Wstęp do sieci neuronowych, wykład 04. Skierowane sieci neuronowe. Algorytmy konstrukcyjne dla sieci skierowanych

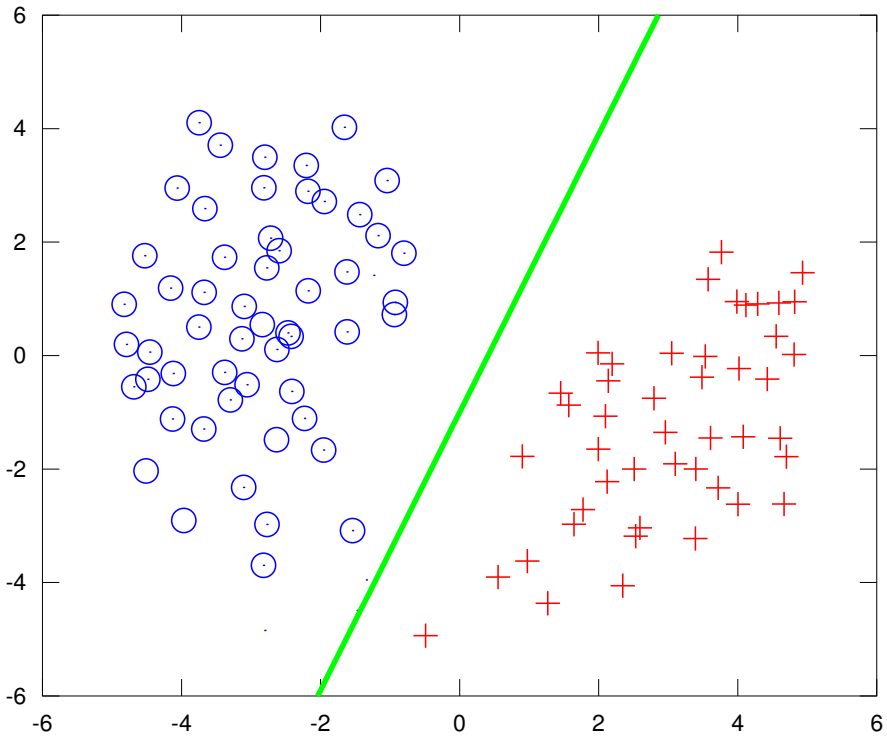
Maja Czoków, Jarosław Piersa

Wydział Matematyki i Informatyki, Uniwersytet Mikołaja Kopernika

2011-10-25

Ograniczenia pojedynczego neuronu

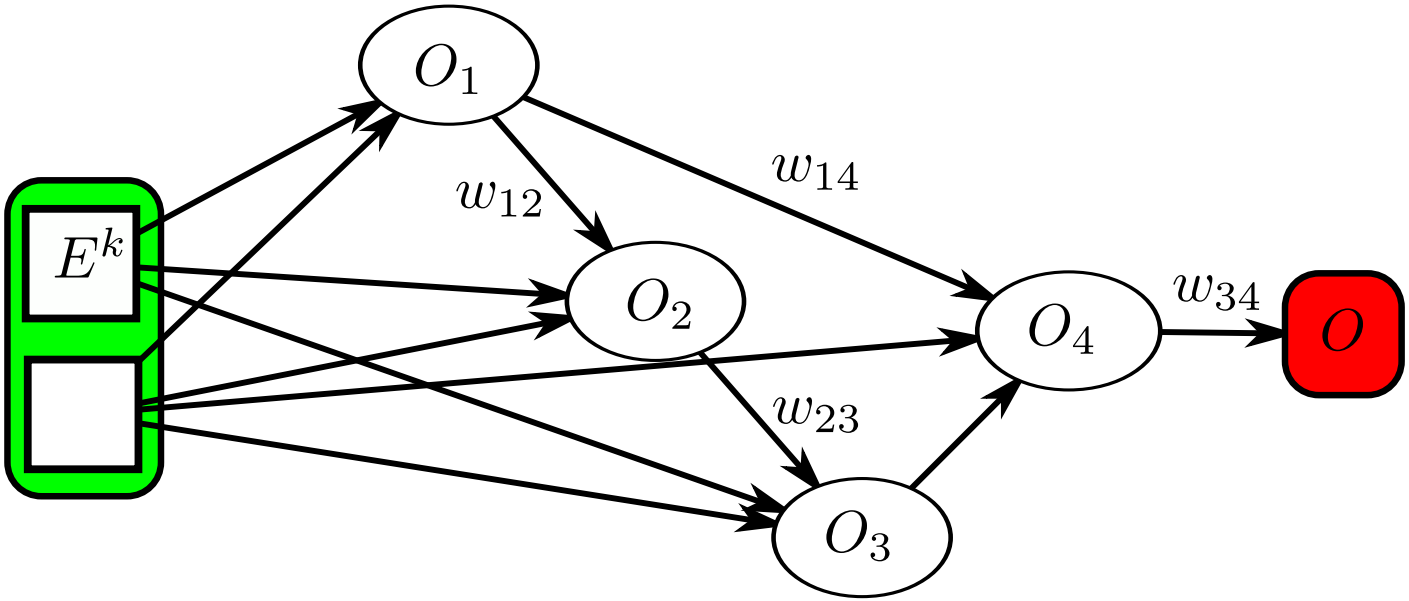
- Klasyfikacja wyłącznie problemów liniowo separowalnych,
- Co, gdy mamy trudniejszy problem?



Ograniczenia pojedynczego neuronu

- Albo bardzo zgrubne rozwiązania, albo perceptron całkowicie głupieje.
- Czy da się temu zaradzić?
- Znalezienie odpowiedzi zajęło ok 5 lat, jej spopularyzowanie kolejne 5.

Koncepcja sieci skierowanej



Koncepcja sieci skierowanej

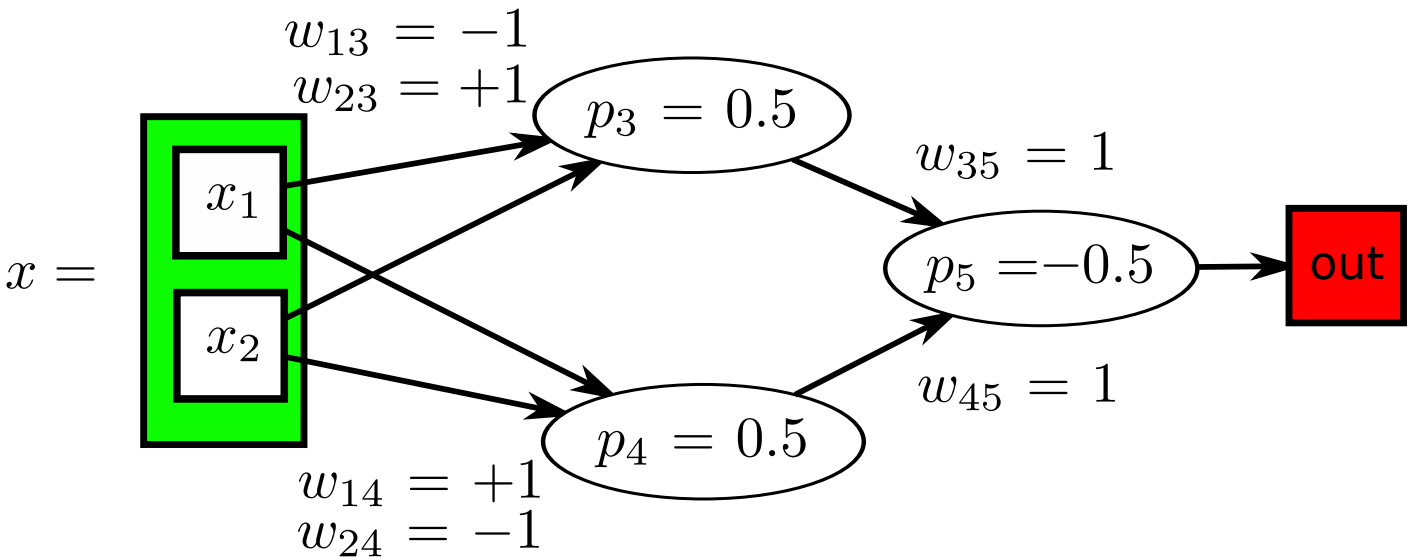
- Wejścia (liczby całkowite, rzeczywiste...) $x_1, .., x_n$,
- Zbiór perceptronów, każdy z własnym zestawem wag,
- Graf sieci — skierowany i acykliczny, rozpięty na perceptronach,
- Wejściami do perceptronów są: dane wejściowe do sieci oraz mogą być wyjścia z dowolnych perceptronów leżących wcześniej,
- Odpowiedzi perceptronów bez potomków (liście) są traktowane jako wyjścia z całej sieci.

