ELEMENTY SZTUCZNEJ INTELIGENCJI

Sztuczne sieci neuronowe

Plan

- Wzorce biologiczne.
- Idea SSN - model sztucznego neuronu.
- Perceptron prosty i jego uczenie regułą delta
- Perceptron wielowarstwowy i jego uczenie metodą wstecznej propagacji błędów
- Zastosowania SSN.

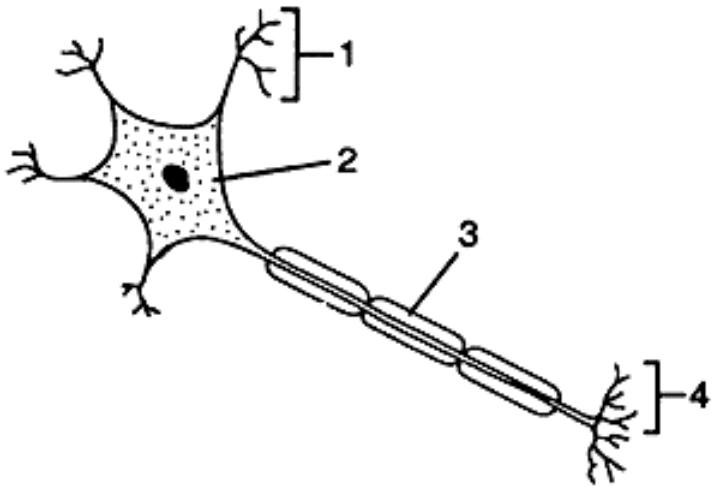

400X
Neuron

Mózg – dane

- Mózg składa się z elementarnych komórek nerwowych – neuronów.
- Ilość neuronów w mózgu to szacunkowo 10 miliardów.
- Każdy neuron:
  - połączony jest z dużą liczbą komponentów (10^4).
  - przeprowadza relatywnie proste obliczenia (ich natura jest niejasna).
- Przez neurony połączone w sieć przechodzą impulsy elektryczne z różną częstotliwością od 1 do 100 Hz i o różnej amplitudzie.
- Mózg jest skuteczny dzięki masowemu zrównolegleniu.

Komórka nerwowa

- Neuron składa się z:
  1. Wielu dendrytów, których celem jest pobieranie impulsów z innych neuronów.
  2. Ciała komórki z jądrem.
  3. Jednego aksonu, który przekazuje impuls dalej.
  4. Synaps – neuroprzekaźników osłabiających lub wzmacniających sygnał.



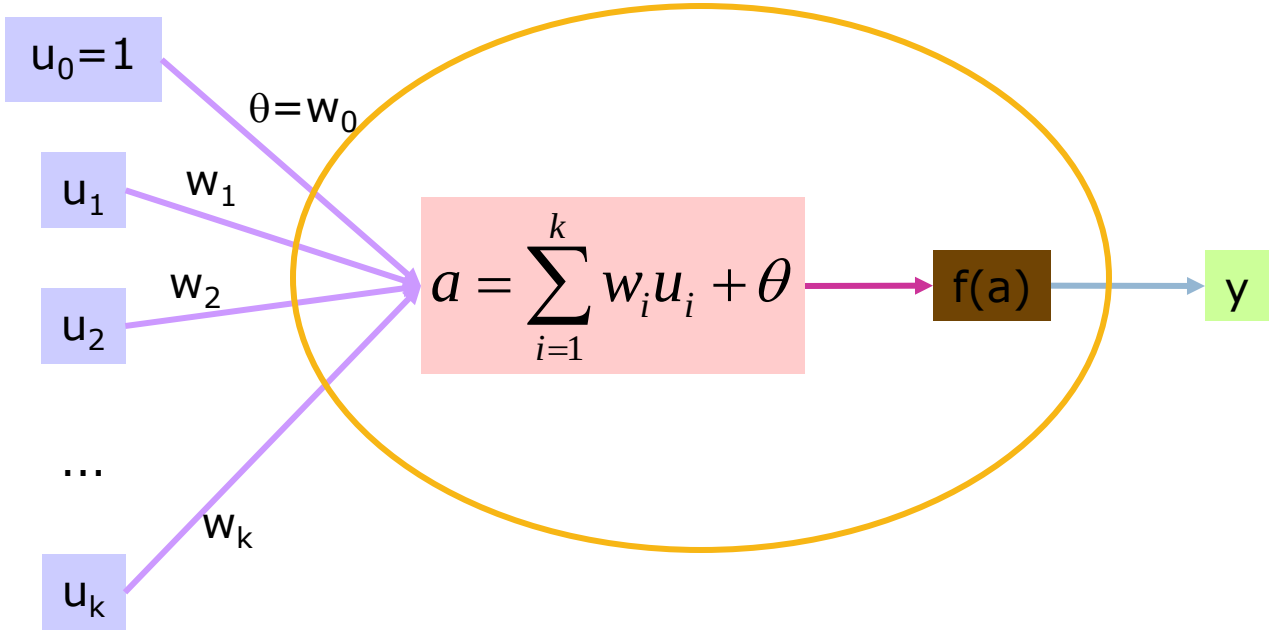
Sztuczne sieci neuronowe – idea

- Uproszczony model mózgu.
- Składa się z pewnej liczby elementów przetwarzających informację (sztucznych neuronów).
- Elementy te są prymitywną imitacją neuronu biologicznego.
- Neurony są połączone poprzez powiązania o nadanych parametrach zwanych wagami, które są modyfikowane w trakcie uczenia.
- Topologia połączeń i wartości parametrów to architektura sieci.
- Rozwiązaniem sieci neuronowej są wartości (sygnały) pojawiające się na wyjściu dla zadanych wartości wejściowych.

Model sztucznego neuronu McCulloch-Pitts'a

- 1943 – pierwszy matematyczny opis neuronu.
- Najprościej neuron można sobie wyobrazić jako przetwornik, który pobiera informacje ze wszystkich, tzw. wejść i na ich podstawie emituje sygnał wyjściowy.
- Każde wejście jest mnożone przez pewną wartość zwana wagą (wzmocnienie lub osłabienie sygnału).
- Sygnały wejściowe są sumowane, by następnie dopasować odpowiedź za pomocą funkcji aktywacji (przejścia neuronu).

Graficzna prezentacja modelu neuronu



Funkcja aktywacji

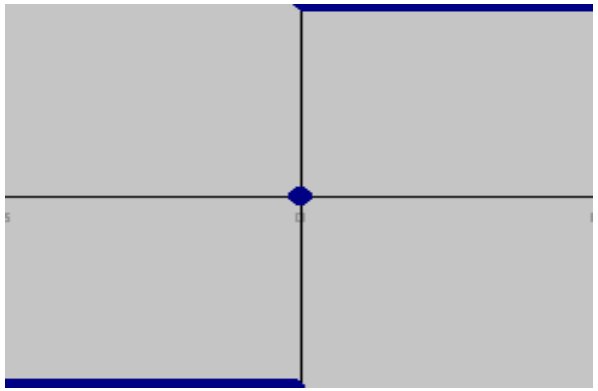
- Zachowanie neuronu jest silnie uzależnione od rodzaju funkcji aktywacji.
- Typy funkcji aktywacji
  - nieciągłe:
    - progowa,
    - signum,
  - ciągłe:
    - liniowa,
    - sigmoidalna
    - gaussa.

Funkcja progowa (unipolarna)

$$f(a) = \begin{cases} 1 & gdy \quad a \geq 0 \\ 0 & gdy \quad a < 0 \end{cases}$$

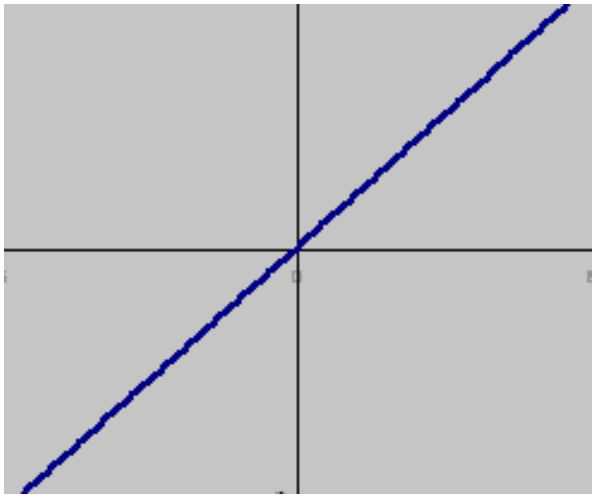
Funkcja signum (bipolarna)

$$f(a)=\begin{cases}1 & gdy \quad a>0\\ 0 & gdy \quad a=0\\ -1 & gdy \quad a<0\end{cases}$$



