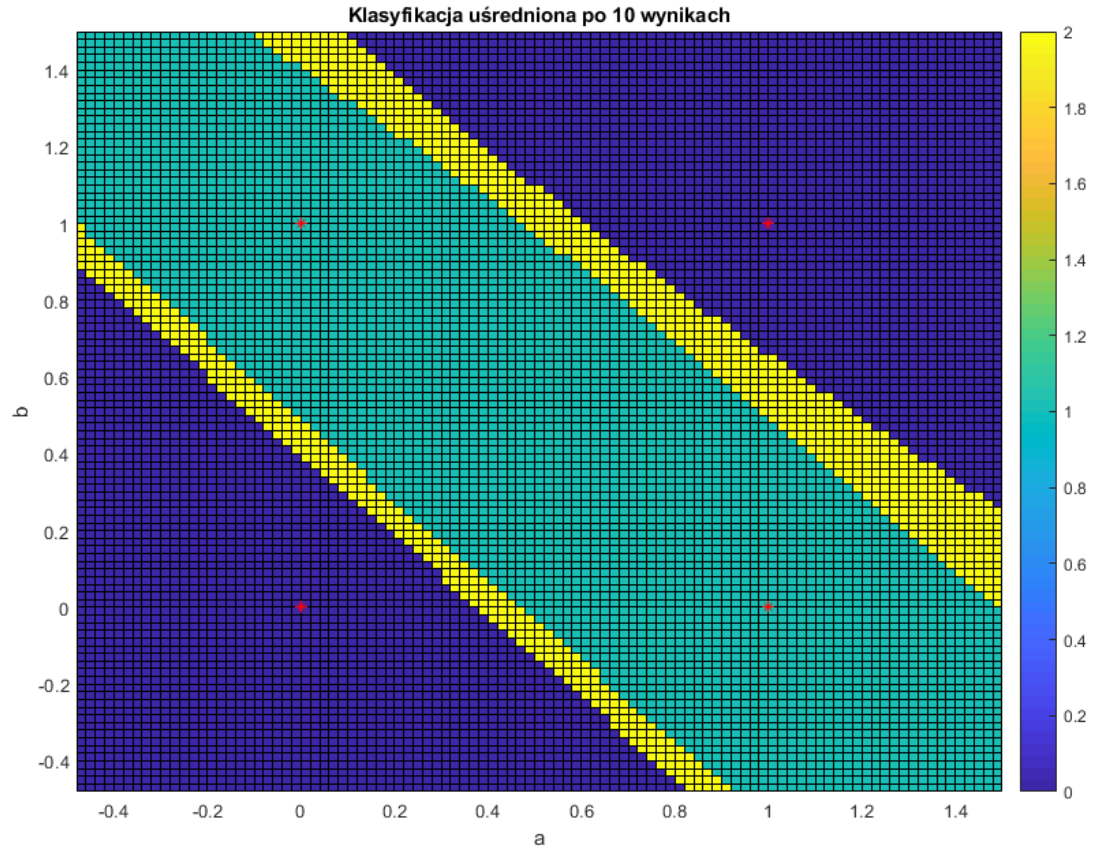
Podział przestrzeni na klasy wygląda następująco:



Ewentualnie



Po uśrednieniu 10 przypadków:



Jak poprawić skuteczność sieci:
1. Kroswalidacja: funkcji uczącej sieć dodać dane sprawdzające i wyznaczyć minimalną skuteczność jaką sieć może osiągnąć. Przy 10000 iteracjach propagacji wstecznej sytuacja taka powinna się zdarzyć nie częściej niż w 15%. Algorytm ten powinien zwiększyć skuteczność sieci w zadanych 4 punktach do 100% czasami zwiększając czas uczenia 2 krotnie.
2. Jeżeli nie chcemy zmieniać funkcji uczenia a chcemy zwiększyć skuteczność uczenia, możemy uśrednić wyjścia sieci uzyskane po około 10 różnych procesach uczenia. Pomysł drugi wydłużył by czas oczekiwania na wynik około 10 razy. Jak widać algorytm z uczeniem byłby skuteczniejszy.