Dynamika sieci skierowanej

- Działanie synchroniczne,
- Każdy neuron czeka, aż wszyscy rodzice zostaną obliczeni,
- Na koniec zwracana jest odpowiedź sieci (liczba lub wektor liczb),
- W implementacji: obliczenia po kolei według kolejności sortowania topologicznego,
- Jeżeli topologia na to pozwala, to można zrównoleglać obliczenia neuronów,

Przykład

Sieć rozwiązująca XOR



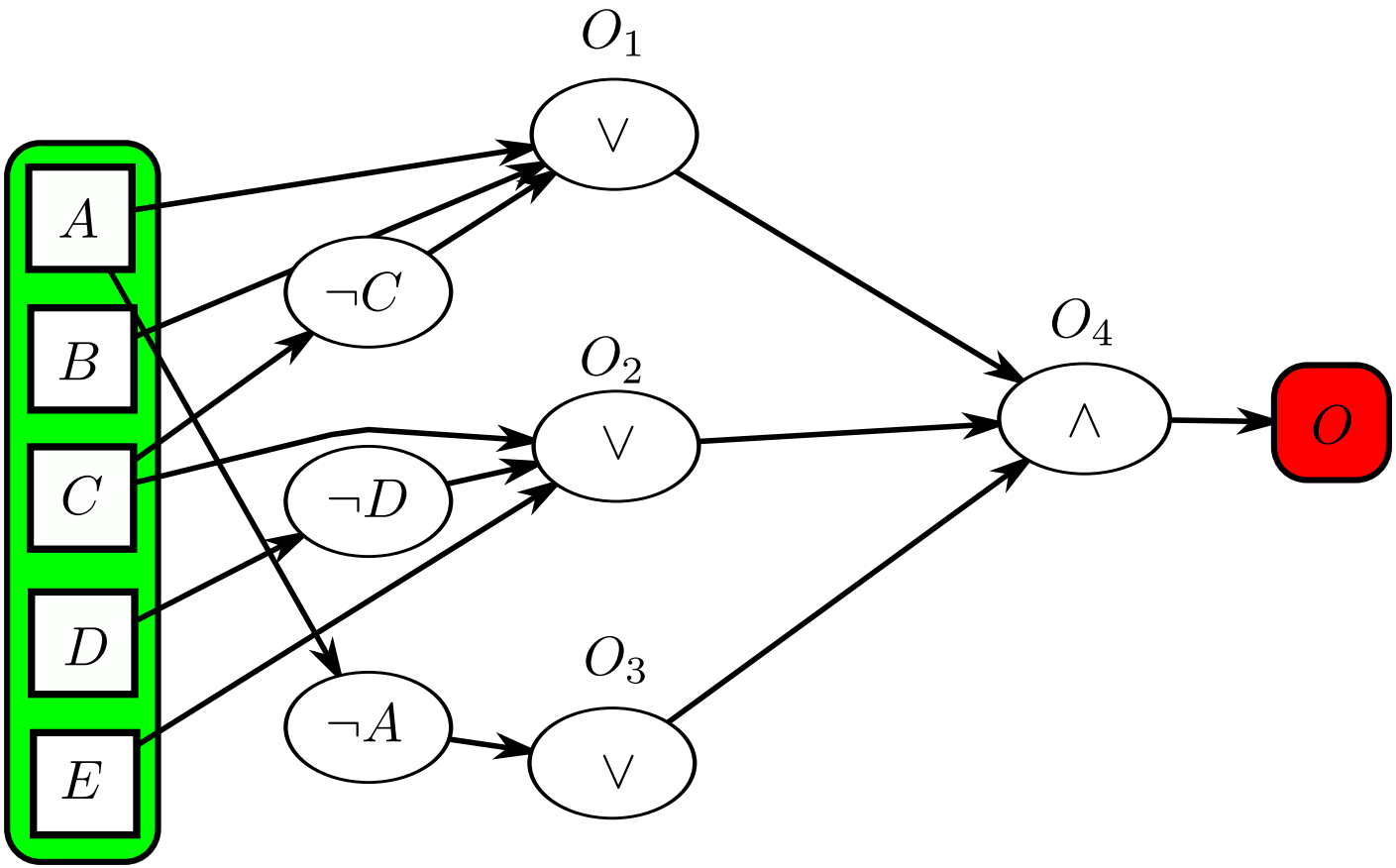
Funkcje logiczne

Koniunktywna postać normalna zdania logicznego.

- Zapis równoważnego zdania w postaci koniunkcji alternatyw zmiennych lub negacji zmiennych,
- $(A \vee B \vee \neg C) \wedge (C \vee \neg D \vee E) \wedge \neg A$
- Każde zdanie rachunku logiki boolowskiej da się zapisać w CNF (Conjunctive Normal Form),
- W szczególności A xor $B \quad = \quad (A \vee B) \wedge (\neg A \vee \neg B)$.