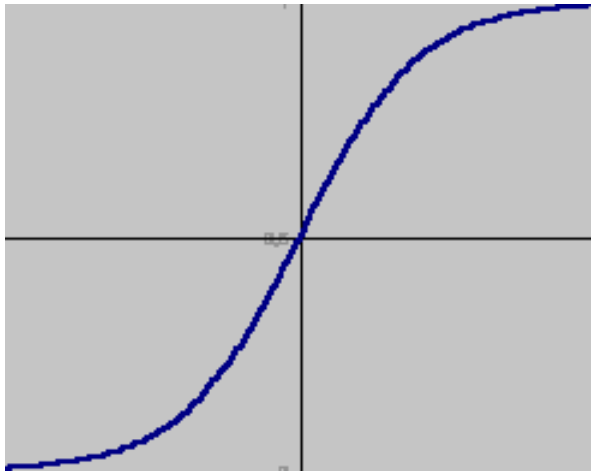
Funkcja liniowa

$$f(a) = a$$



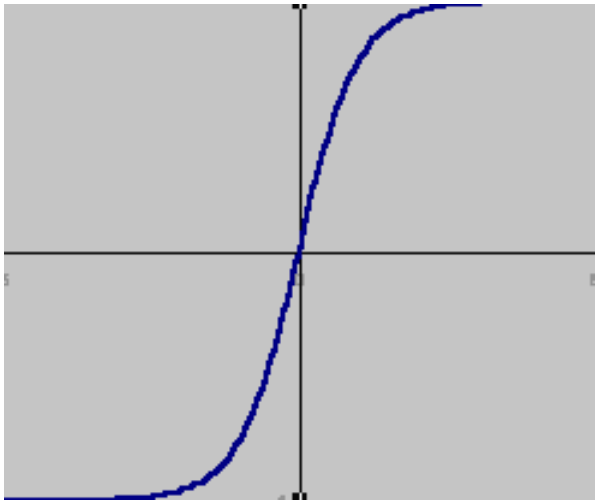
Funkcja sigmoidalna

$$f(a) = \frac{1}{1 + e^{\beta a}}$$



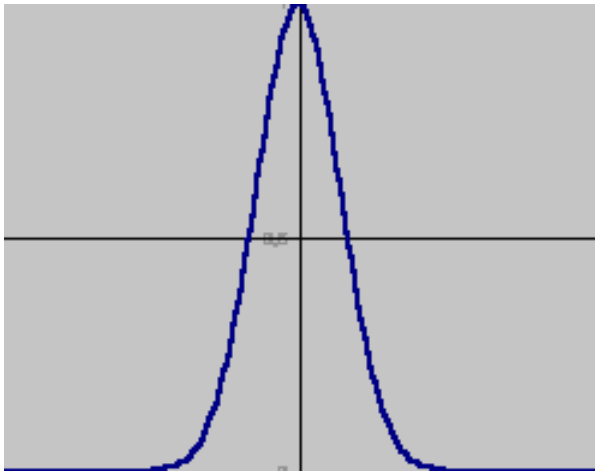
Tangens hiperboliczny

$$f(a) = \frac{e^{\beta a} - e^{-\beta a}}{e^{\beta a} + e^{-\beta a}}$$



Funkcja Gaussa

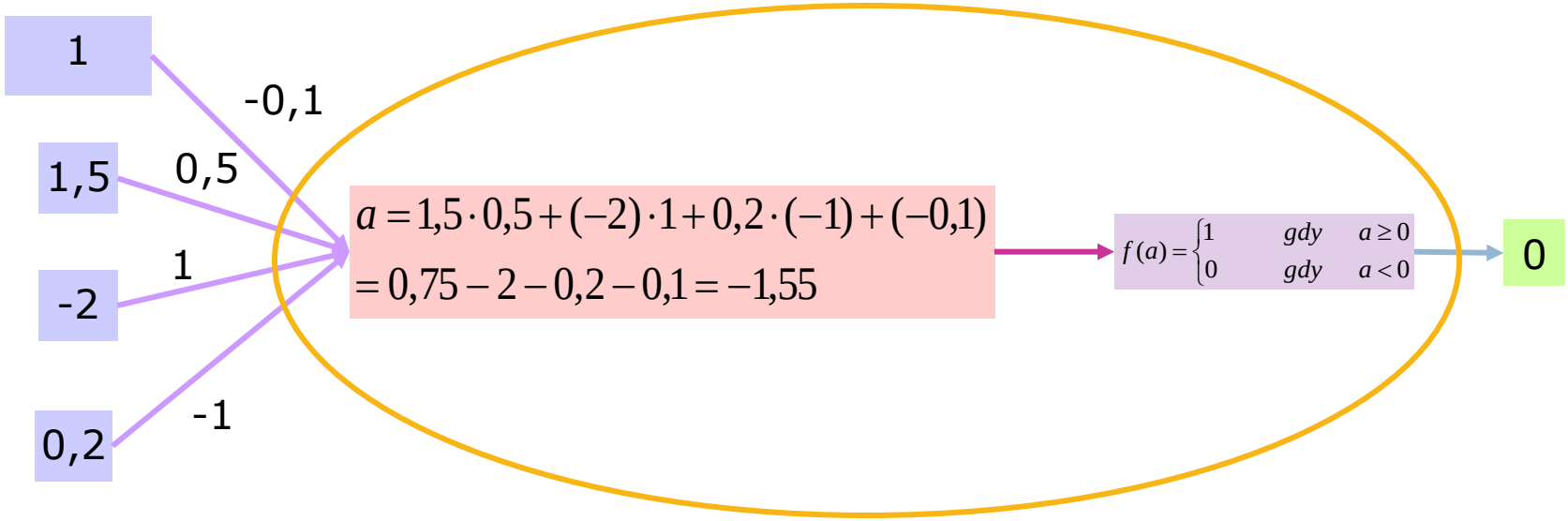
$$f(a) = e^{-\frac{\beta}{a^2}}$$



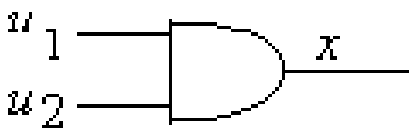
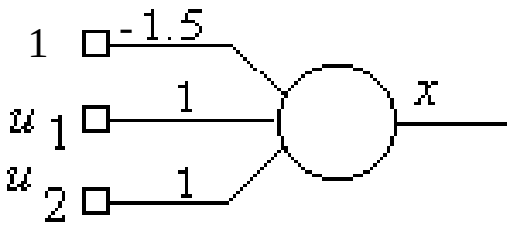
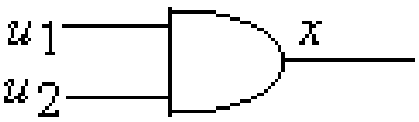
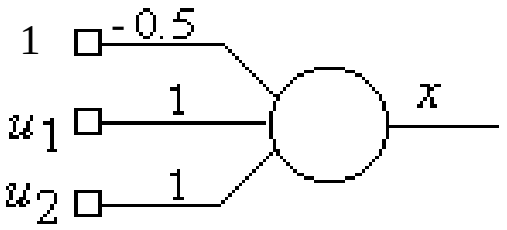
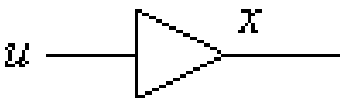
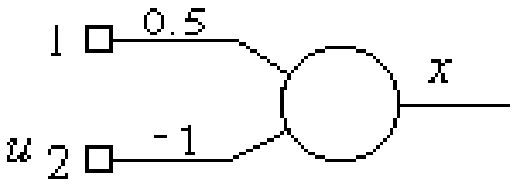
Cechy sztucznego neuronu

- Wejścia i wagi są liczbami rzeczywistymi dodatnimi i ujemnymi.
- Jeżeli jakaś cecha (wejście) powoduje odpalenie neuronu, waga będzie dodatnia, a jeżeli cecha ma działanie hamujące to waga jest ujemna.
- Neuron dokonuje sumowania i dopasowania do progu (biasu) θ.
- Przyjmuje się traktowanie progu θ jako wagi w_0, gdzie wejście jest zawsze równe 1.

Przykład działania



Neuron jako bramka logiczna

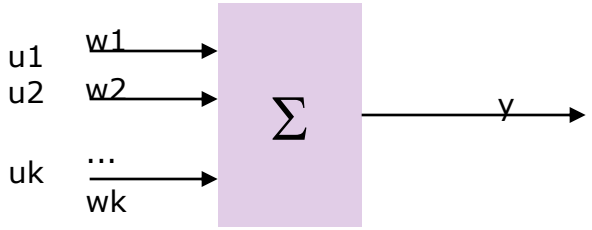
BOOLEAN FUNCTION	LOGICAL GATE	ARTIFICIAL NEURON
AND $x = u_1 \wedge u_2$		
OR $x = u_1 \vee u_2$		
NOT $x = \neg u$		

Typy sieci neuronowych

- Ze względu na funkcję aktywacji:
  - liniowe (f(a) = a)
  - nieliniowe
  - radialne (funkcja gaussa)
- Ze względu na architekturę
  - jednokierunkowe
  - ze sprzężeniami zwrotnymi
  - mieszane

Neuron liniowy

- Bez funkcji aktywacji (funkcja aktywacji liniowa).



- Działanie neuronu można opisać równaniem wektorowym:
  - $y = W^T U$,
  - gdzie $U=<u_1,u_2,...,u_k>^T$, wektor wejść
  - a $W =<w_1,w_2,...,w_k>^T$, wektor wag.
- Wyjście y będzie miało największą wartość, gdy położenie wektora wejściowego U, będzie najbardziej przypominać położenie wektora wag W.
- W wektorze wag zatem pamiętany jest sygnał wzorcowy.

Perceptron prosty

- Najstarszą koncepcją [Rosenblatt 1962] sieci neuronowej jest perceptron.
- Perceptron prosty składa się z jednej warstwy neuronów.
- W najprostszym przypadku składa się z pojedynczego neuronu.
- Jeżeli funkcja aktywacji to signum i sieć składa się z 1 neuronu, to
  - zadaniem perceptronu jest klasyfikacja wektora $u=[u_1,...,u_n]^T$ do jednej z dwóch klas: L_1 (sygnał wyjścia 1) lub L_2 (sygnał wyjścia równy -1).
  - Zatem perceptron dzieli przestrzeń wejść N-wymiarową hyperpłaszczyzną (granicą decyzyjną) o równaniu:

$$\sum_{i=1}^{k} w_i u_i + \theta = \sum_{i=0}^{k} w_i u_i = 0$$

Analiza działania perceptronu

- Zakładamy, że perceptron składa się z 1 neuronu. Wejściami są dwie liczby u_1 i u_2. Perceptron dzieli płaszczyznę dwuwymiarową granicą decyzyjną o wzorze:

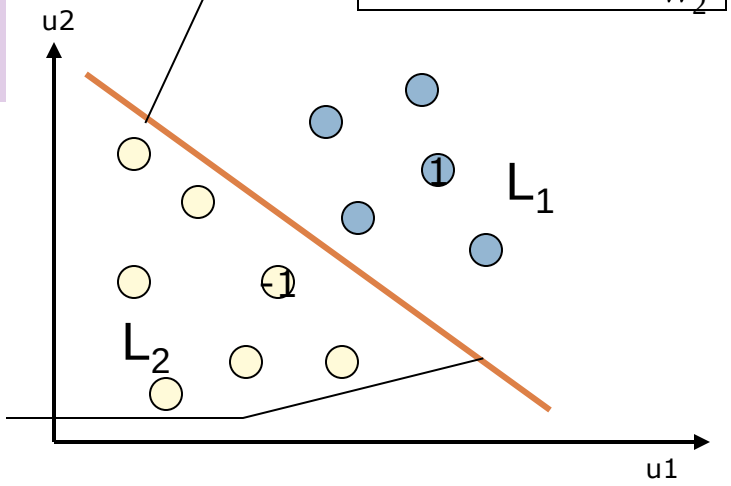
$$w_1 u_1 + w_2 u_2 + w_0 = 0$$

- stąd otrzymujemy:

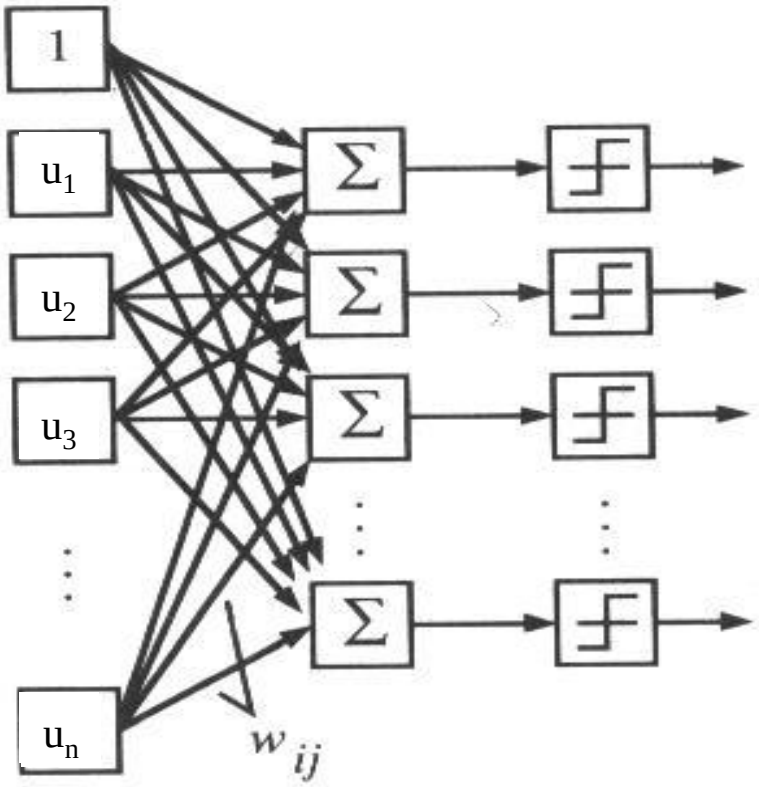
$$u_2 = -\frac{w_1}{w_2} u_1 - \frac{w_0}{w_2}$$

współczynnik kierunkowy $-\frac{w_1}{w_2}$

przesunięcie $-\frac{w_0}{w_2}$



Perceptron jednowarstwowy



Klasyfikacja do większej liczby klas

- Wartości wyjścia są kodowane ciągiem o długości *m* (liczba neuronów w warstwie) wartościami 1 i –1 (funkcja aktywacji signum). Jeżeli potrzebujemy *m* elementów na kodowanie wymagamy *m* neuronów w perceptronie.
- np.. Klasa C=(-1,-1,1,-1). Przyporządkowanie wejść do klasy C nastąpi, gdy wyjścia z sieci będą kolejno przyjmować wartości: -1,-1,1,-1.
- Przyjmuje się, że potrzeba *m* wyjść by przyporządkować do *m* klas.

Uczenie się SSN z nauczycielem

- Uczenie się jest procesem zmiany wartości wag i wartości progowych w SSN. Rozpatrując pojedynczy neuron: uczenie zmienia zatem współczynnik kierunkowy i przesunięcie granicy decyzyjnej.
- Jest to proces iteracyjny.
- Wymaga zebrania danych:
  - Należy zebrać dane w parach: wejścia i wyjścia. np. cechy obiektów poddawanych klasyfikacji i klasa do której należą.
  - Zbiór danych musi być na tyle duży by wybrać z niego dwie grupy danych:
    - dane uczące i
    - testujące.

Uczenie z nauczycielem cd.

- Uczenie z nauczycielem polega na wskazaniu poprawnej klasyfikacji sygnału do odpowiednich klas pomimo braku znajomości wag.
- Po zakończeniu uczenia perceptron powinien poprawnie wskazywać przynależność podanego elementu do danej klasy, nawet dla sygnałów nie wchodzących w skład zbioru uczącego (dla zbioru testowego).

Uczenie regułą DELTA

- Uczenie z nauczycielem dla sieci typu perceptron prosty:
  - Zakładamy, że wektor wejściowy U jest związany zależnością funkcyjną z wyjściem *y*: *y*=f(U).
  - Funkcja f nie musi być znana. Znane musi być *z*, które stanowi żądanie odnośnie sygnału wyjściowego: *z*=f(U).
- Algorytm uczenia regułą DELTA wymaga podania dla każdego zestawu wejść U wartości odpowiedzi *z* (zadana wartość odpowiedzi).
- Neuron na zadane sygnały U odpowiada pewną wartością *y*. Jeżeli jest ona różna od *z* to neuron nie jest nauczony odpowiedzi na sygnał U. Błąd popełniony przez sieć jest równy: $\delta = z - y$.

DELTA cd.

- Zła odpowiedź wymaga korekty. Dotychczasowy wektor wag zmieni się według wzoru:
  - W'=W+ $\eta\delta$U,
  - gdzie η współczynnik liczbowy decydujący o szybkości uczenia.
- Uczenie może odbywać się dla wielu punktów „idealnych". Taki zbiór danych nazywa się zbiorem uczącym. Składa się on z par: $<U_j, z_j>$.
- W pierwszym kroku wymagane jest, by dane były jakieś wartości wag. Dobiera się je losowo.
- Celem procesu uczenia jest uzyskanie odpowiedzi neuronu *y* zgodnych z zadanymi odpowiedziami *z* , co można określić jako proces minimalizacji funkcji:

$$Q = \frac{1}{2}\sum_{j=1}(z_j - y_j)^2$$

gdzie j liczbą wzorców.

Algorytm uczenie metodą DELTA

- *Dane*: problem klasyfikacji obiektu na podstawie *n* cech. Wektor wejść: (u_1,..., u_n). Klasyfikacja do dwóch klas: L_1 (1) i L_2 (-1).
- *Obliczane*: Zbiór wag (w_0,..,w_n).

1. Stwórz perceptron z n+1 wejściami, wejście u_0 zawsze ma wartość 1 (bias).
2. Inicjalizuj losowo wagi.
3. Dla wszystkich k wzorców uczących:
   a. Jeżeli wszystkie przykłady sklasyfikowane, wyświetl wagi i zakończ.
   b. W przeciwnym przypadku oblicz *Err* , które będzie równe *u* (bo δ=1), gdy sklasyfikowane do klasy: –1, a powinno być do klasy: 1 lub *–u* (bo δ=-1) w przeciwnym wypadku.
   c. Modyfikuj wagi :

$$w_{t+1} = w_t + \eta Err$$

Przykład – dane

k=10 wzorców

	u_1	u_2	
	475 Hz	**557 Hz**	**Quality OK?**
	0.958	0.003	yes
	1.043	0.001	yes
	1.907	0.003	yes
	0.780	0.002	yes
	0.579	0.001	yes
	0.003	0.105	no
	0.001	1.748	no
	0.014	1.839	no
	0.007	1.021	no
	0.004	0.214	no

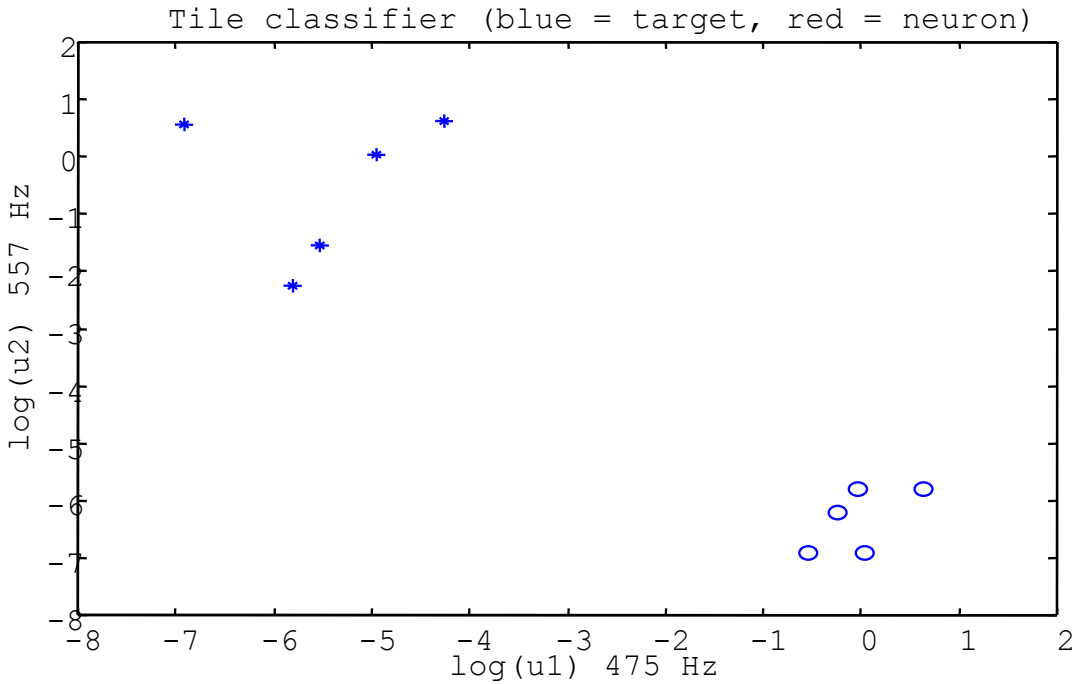
L1

L2

Zadanie: Nauczyć sieć klasyfikacji do 2 klas: „yes (pęknięta)” i „no (nie pęknięta)” dla zbioru opisującego dane z systemu rozpoznawania pękniętych dachówek. Uderzenia w pojedynczy element są rejestrowane i filtrowane.

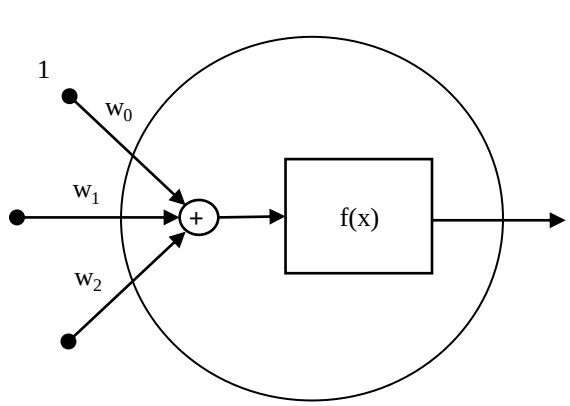
fuzzy.iau.dtu.dk/demo/detron1/detron1.PPT

Przykład – wykres

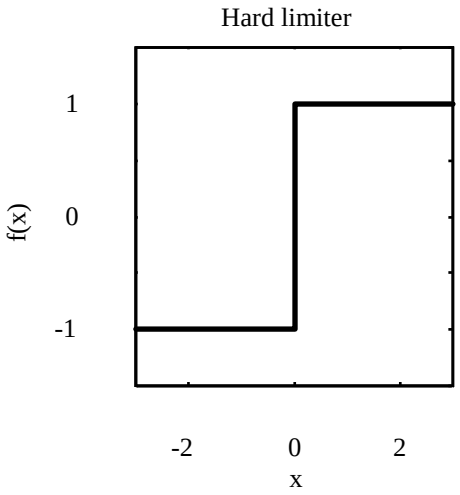


- o – klasa wartości „yes”;
- * – klasa wartości „no”.

Przykład – perceptron



(a)



(b)

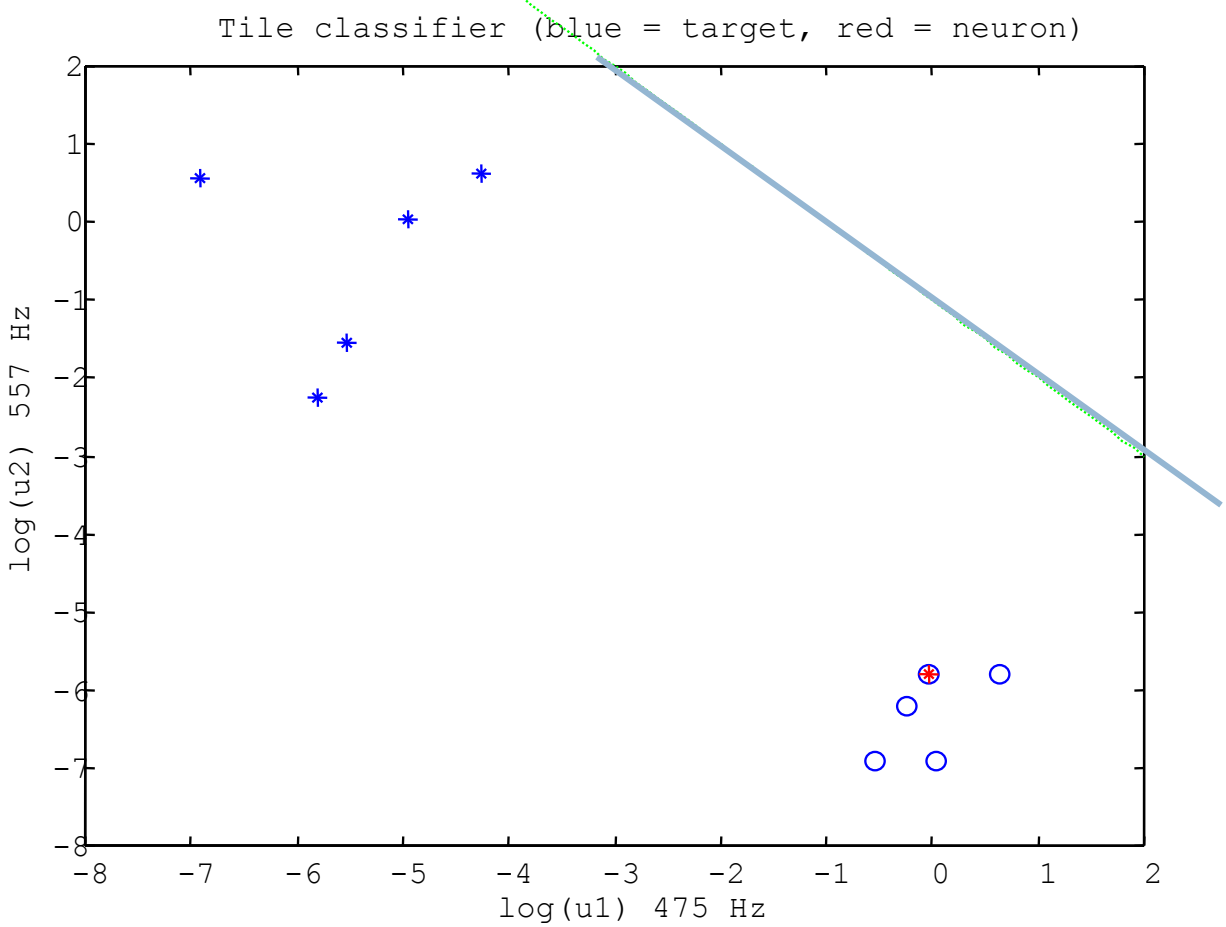
Sieć zastosowana do zadania:

- perceptron prosty,
- funkcja aktywacja signum.

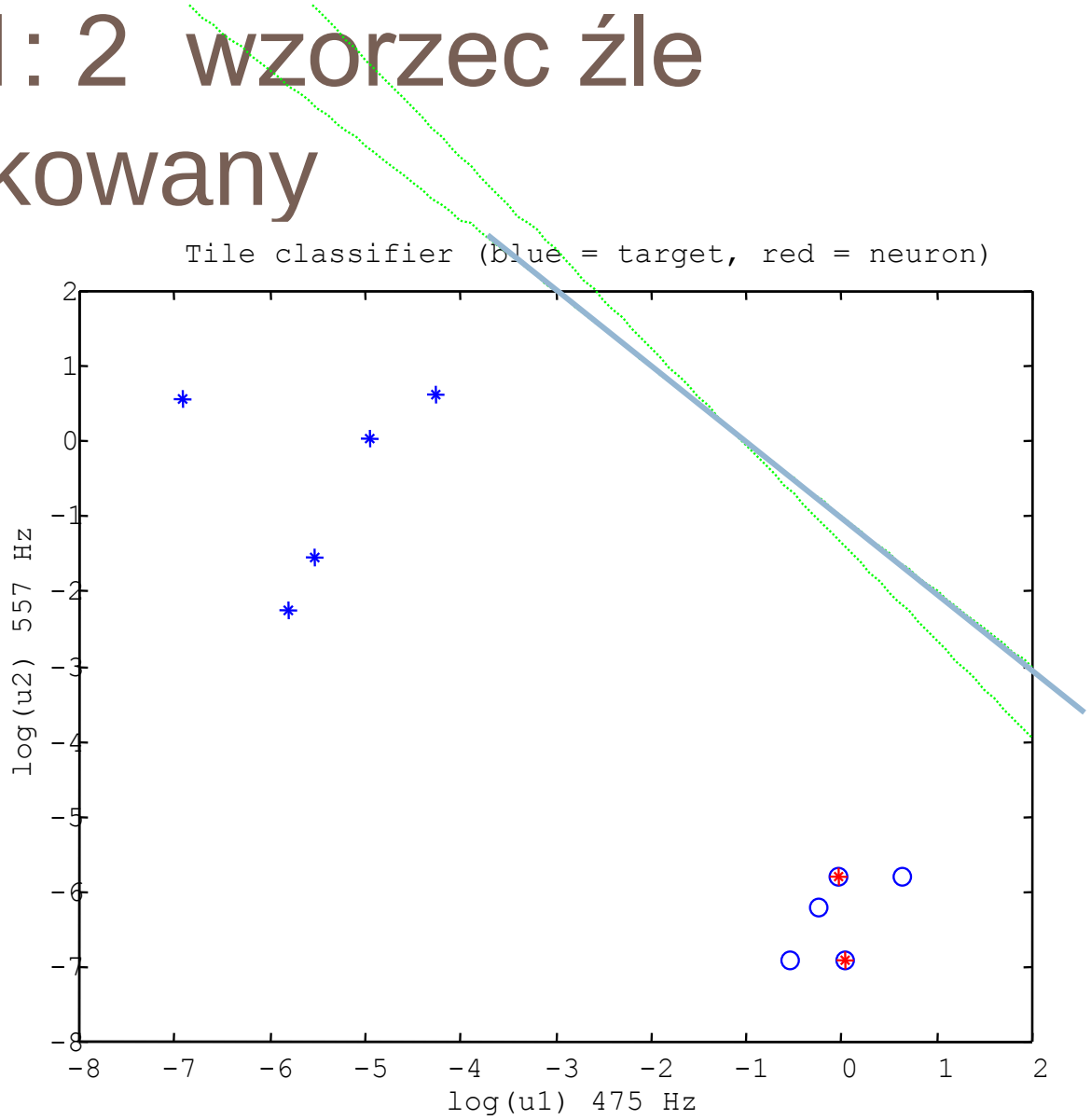
Przykład: opis postępowania

- W procesie uczenia podajemy kolejne cele jako wejścia (oznaczane na czerwono) i sprawdzamy jakość klasyfikacji.
- Każdy punkt niewłaściwie sklasyfikowany (zaznaczony znakiem czerwonym odpowiedniej klasy) zmienia wagi i powoduje przesunięcie granicy decyzyjnej (zielona linia).