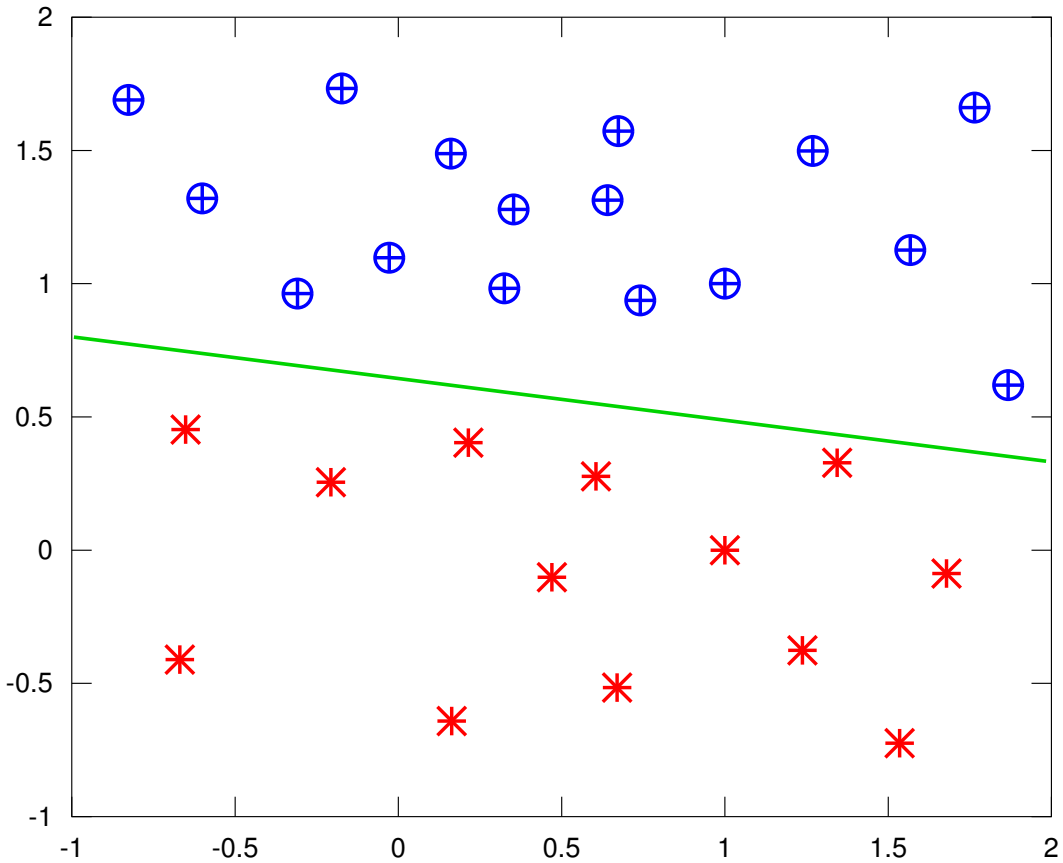
Funkcje logiczne

$$(A \vee B \vee \neg C) \wedge (C \vee \neg D \vee E) \wedge \neg A$$



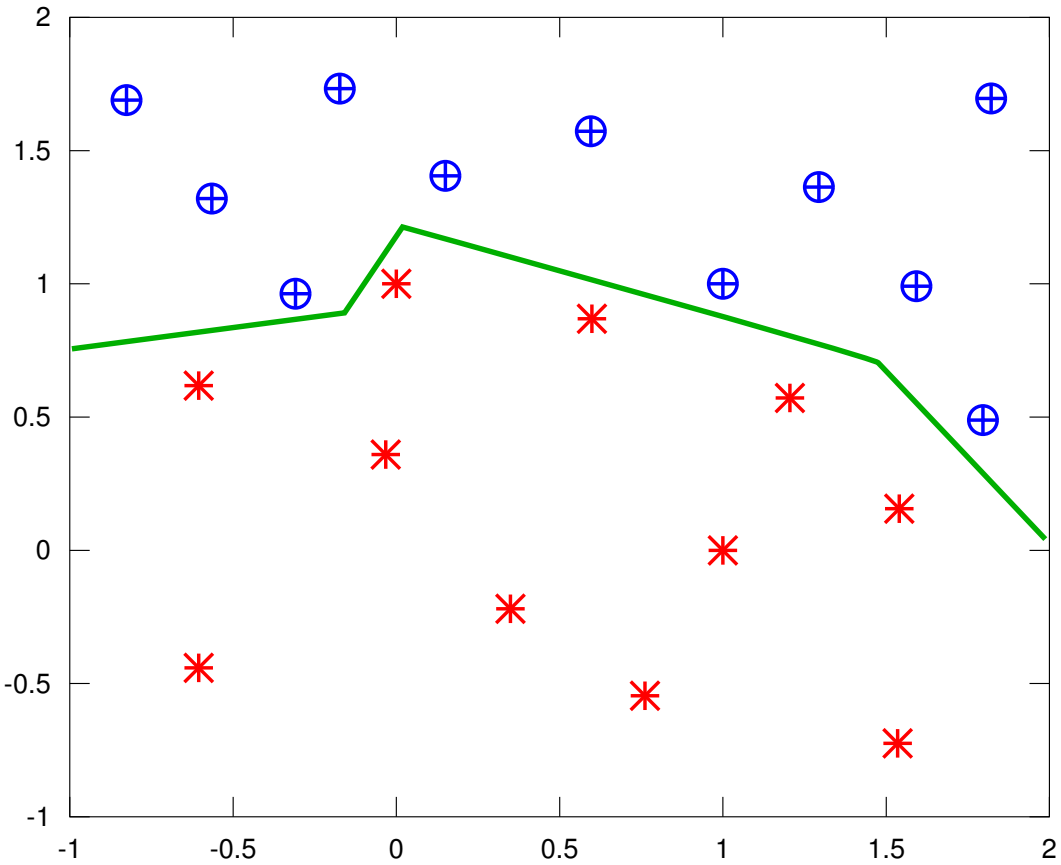
Funkcje na $\mathbb{R}^d$

Sieć bez warstw ukrytych jest w stanie przybliżać funkcje liniowo separowalne.



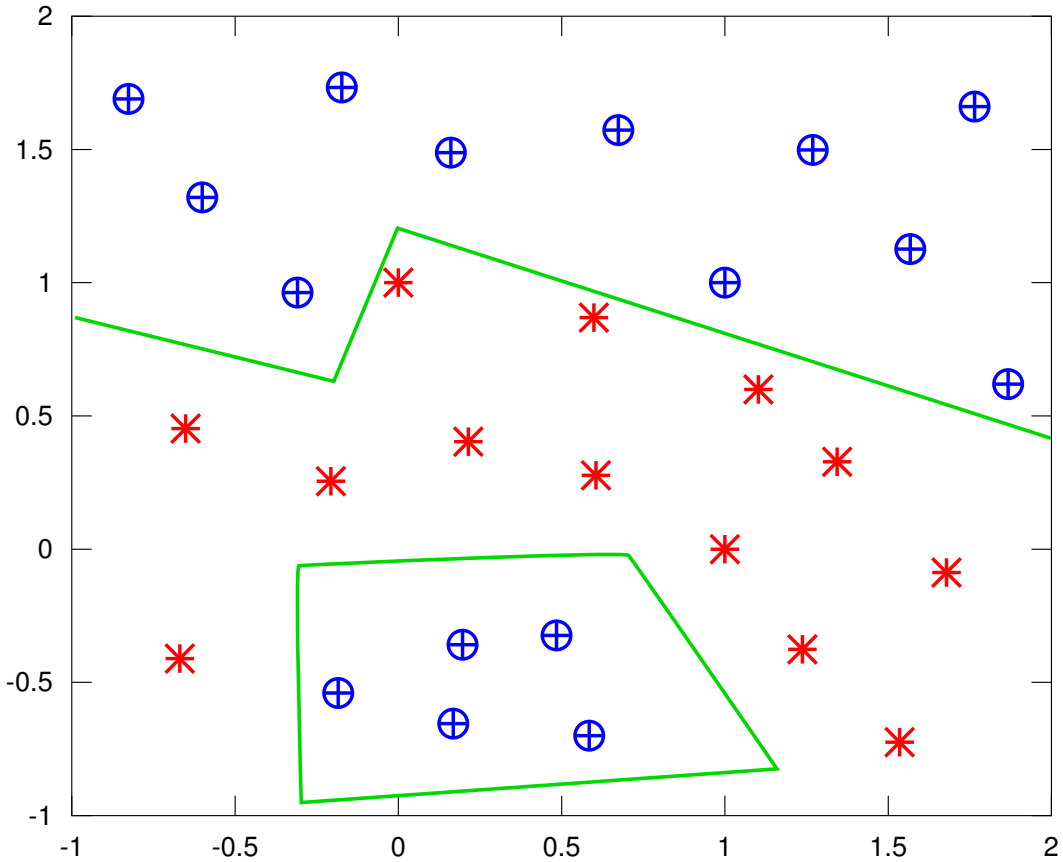
Funkcje na $\mathbb{R}^d$

Sieci z jedną warstwą ukrytą są w stanie przybliżać funkcje ciągłe.