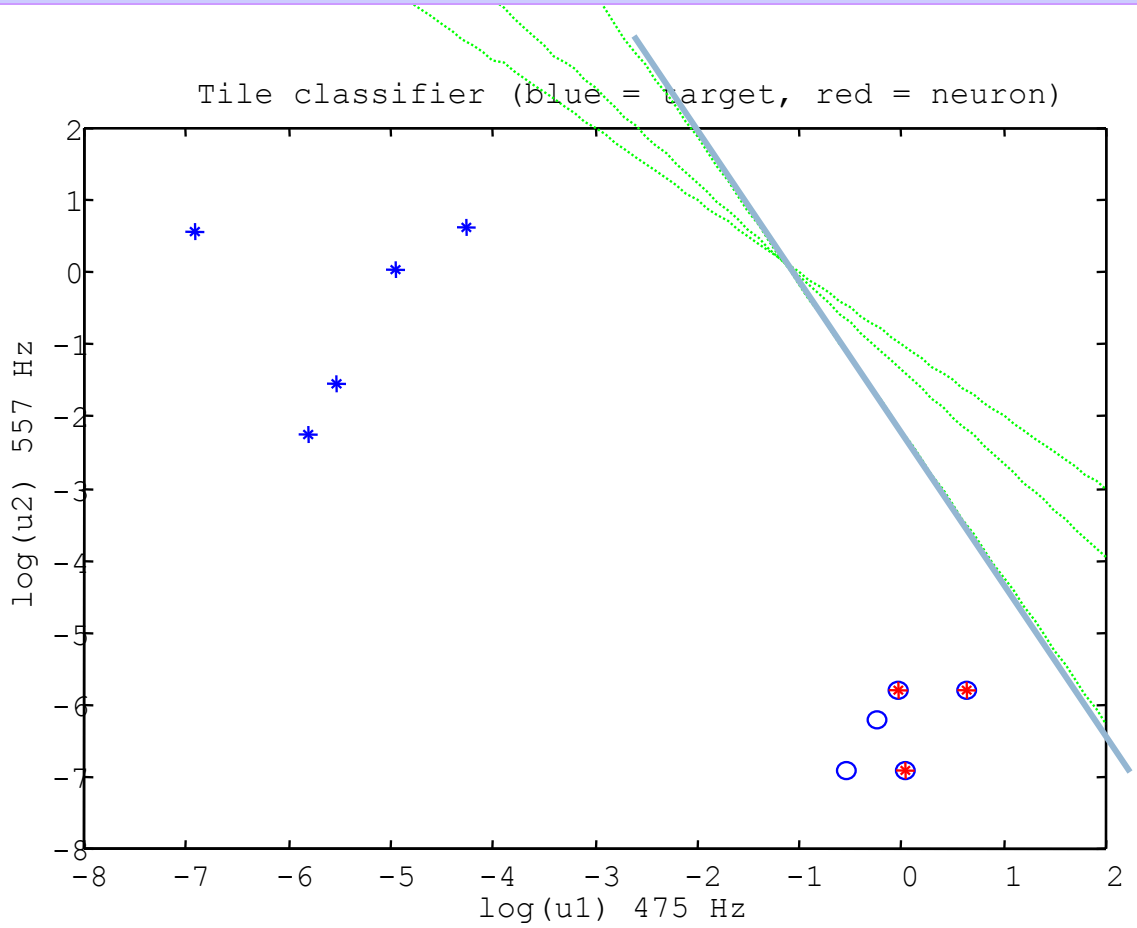
Epoka 1: 1 wzorzec źle sklasyfikowany, zmiana położenia granicy decyzyjnej



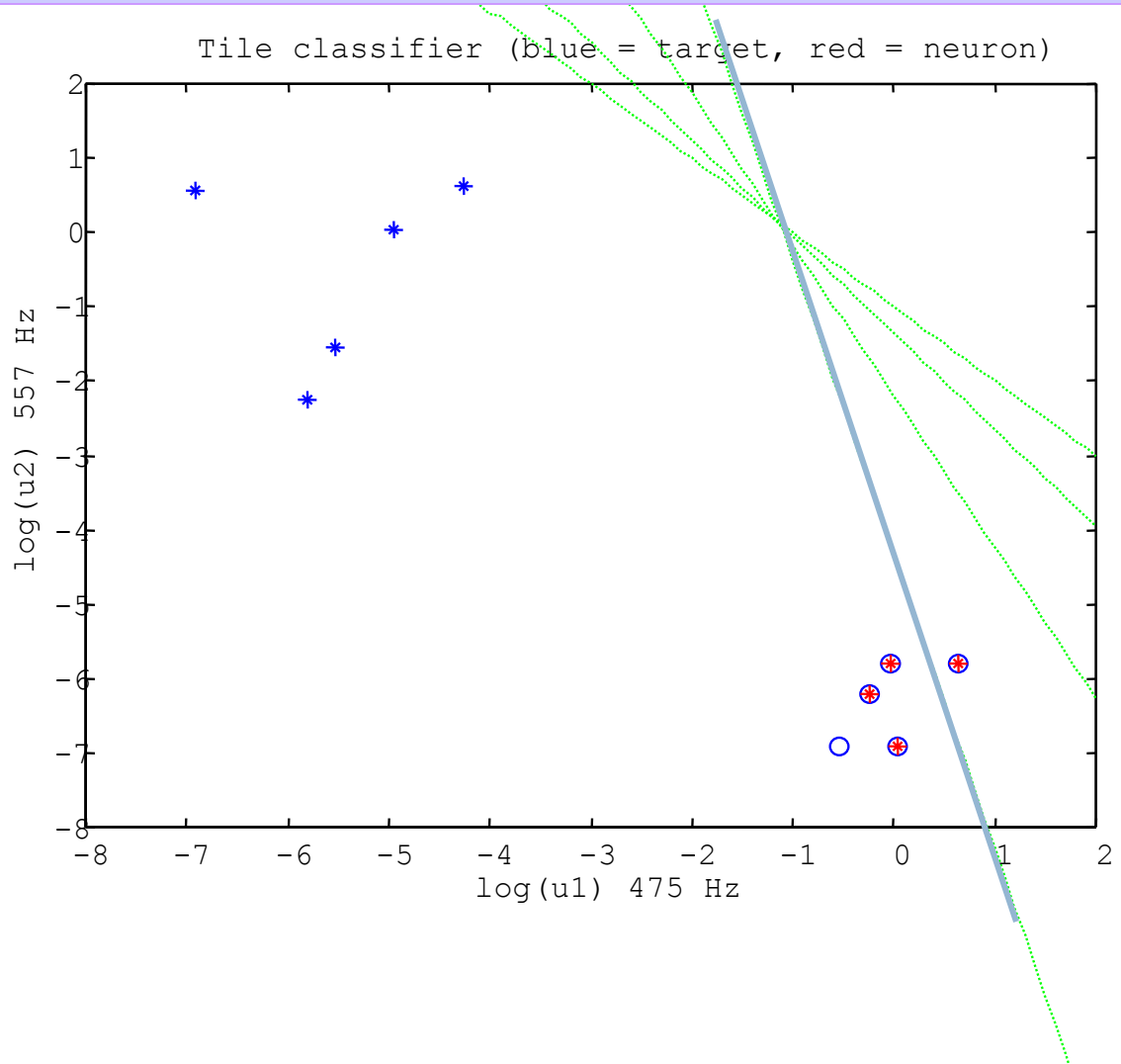
Epoka 1: 2 wzorzec źle sklasyfikowany



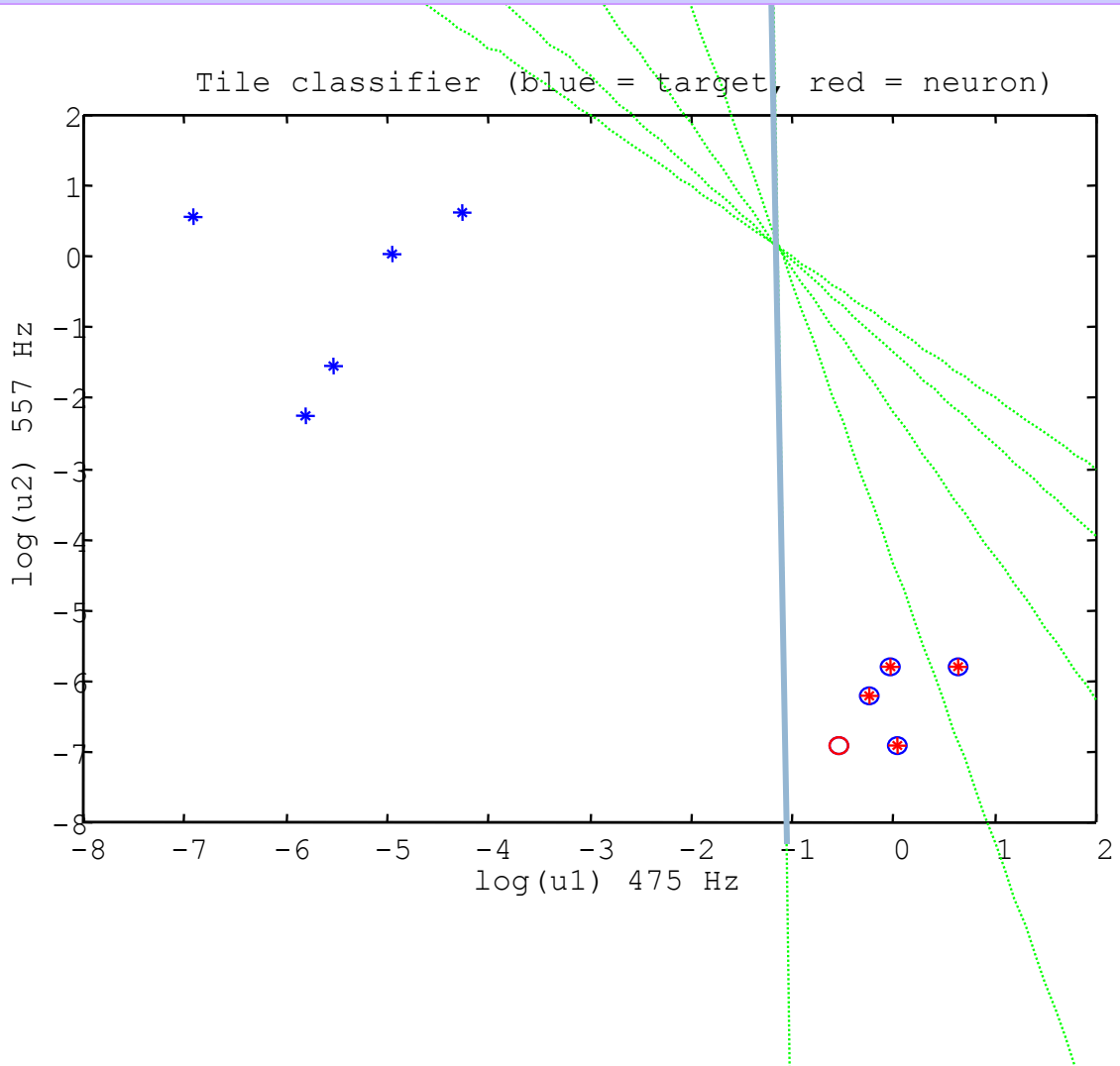
Epoka 1: 3 wzorzec źle sklasyfikowany



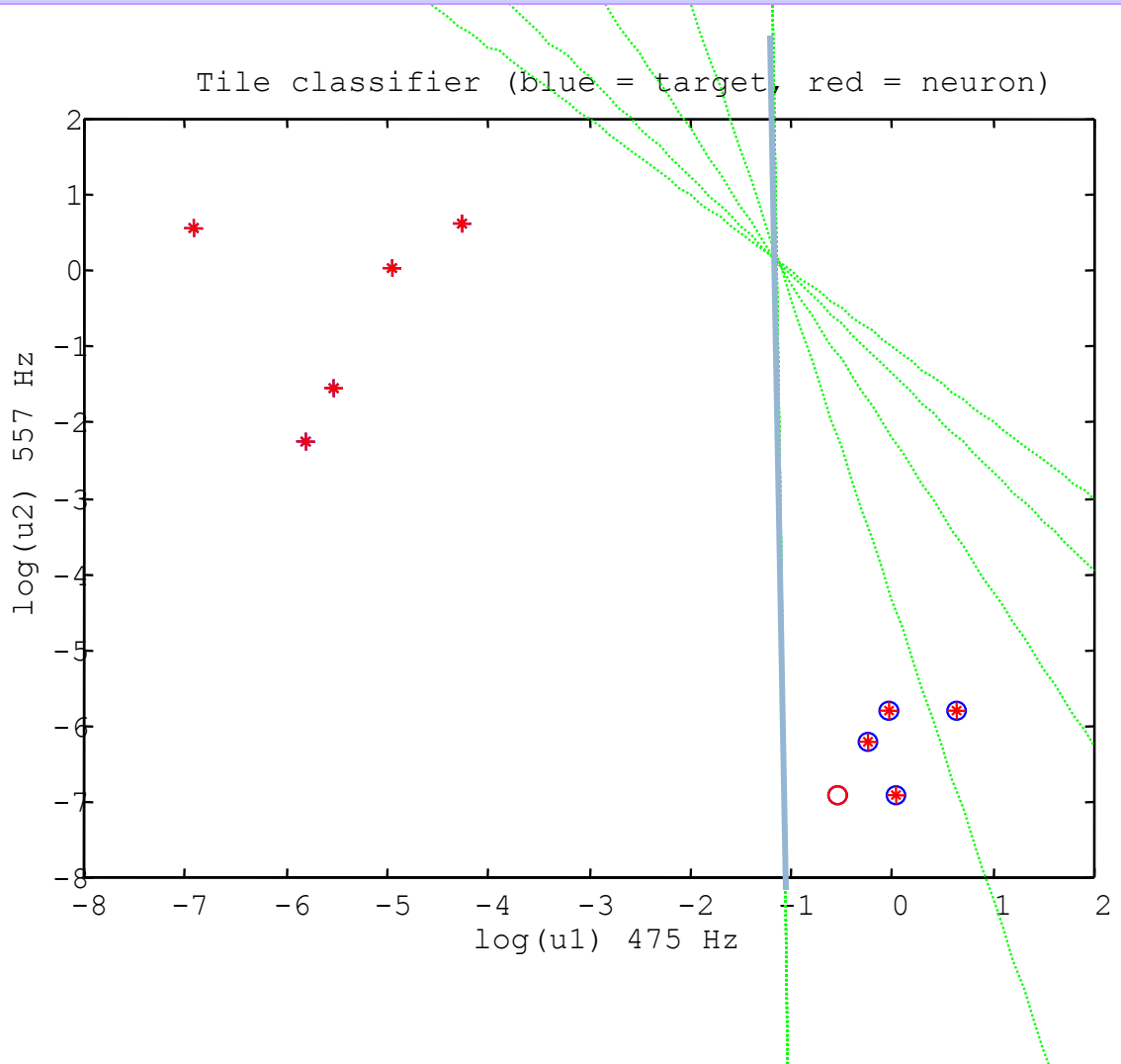
Epoka 1: 4 wzorzec źle sklasyfikowany



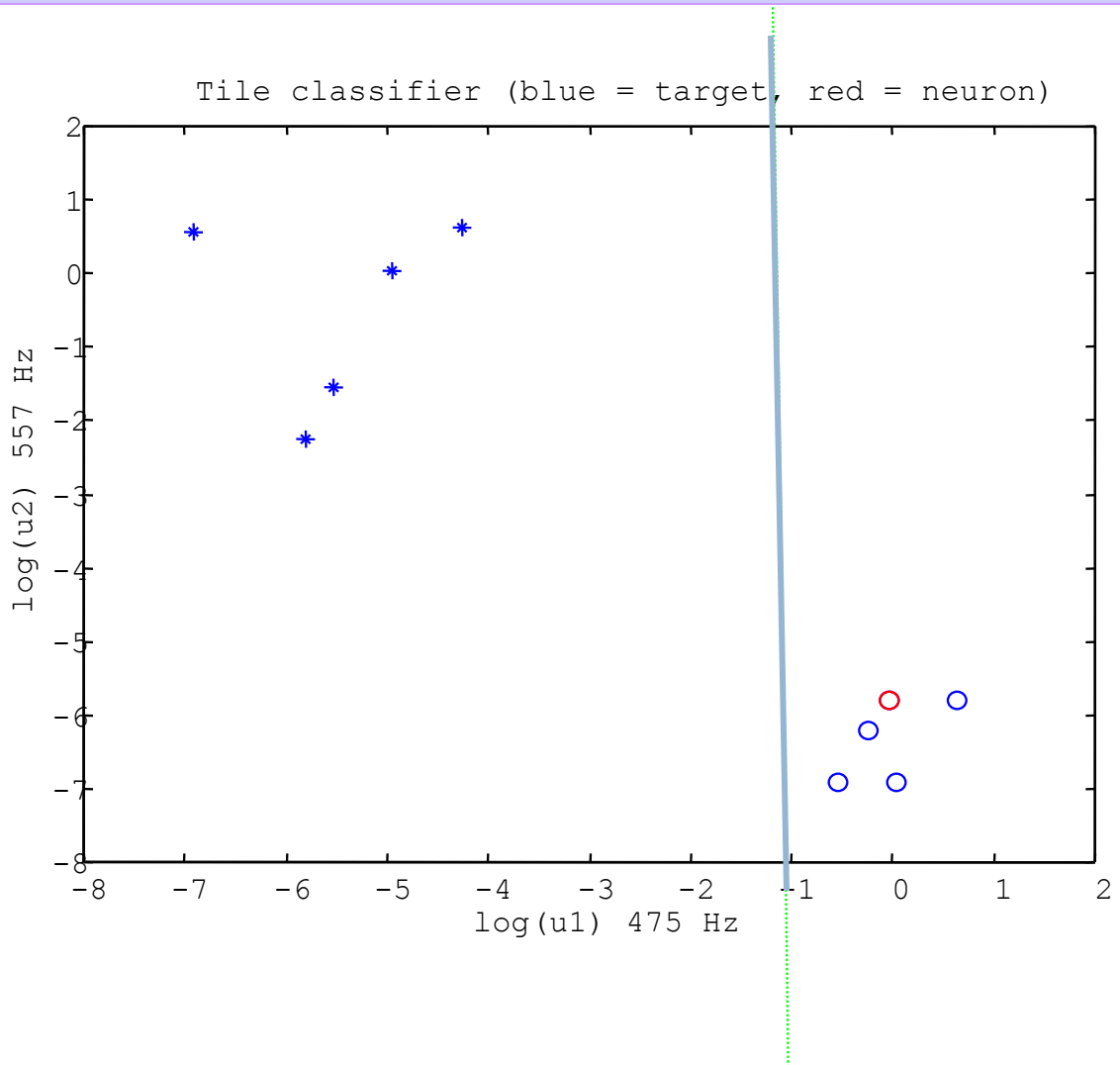
Epoka 1: 5 wzorzec dobrze sklasyfikowany



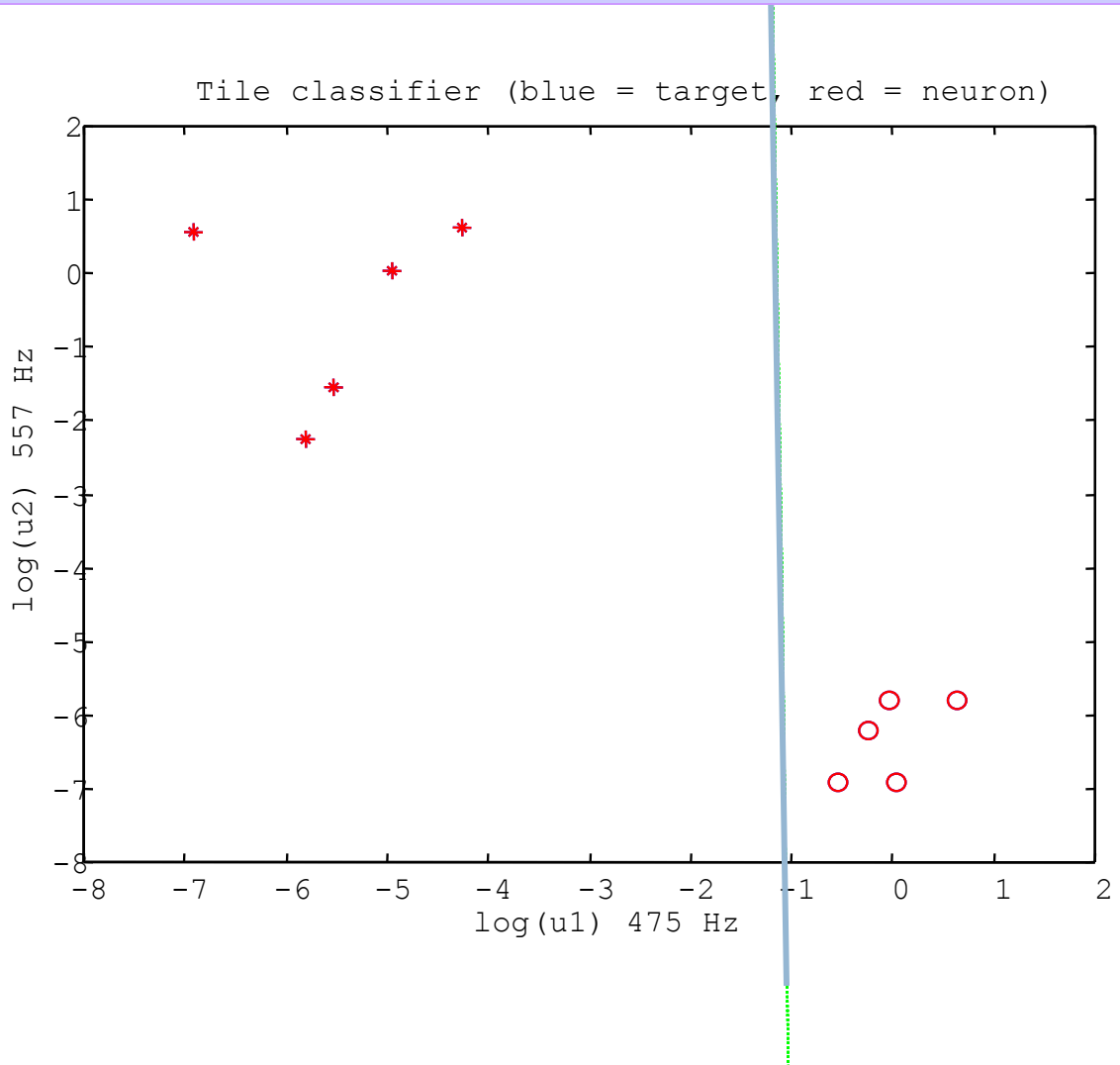
Epoka 1: wzorce 6 do 10 dobrze sklasyfikowane



Epoka 2: 1 wzorzec dobrze sklasyfikowany



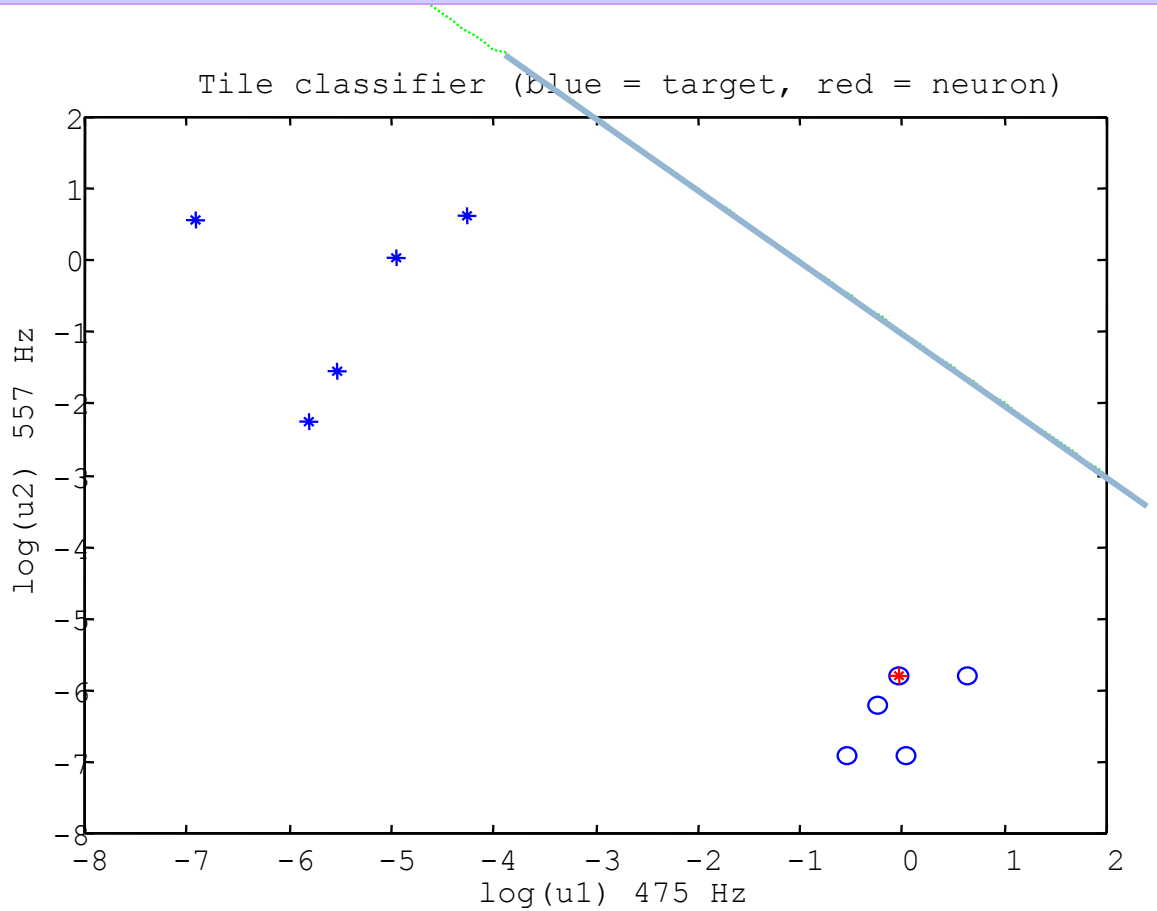
Epoka 2: wzorce 2 do 10 dobrze sklasyfikowane



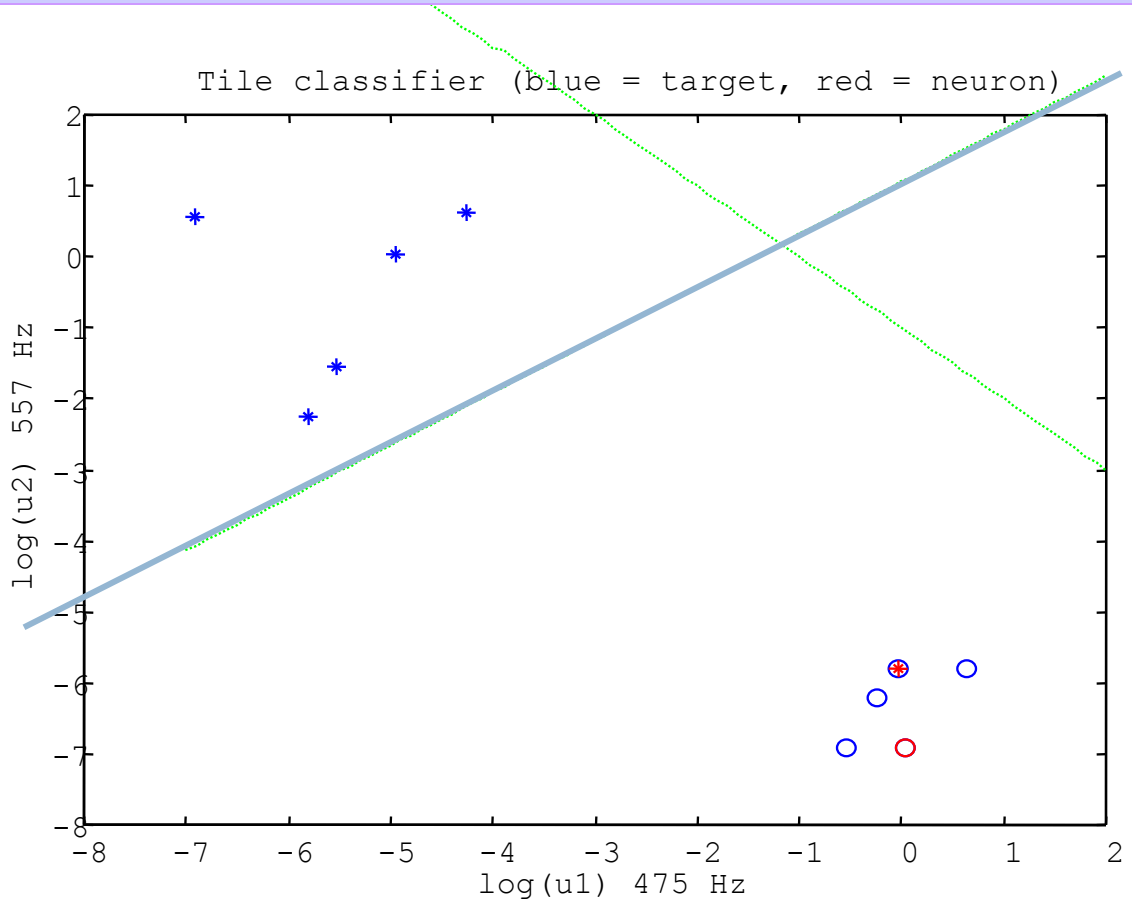
ZWIĘKSZAMY WSPÓŁCZYNNIK UCZENIA Z 0,01 NA 0,1