Funkcje na $\mathbb{R}^d$

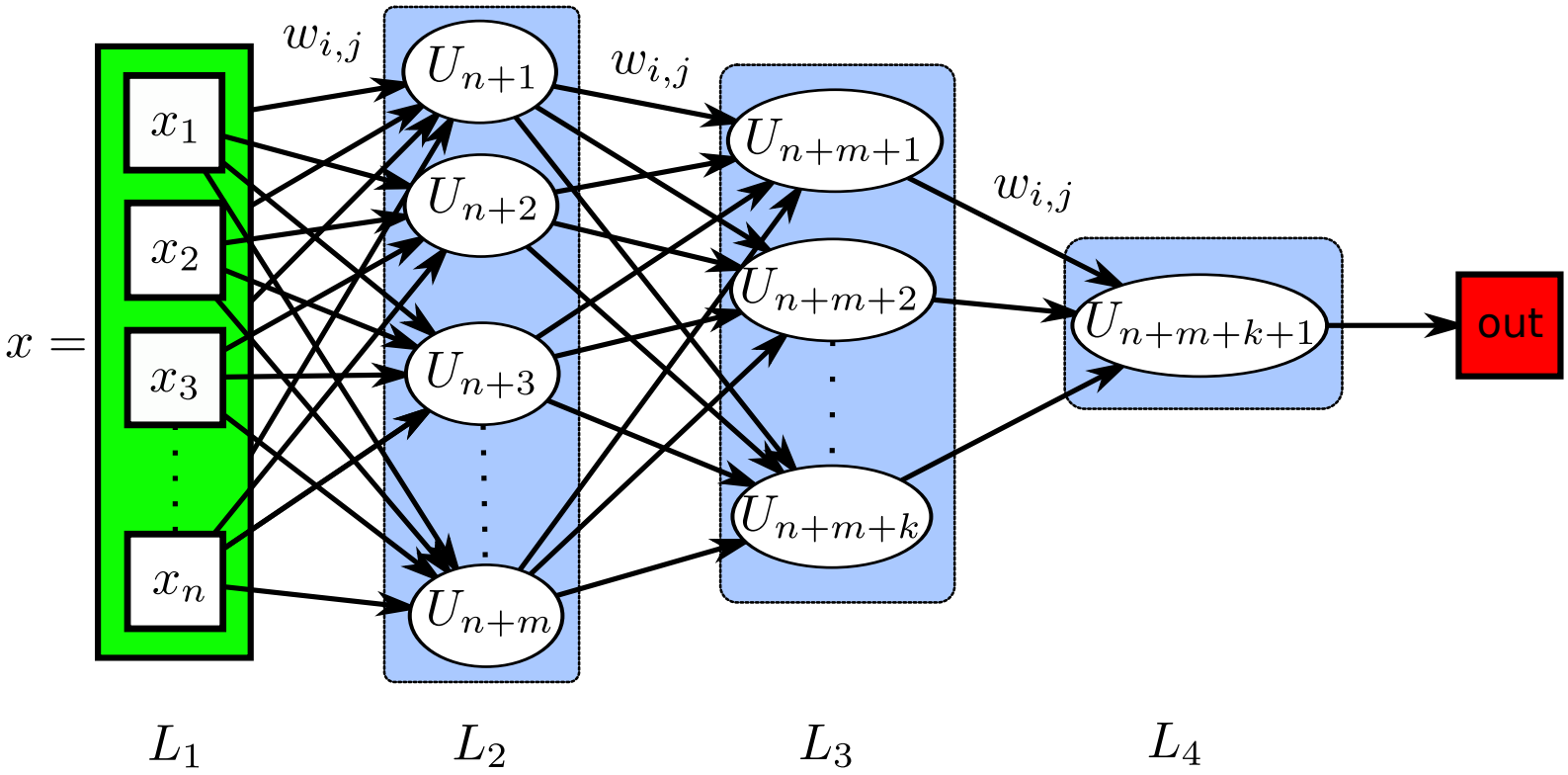
Sieci z dwiema lub więcej warstwami ukrytymi są w stanie przybliżać funkcje nieciągłe.



Jakie grafy sieci?

- Dające dobre rezultaty,
- Proste w implementacji (regularne, niezbyt gęste),
- Zdatne do zrównoleglenia (warstwowe).

Sieć (wielo) warstwowa ((multi) layer network)



Zagadnienie

Daną mamy listę przykładów uczących tj.

- punkt $E^k \in \mathbb{R}^n$,
- odpowiadającą mu poprawną klasyfikację $C^k \in \{-1, +1\}$.

Zagadnienie

Daną mamy listę przykładów uczących tj.

- punkt $E^k \in \mathbb{R}^n$,
- odpowiadającą mu poprawną klasyfikację $C^k \in \{-1, +1\}$.

Chcemy znaleźć skierowaną **sieć neuronów**, to jest graf wraz z wagami, który klasyfikuje poprawnie możliwie najwięcej spośród danych uczących.

Zagadnienie

Daną mamy listę przykładów uczących tj.

- punkt $E^k \in \mathbb{R}^n$,
- odpowiadającą mu poprawną klasyfikację $C^k \in \{-1, +1\}$.

Chcemy znaleźć skierowaną **sieć neuronów**, to jest graf wraz z wagami, który klasyfikuje poprawnie możliwie najwięcej spośród danych uczących.

$$w_{i,j} = ?$$

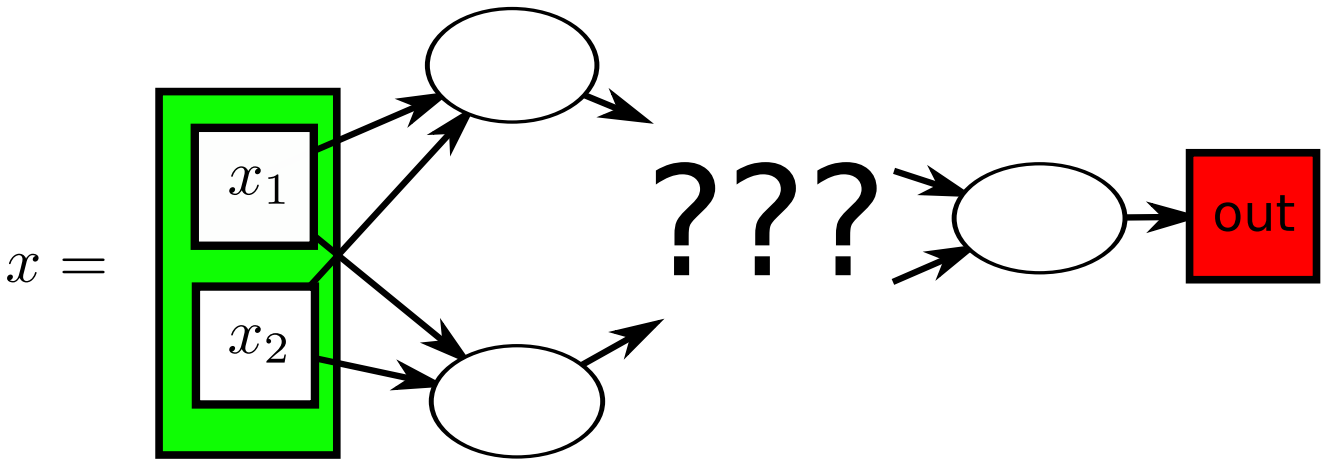
Zagadnienie

Daną mamy listę przykładów uczących tj.