i dokonujemy ponownego nauczania sieci

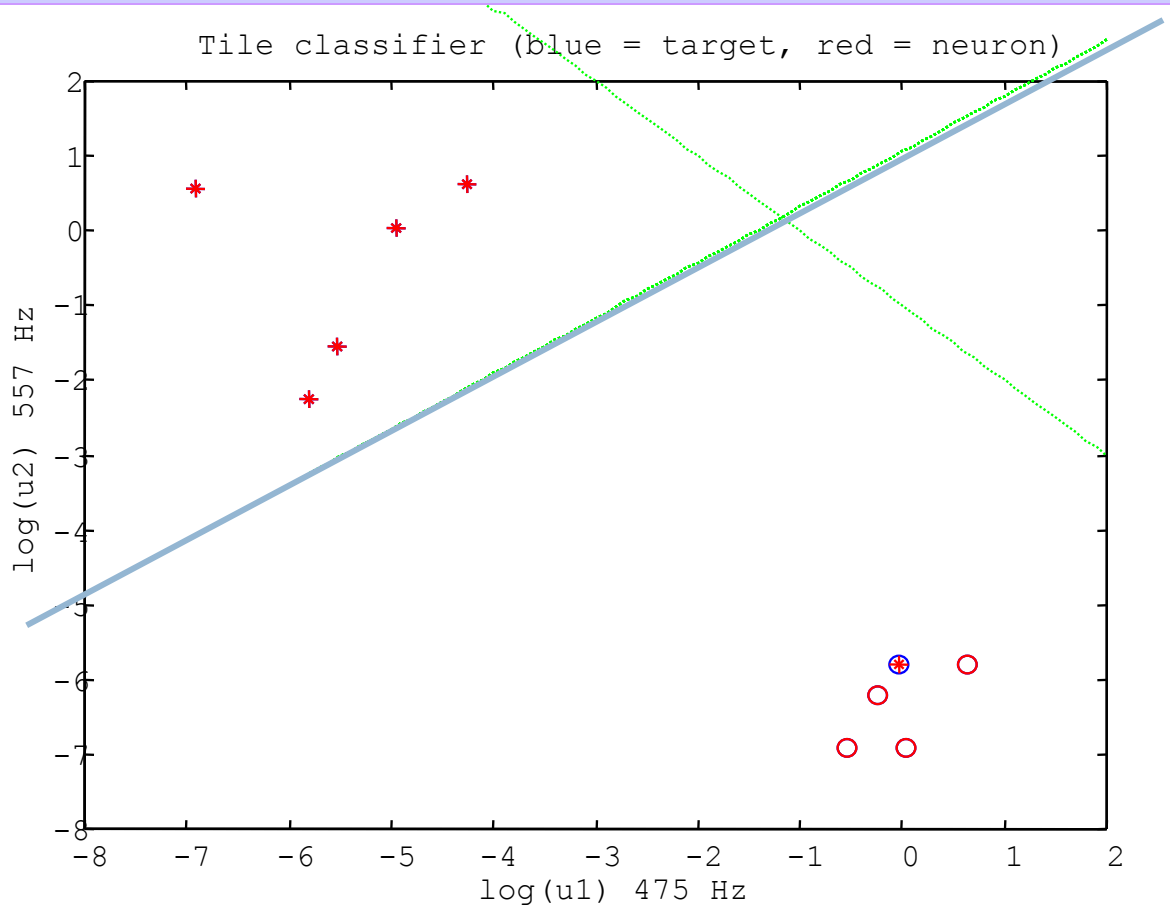
Epoka 1: 1 wzorzec źle sklasyfikowany



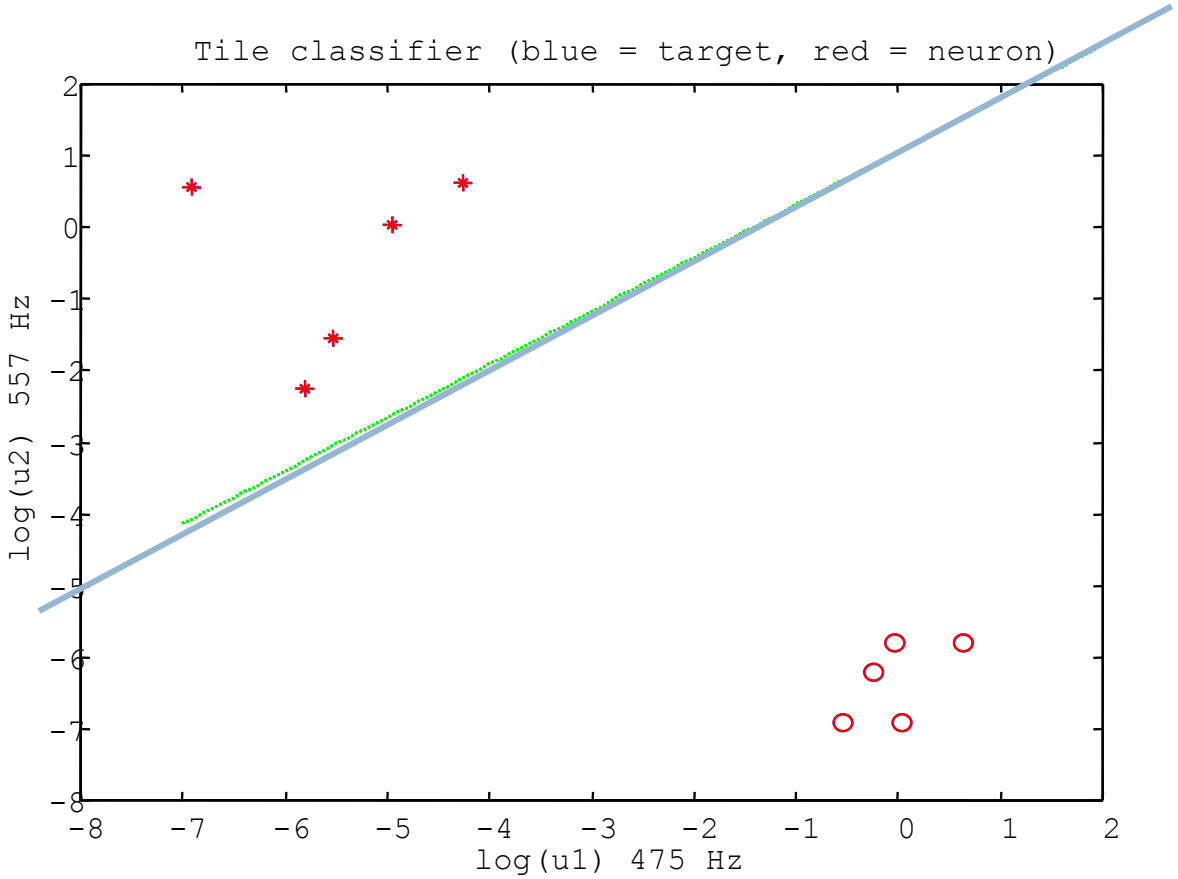
Epoka 1: 2 wzorzec dobrze sklasyfikowany



Epoka 1: wzorce od 3 do 10 dobrze sklasyfikowane



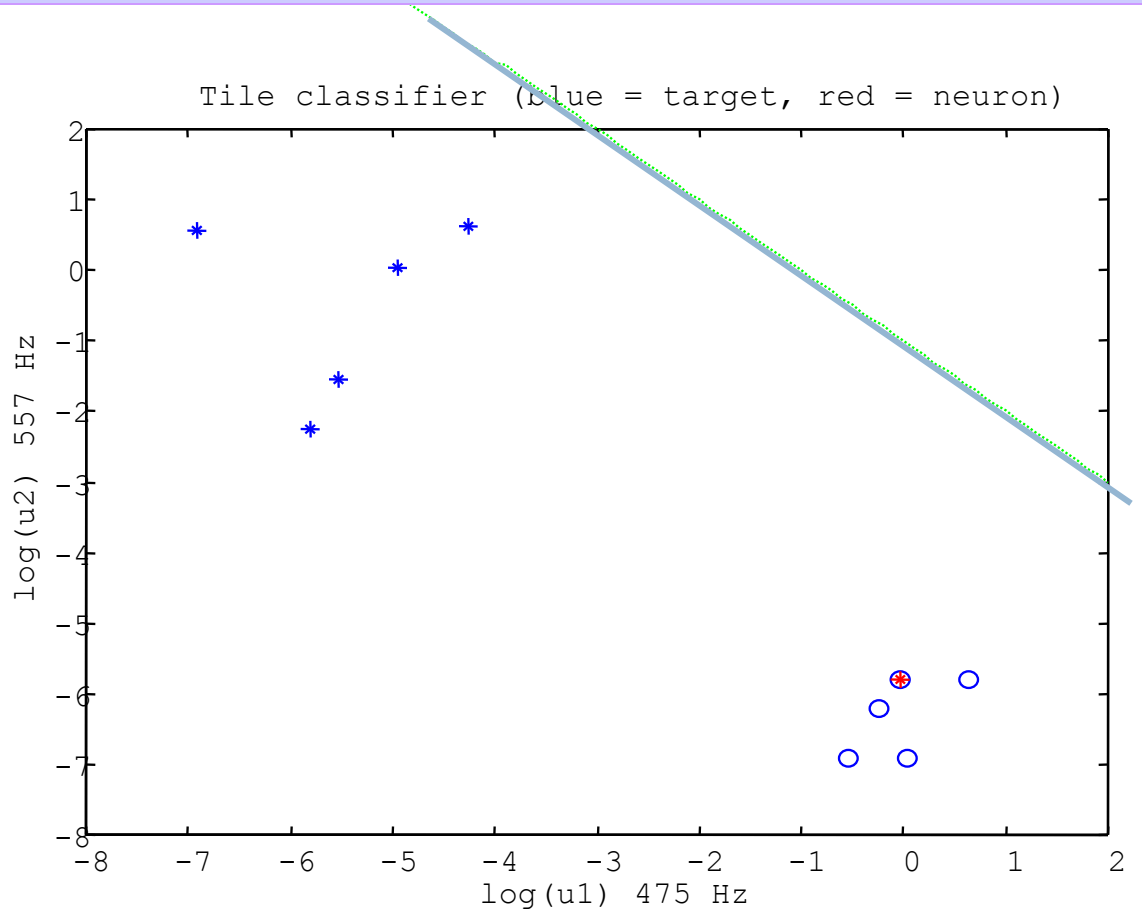
Epoka 2: wzorce od 1 do 10 poprawne.
Granica pozostaje ustalona, nauczanie zakończone



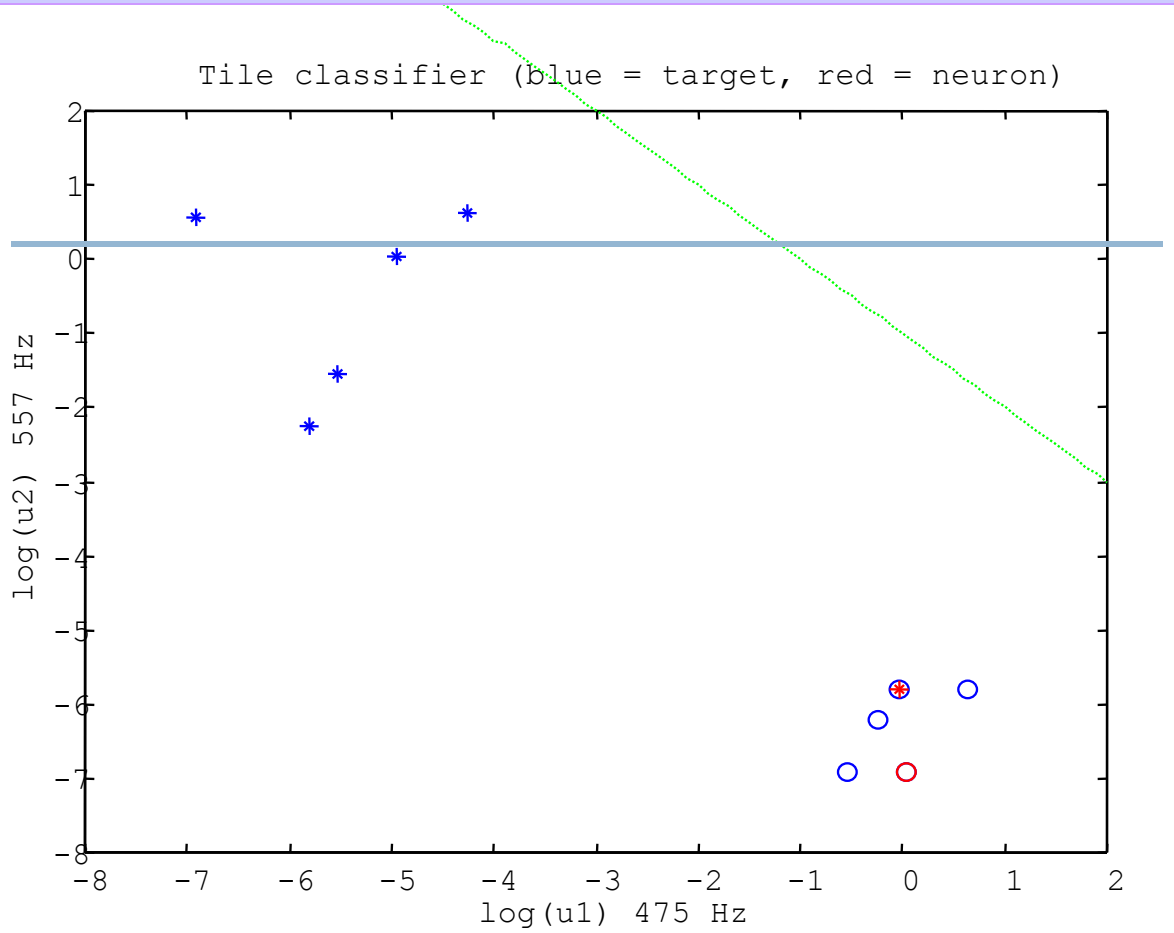
ZWIĘKSZAMY WSPÓŁCZYNNIK UCZENIA Z 0,1 NA 10

i dokonujemy ponownego nauczania sieci

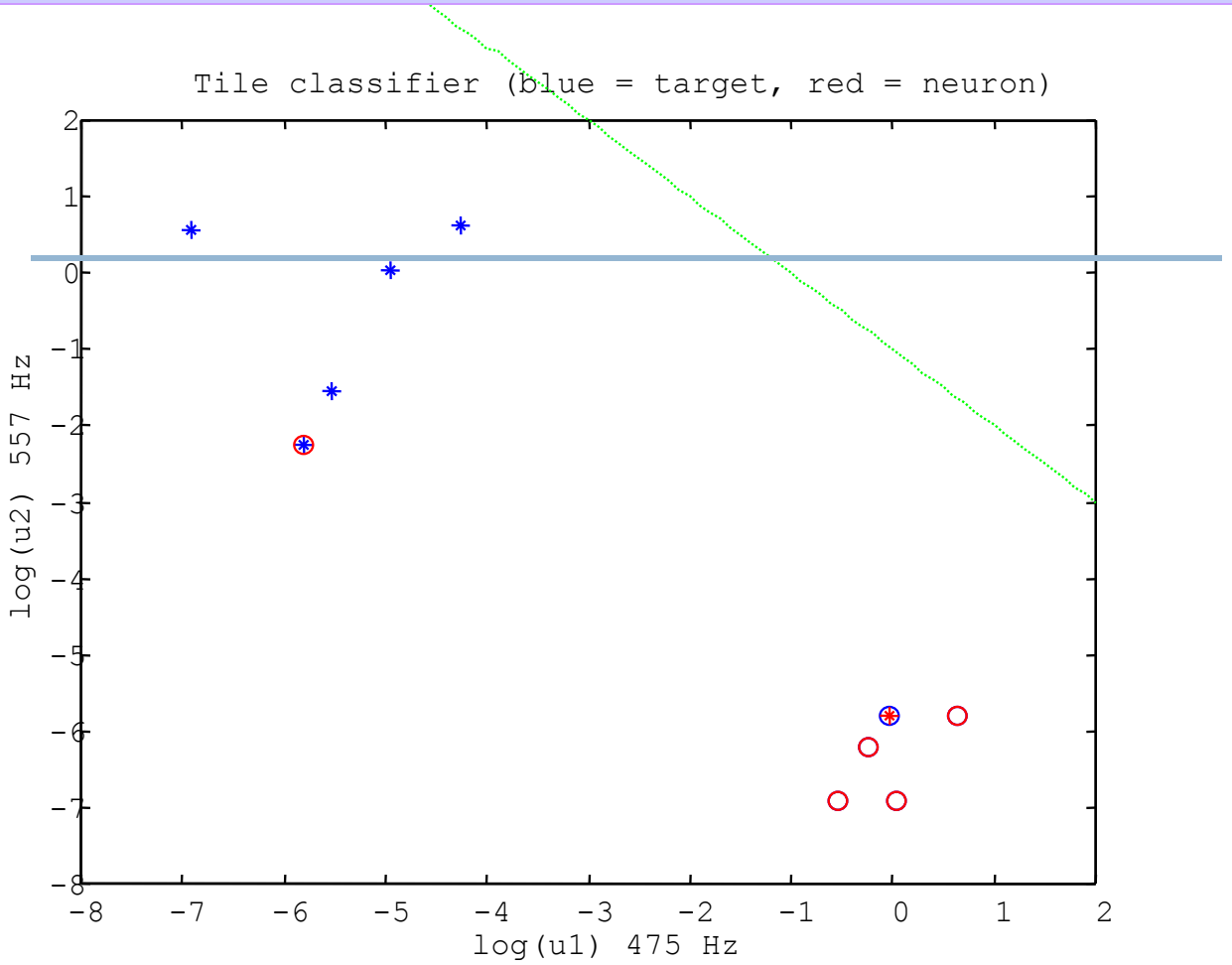
Epoka 1: 1 wzorzec źle sklasyfikowany



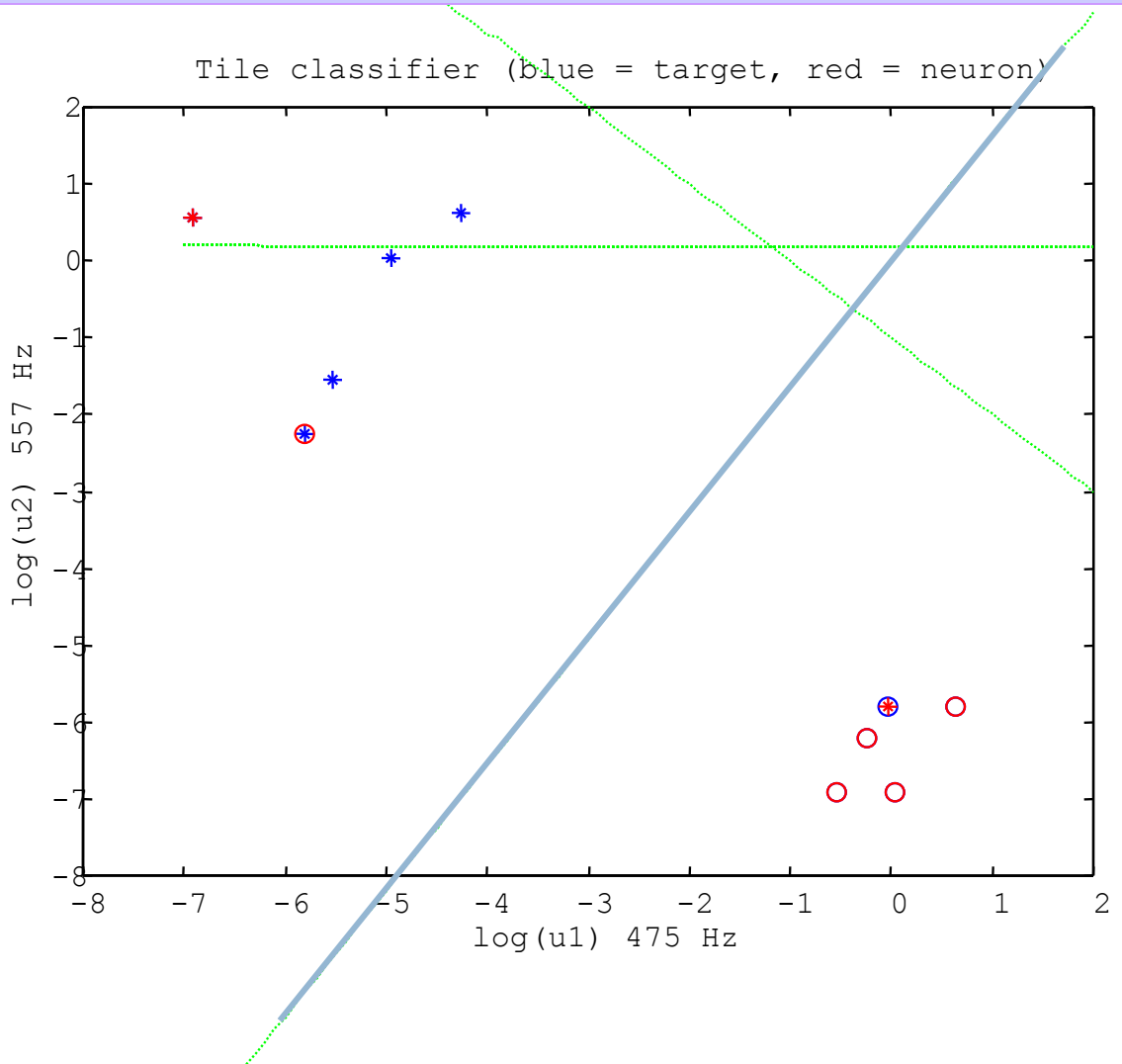
Epoka 1: 2 wzorzec dobrze sklasyfikowany



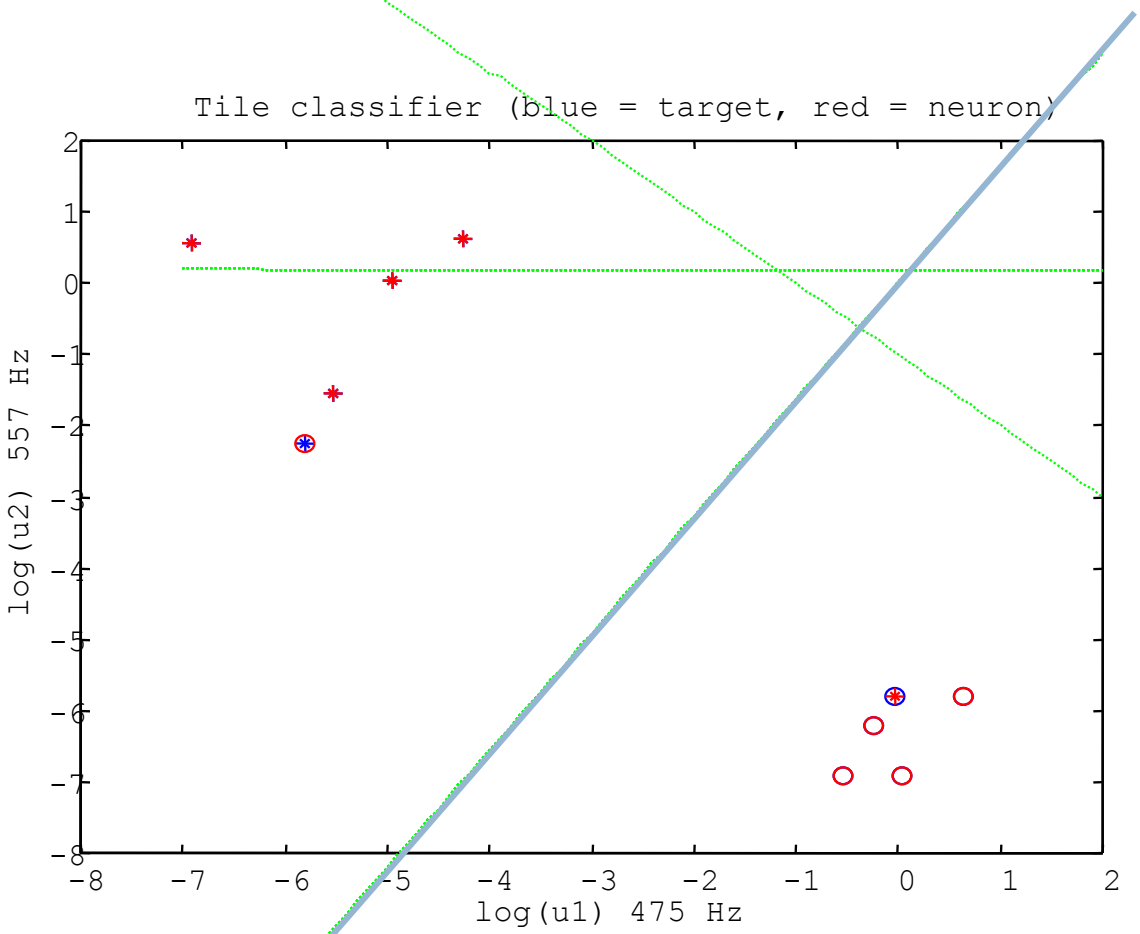
Epoka 1: wzorce od 3 do 5 dobrze sklasyfikowane. 6 niepoprawny, zmienia położenie granicy decyzyjnej.



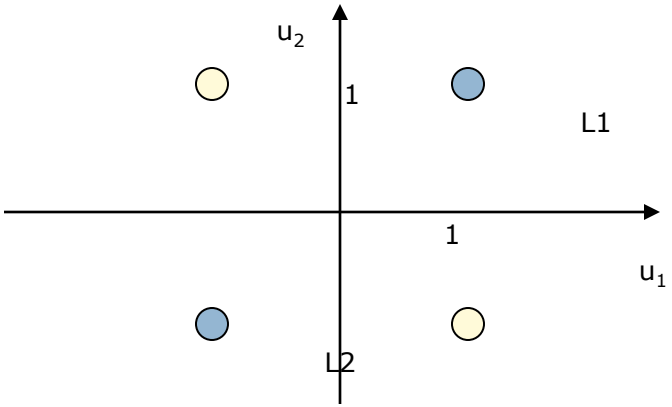
Epoka 1: 7 wzorzec dobrze sklasyfikowany



Epoka 1: wzorce 8 - 10 (poprawne). Zbieżność niestabilna, współczynnik nauczania zbyt duży.