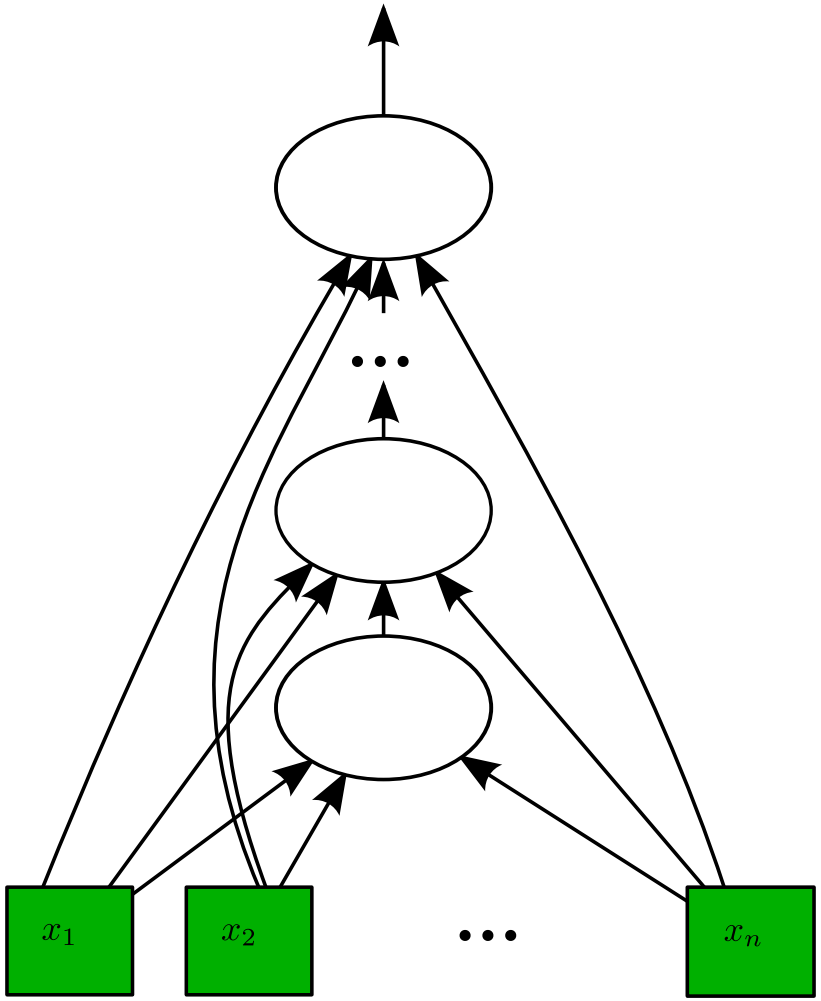
- punkt $E^k \in \mathbb{R}^n$,
- odpowiadającą mu poprawną klasyfikację $C^k \in \{-1, +1\}$.

Chcemy znaleźć skierowaną **sieć neuronów**, to jest graf wraz z wagami, który klasyfikuje poprawnie możliwie najwięcej spośród danych uczących.

$$w_{i,j} = ?$$



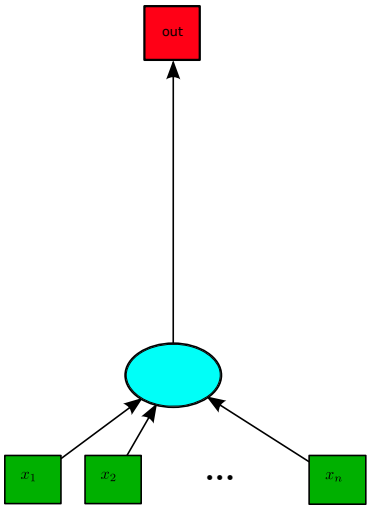
Algorytm Wieżowy



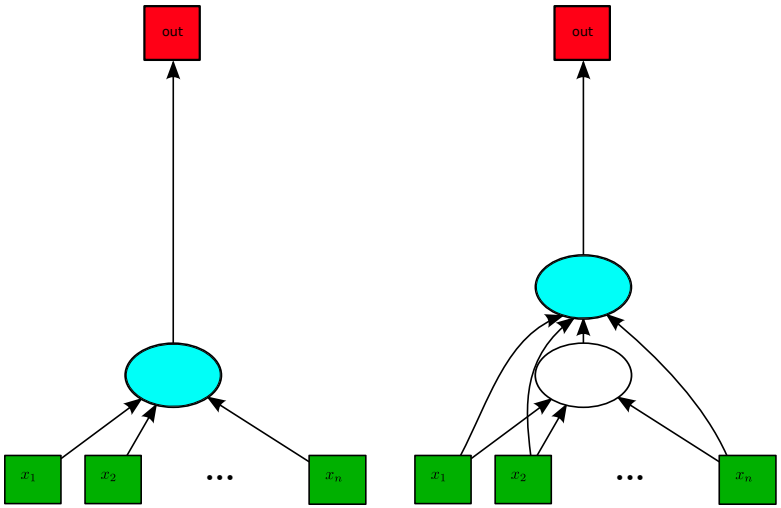
Algorytm wieżowy

1. Rozpocznij od sieci składającej się z pojedynczego perceptronu z progową funkcją aktywacji,
2. Naucz jedyny perceptron algorytmem kieszonkowym (z zapadką),
3. Powtarzaj aż do uzyskania zadowalającego rezultatu (tj. zadowalający poziom klasyfikacji, limit czasowy itp.)
   - Na szczyt wieży dodaj kolejny neuron. Jego wejściami będą dane uczące $E_1, .., E_n$ **oraz** wyjście neuronu leżącego bezpośrednio niżej,
   - Naucz szczytowy neuron algorytmem kieszonkowym (z zapadką), za $n+1$-sze wejścia przyjmij wyniki z neuronu niższego,
   - Jako wynik całej sieci zwracany będzie wynik nowo-dodanego perceptronu,
4. Zwróć wynikową sieć.

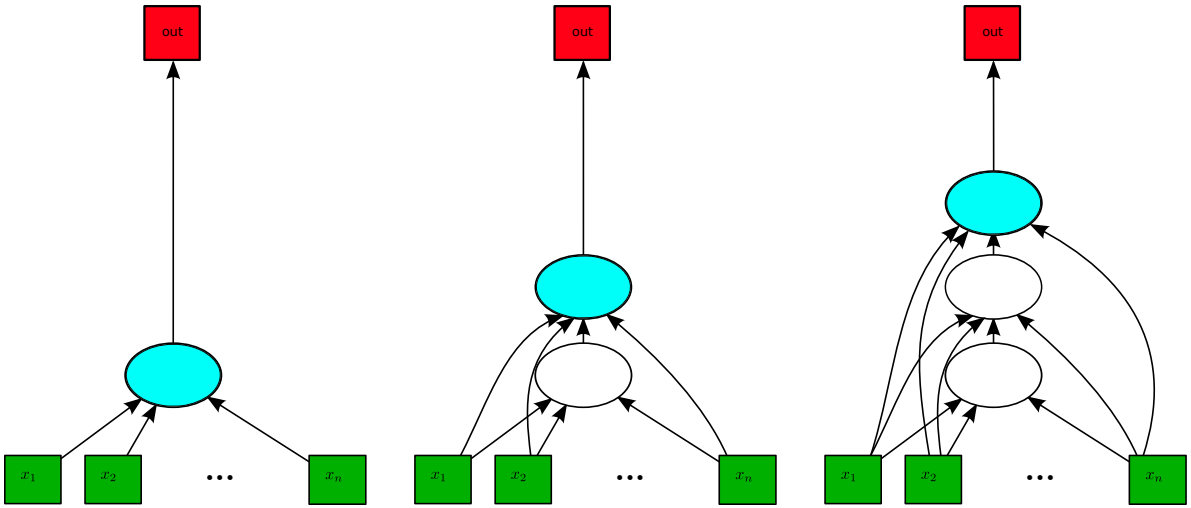
Algorytm wieżowy



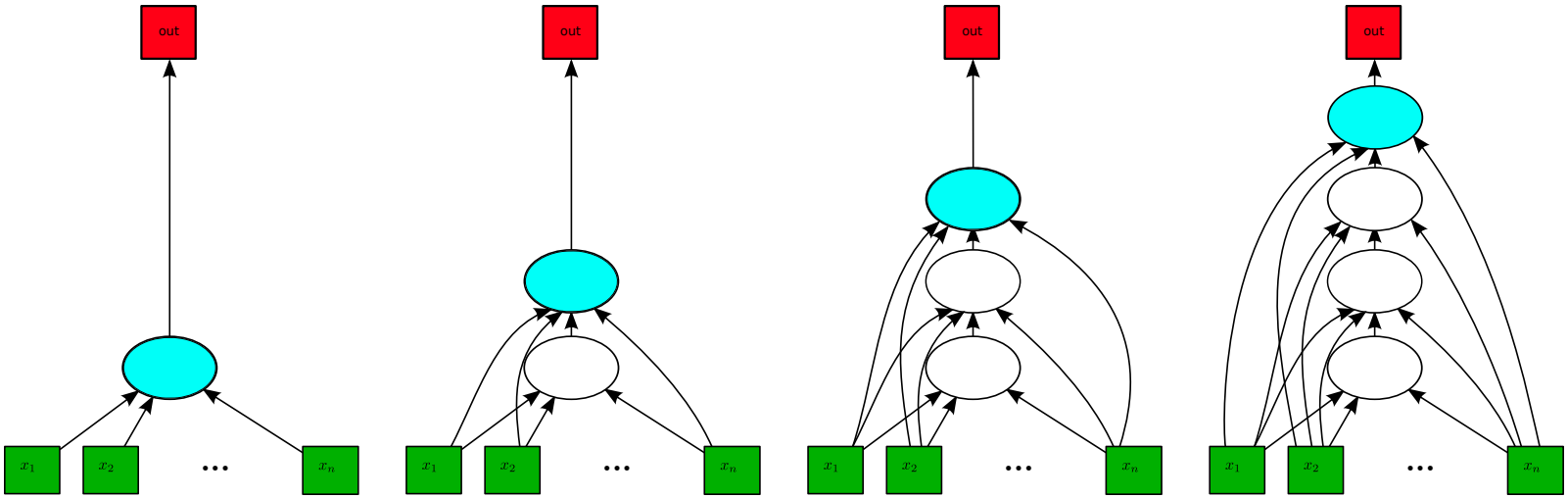
Algorytm wieżowy



Algorytm wieżowy



Algorytm wieżowy



Uzasadnienie

Przyjmijmy oznaczenia:

- O_j — jednostka j-ta,
- O_{j+1} — jednostka $j+1$-sza, ta jednostka będzie uczona,
- E^k — k-ty przykład uczący, źle klasyfikowany przez O_j,
- n — ilość wejść (wymiar danych),
- θ_{j+1} — próg jednostki O_{j+1},
- $0 < \varepsilon \ll 1$ — mała stała dodatnia,
- $w_{i,j+1}$ — wagi neuronu O_{j+1} stowarzyszone z danymi wejściowymi,
- $w_{j,j+1}$ — waga neuronu O_{j+1} stowarzyszona wejściem pochodzącym z jednostki O_j.

Uzasadnienie

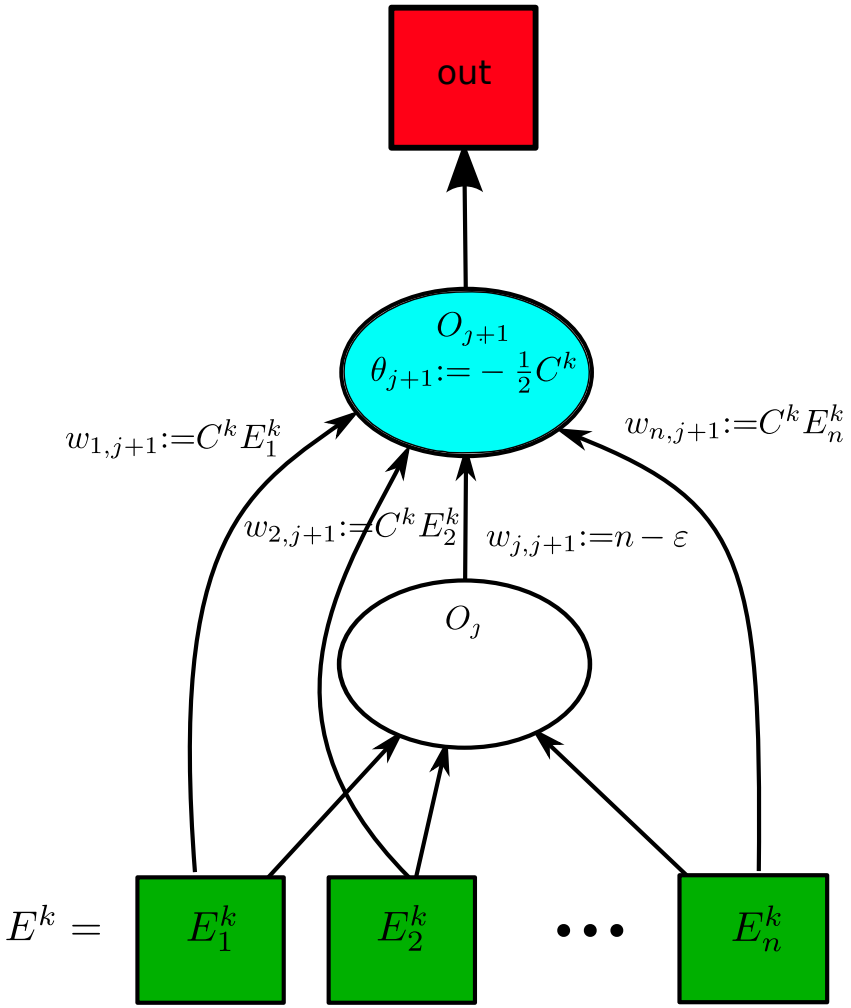
Przypiszmy:

- $w_{i,j+1} := C^k E_i^k$
- $w_{j,j+1} := n - \varepsilon$
- $\theta_{j+1} := -\frac{1}{2} C^k$

Uzasadnienie

Przypiszmy:

- $w_{i,j+1} := C^k E_i^k$
- $w_{j,j+1} := n - \varepsilon$
- $\theta_{j+1} := -\frac{1}{2} C^k$



Uzasadnienie

Pokażemy, że sieć z dodanym neuronem O_{j+1} jest w stanie zaklasyfikować poprawnie przynajmniej o jeden przykład uczący więcej.

Uzasadnienie

Pokażemy, że sieć z dodanym neuronem O_{j+1} jest w stanie zaklasyfikować poprawnie przynajmniej o jeden przykład uczący więcej.

- Dla przykładu E^k neuron O_{j+1} otrzyma na wejściu

$$(n-\varepsilon)(-1)C^k + \sum_i C^k E_i^k \cdot E_i^k = nC^k - nC^k + \varepsilon C^k = \varepsilon C^k$$

$$\theta_{j+1} = -\frac{1}{2}C^k,$$

więc sieć zwróci oczekiwaną odpowiedź na E^k.