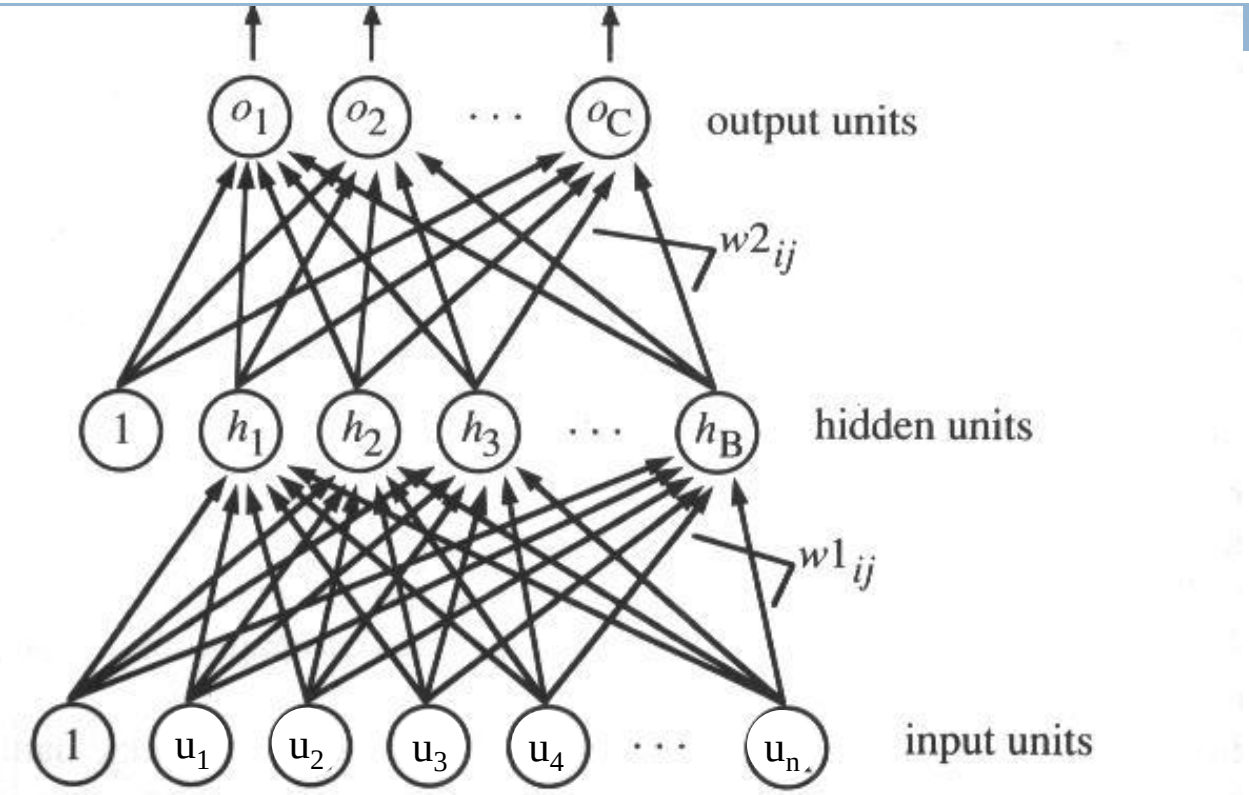


Problem typu XOR



- Brak liniowej separowalności.
- Przy liniowym podziale z jedną granica decyzyjną nie uzyska się podziału.

Perceptron wielowarstwowy

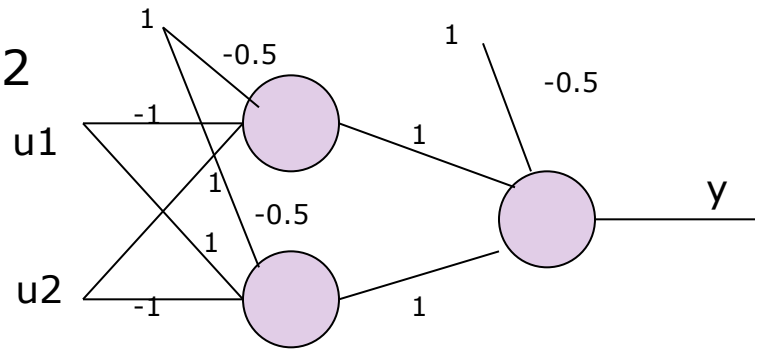


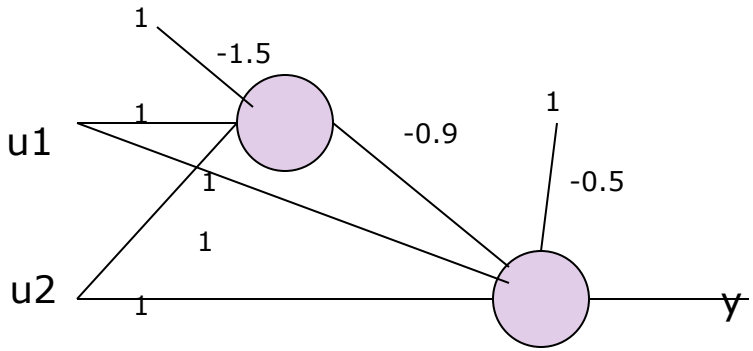
Perceptrony wielowarstwowe mogą zamodelować wszystko. Jeżeli 1 element przetwarzający może być dowolną bramką logiczną AND, OR, NOT, to na podstawie określenie minimalnej biblioteki wiemy, że można a za pomocą takich elementów opisać dowolnie złożona funkcję.

Rozwiązanie problemu XOR

u1	u2	y
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

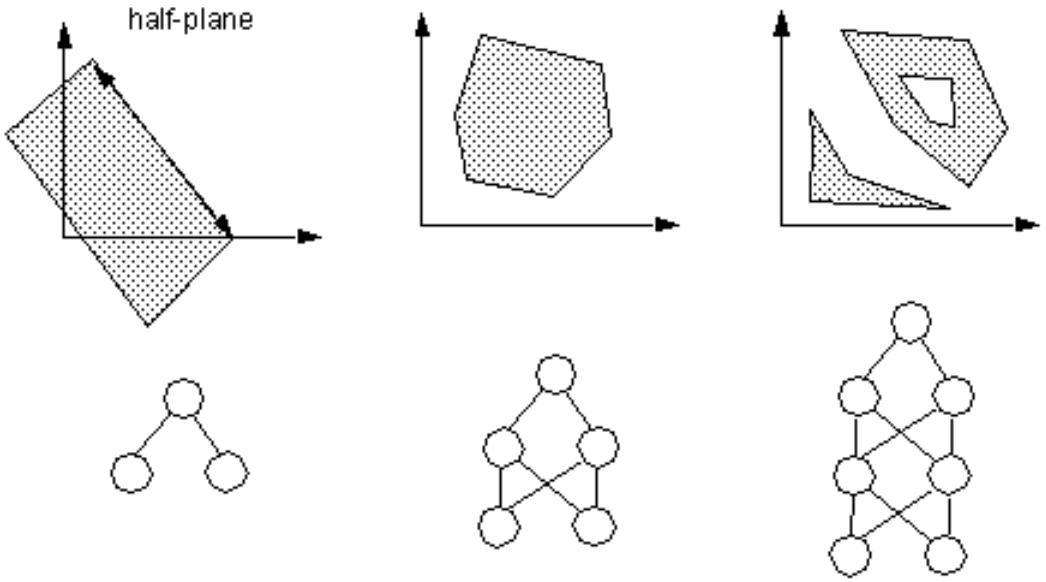
y= ~u1u2+u1~u2





- Funkcja aktywacji: progowa.

Płaszczyzny klasyfikacji w zależności od modelu sieci



Jak długo uczyć?

- Cel: znalezienie punktu równowagi pomiędzy dwoma celami: poprawną odpowiedzią na wzorce uczące i poprawną odpowiedzią na wzorce testujące (nowe) (zapamiętywanie kontra generalizacja)
- Można przyjąć, że zaprzestajemy uczenia po osiągnięciu saysfakcjonującego poziomu błędu np.. 5%
- Zadana liczba epok powoduje uniknięcie nieskończonych pętli.
- Jeżeli uczeniu przebiega zbyt długo ryzykuje się przeuczenie sieci.

Generalizacja

- Najbardziej oczekiwana cecha sieci
- Sieci bardzo skomplikowane (duża liczba bardziej złożony model – funkcję odwzorowania i są przez to bardziej skłonne do przeuczenia (nadmiernego dopasowania).
- Sieci z niewielką liczną neuronów (wag) mogą być niewystarczający silne do zamodelowania poszukiwanej funkcji odwzorowania wejść w wyjście (niedouczenie).
- Prawie zawsze większe sieci generują mniejszy błąd.

Dane dla sieci neuronowych

- Należy zebrać dane niezbędne w procesie uczenia:
  - pewną liczbę *przypadków*, z których każdy zawiera wartości dostępnych *zmiennych* wejściowych i wyjściowych
  - określić, które zmienne powinny zostać uwzględnione i ile a także jakie przypadki należy zgromadzić.
- Wybierając zmienne (przynajmniej początkowy ich zestaw) kierujemy się intuicją. Przy pierwszej próbie powinno się uwzględnić wszystkie zmienne, które, mogą mieć znaczenie, gdyż potem, na etapie projektowania, zbiór ten będzie redukowany.

http://www.statsoft.pl/textbook/glosfra_stat.html?http%3A%2F%2Fwww.statsoft.pl%2Ftextbook%2Fglosu.html

Dane dla sieci neuronowych

- Dane numeryczne są przeskalowywane do właściwego dla sieci przedziału (normalizacja).
- Wartości brakujące są zastępowane wartościami średnimi (lub innymi statystykami) obliczanymi na podstawie wartości zmiennych dostępnych w ciągu uczącym.
- Typ danych nominalnych takich jak *Płeć* = {*Mężczyzna*, *Kobieta*} zamienia się na wartości numeryczne. Przy dużej liczbie klas łączy się je w grupy.
- Wartości rzeczywiste poddaje się procesowi dyskretyzacji.

http://www.statsoft.pl/textbook/glosfra_stat.html?http%3A%2F%2Fwww.statsoft.pl%2Ftextbook%2Fglosu.html

Dane dla sieci neuronowych

- Rozmiar zbioru uczącego uzależnia się od rozmiaru sieci.
  - np.. że liczba przypadków powinna być dziesięciokrotnie większa od liczby połączeń występujących w sieci.
- W rzeczywistości liczba potrzebnych przypadków jest również uzależniona od złożoności zależności funkcyjnej poddawanej modelowaniu. Jednak zależność ta ma z reguły nieznaną postać trudno jest więc podać naprawdę wyczerpujący i trafny przepis, określający, ile elementów ciągu uczącego jest naprawdę nieodzownych.

http://www.statsoft.pl/textbook/glosfra_stat.html?http%3A%2F%2Fwww.statsoft.pl%2Ftextbook%2Fglosu.html

Zastosowania sieci neuronowych - predykcja

- Klasa zadań, w których na podstawie zbioru danych wejściowych określa się dane wyjściowe.
- *Przykłady*: ocena zdolności kredytowej, prognozy zmiany rynku, automatyczne sterowanie, gra na giełdzie.
- W wyniku procesu uczenia SSN nabywa zdolności określania wyjść dla zadanych wejść. Uczenie polega na prezentowaniu materiałów empirycznych z przeszłości.
- Nie trzeba znać, ani stawiać hipotez o naturze związku pomiędzy danymi wejściowymi a wyjściowymi - NIE TRZEBA ZNAĆ ZALEŻNOŚCI WEJŚCIE - WYJŚCIE.

Zastosowania sieci neuronowych - klasyfikacja i rozpoznanie

- Zadania, w których przydzielamy obiekty na podstawie pewnych cech do klas lub rozpoznajemy zgodność ze znanym wzorcem.
- *Przykłady*: rozpoznawanie znaków, twarzy, identyfikacja regionów zagrożonych bezrobociem.
- SSN przewiduje identyfikator klasy, do której może zaliczyć dane wejściowe.
- SSN sama znajduje istotne cechy podawanych danych, bez udziału jakiekolwiek teorii.

Zastosowania sieci neuronowych - kojarzenie danych

- Zadania polegające na kojarzeniu dużej liczby faktów.
- *Przykłady*: wnioskowanie powiązań ekonomicznych.
- SSN automatyzują proces wnioskowania na podstawie zgromadzonych danych, w których wyszukują trendy, istotne powiązania z bardzo dużej liczby różnych współczynników

Zastosowania sieci neuronowych - analiza danych

- Znalezienie związków mających charakter przyczynowy dla danych ze zbioru wejściowego.
- *Przykłady*: analiza danych ekonomicznych i przewidywanie intencji podmiotów gospodarczych, określanie przyczyn niepowodzeń.
- SSN dają nowe możliwości znalezienia zależności pomiędzy danymi mającymi faktyczną przyczynę, ale mogą być to również związki incydentalne.

Zastosowania sieci neuronowych - filtracja sygnałów

- Wycinanie sygnałów zaszumionego kanału, eliminacja zakłóceń o charakterze losowym.
- *Przykłady*: diagnostyka medyczna, telekomunikacja.
- Zadaniem SSN jest wyczyszczenie sygnału z przypadkowych zniekształceń. Sieć zapoznana z pewnym zestawem idealnych wzorców potrafi dopasowywać do nich zniekształcone wzorce.

Zastosowania sieci neuronowych – optymalizacja

- Szukanie optimów dla danego zagadnienia.
- *Przykład*: optymalne decyzje gospodarcze, optymalizacja ścieżek, układów cyfrowych.
- SSN poszukuje stanów stabilnych dla pewnego zestawu danych. Charakteryzują się one optymalnym rozłożeniem energii.