Uzasadnienie

- Dla pozostałych przykładów $E^j, j \neq k$ neuron O_{j+1} zwróci tę samą odpowiedź co neuron O_j:

$$(n - \varepsilon)O_j + \sum_i C^j E_i^j \cdot E_i^k,$$

Uzasadnienie

- Dla pozostałych przykładów $E^j, j \neq k$ neuron O_{j+1} zwróci tę samą odpowiedź co neuron O_j:

$$(n - \varepsilon)O_j + \sum_i C^j E_i^j \cdot E_i^k,$$

  - Jeżeli dla każdego i $E_i^j = E_i^k$ oraz $C^k = C^k$, to oba przykłady E^k oraz E^j są te same i E^j też będzie poprawnie klasyfikowany,

Uzasadnienie

- Dla pozostałych przykładów $E^j, j \neq k$ neuron O_{j+1} zwróci tę samą odpowiedź co neuron O_j:

$$(n - \varepsilon)O_j + \sum_i C^j E_i^j \cdot E_i^k,$$

  - Jeżeli dla każdego i $E_i^j = E_i^k$ oraz $C^k = C^k$, to oba przykłady E^k oraz E^j są te same i E^j też będzie poprawnie klasyfikowany,
  - Jeżeli dla każdego i zachodzi $E_i^j = E_i^k$ oraz $C^k \neq C^k$, to oba E^k oraz E^j są te same, ale mają oczekiwane różne odpowiedzi — dane są sprzeczne i stuprocentowa klasyfikacja możliwa nie jest,

Uzasadnienie

- Dla pozostałych przykładów $E^j, j \neq k$ neuron O_{j+1} zwróci tę samą odpowiedź co neuron O_j:

$$(n - \varepsilon)O_j + \sum_i C^j E_i^j \cdot E_i^k,$$

  - Jeżeli dla każdego i $E_i^j = E_i^k$ oraz $C^k = C^k$, to oba przykłady E^k oraz E^j są te same i E^j też będzie poprawnie klasyfikowany,
  - Jeżeli dla każdego i zachodzi $E_i^j = E_i^k$ oraz $C^k \neq C^k$, to oba E^k oraz E^j są te same, ale mają oczekiwane różne odpowiedzi — dane są sprzeczne i stuprocentowa klasyfikacja możliwa nie jest,

Poza powyższymi przypadkami mamy

$$\left| \sum_i C^j E_i^j \cdot E_i^k \right| \leq n - 1.$$

Uzasadnienie

Suma ważona w jednostce O_{j+1} wyniesie zatem:

$$(n-\varepsilon)O_j + \sum_i C^j E_i^j \cdot E_i^k \leq (n-\varepsilon)O_j + (n-1)C^j,$$

Uzasadnienie

Suma ważona w jednostce O_{j+1} wyniesie zatem:

$$(n-\varepsilon)O_j + \sum_i C^j E_i^j \cdot E_i^k \leq (n-\varepsilon)O_j + (n-1)C^j,$$

Oraz próg:

$$\theta_{j+1} = -\frac{1}{2}C^k,$$

Uzasadnienie

Suma ważona w jednostce O_{j+1} wyniesie zatem:

$$(n-\varepsilon)O_j + \sum_i C^j E_i^j \cdot E_i^k \leq (n-\varepsilon)O_j + (n-1)C^j,$$

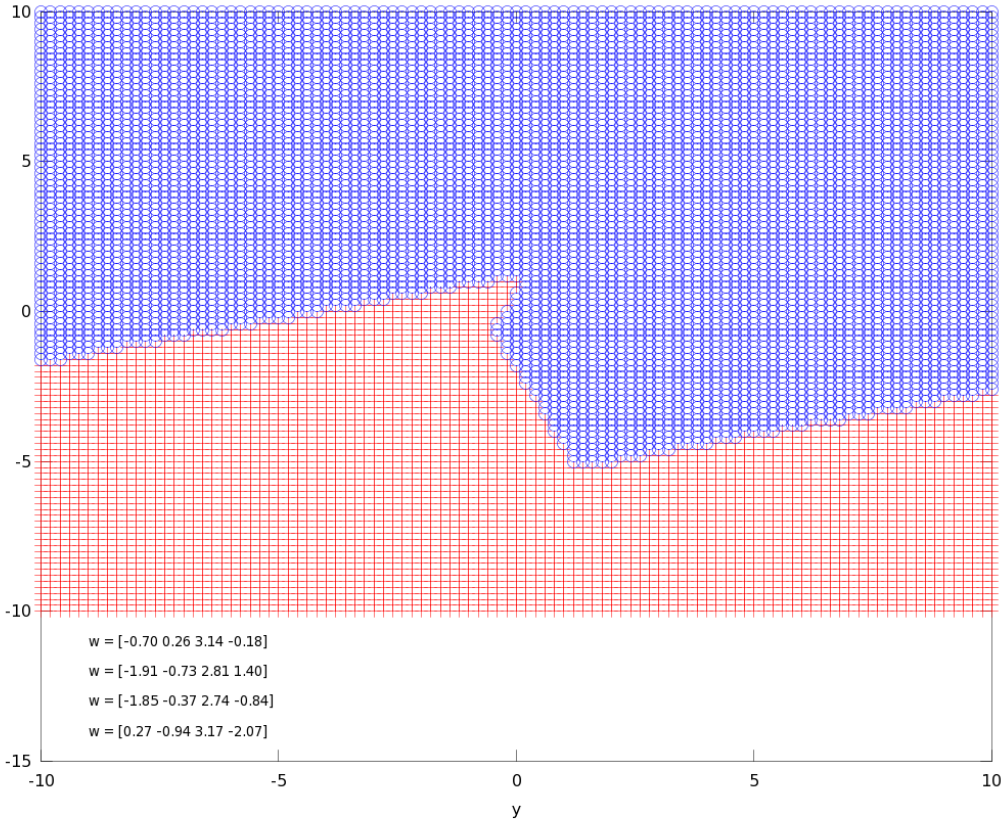
Oraz próg:

$$\theta_{j+1} = -\frac{1}{2}C^k,$$

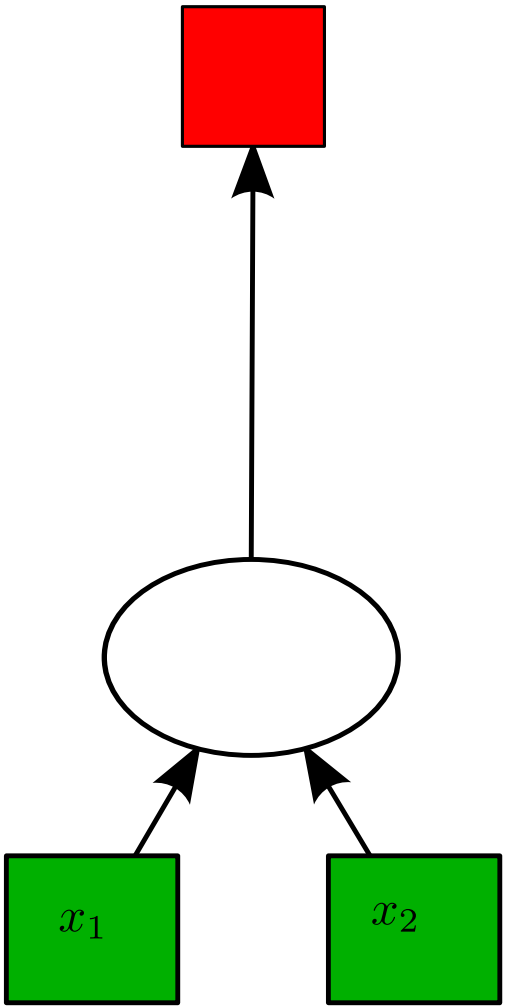
Jeżeli zatem ε nie będzie za duży, to zwrócona przez O_{j+1} odpowiedź będzie taka sama jak ta, którą zwrócił O_j.

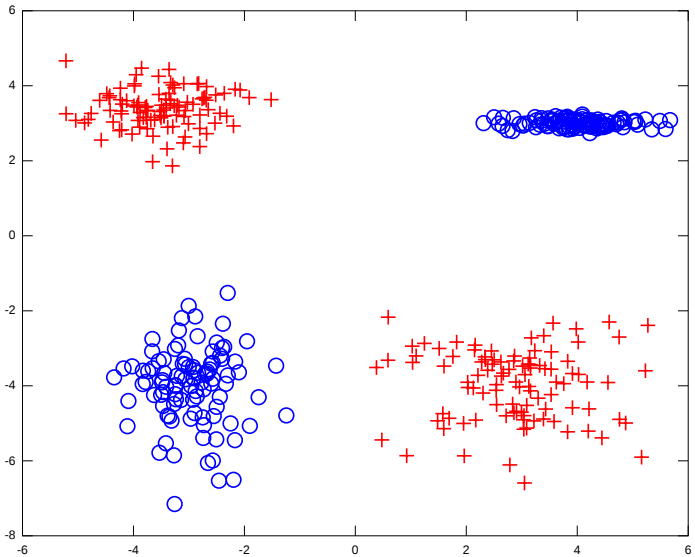
Mapy klasyfikacyjne

click

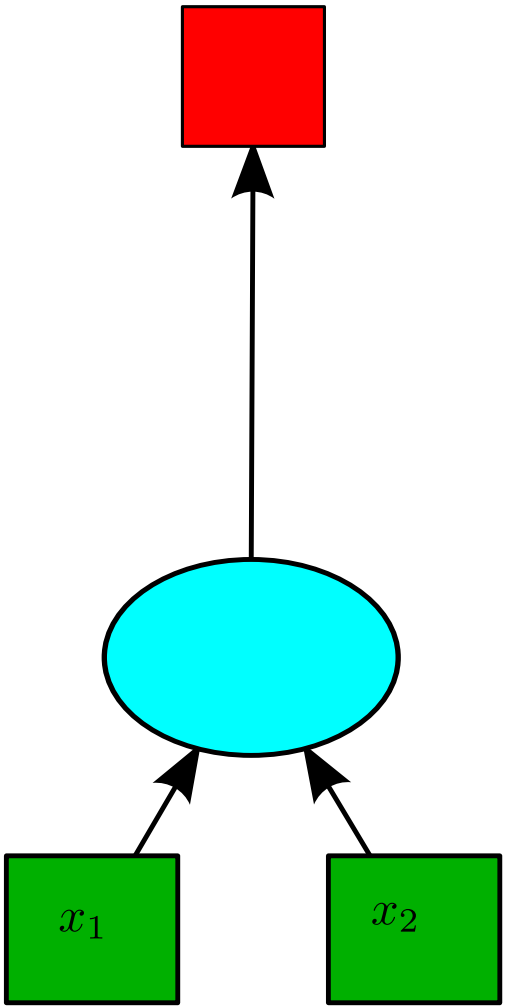


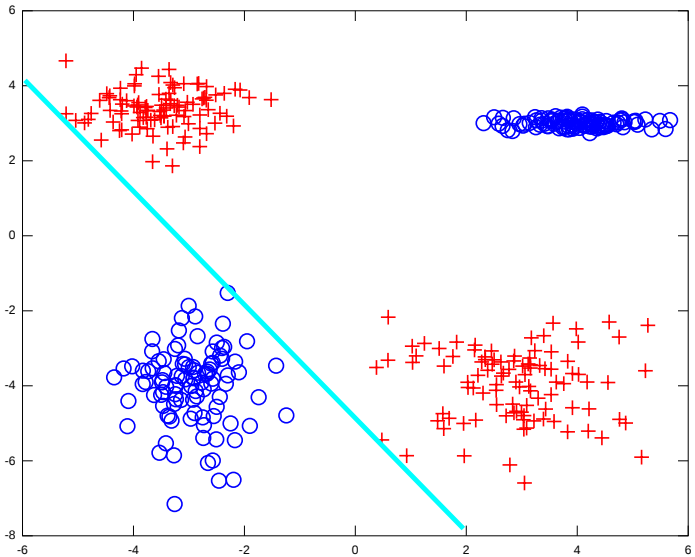
Przykład 1/4



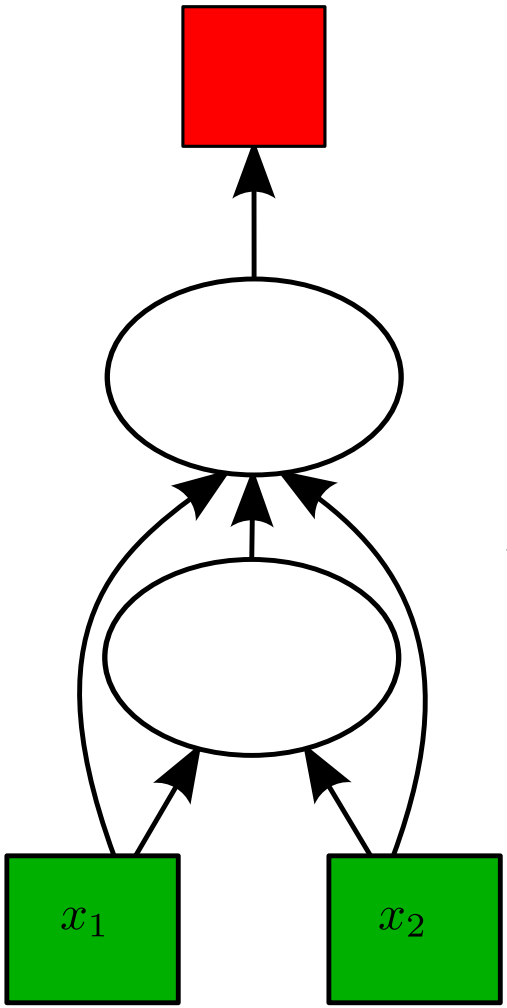


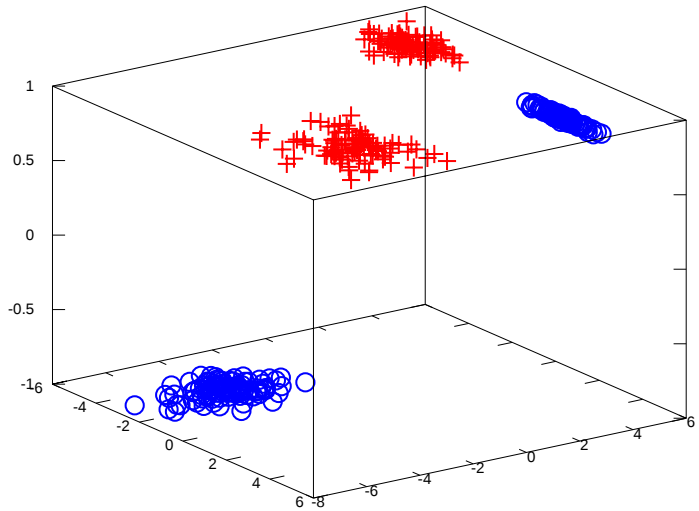
Przykład 2/4



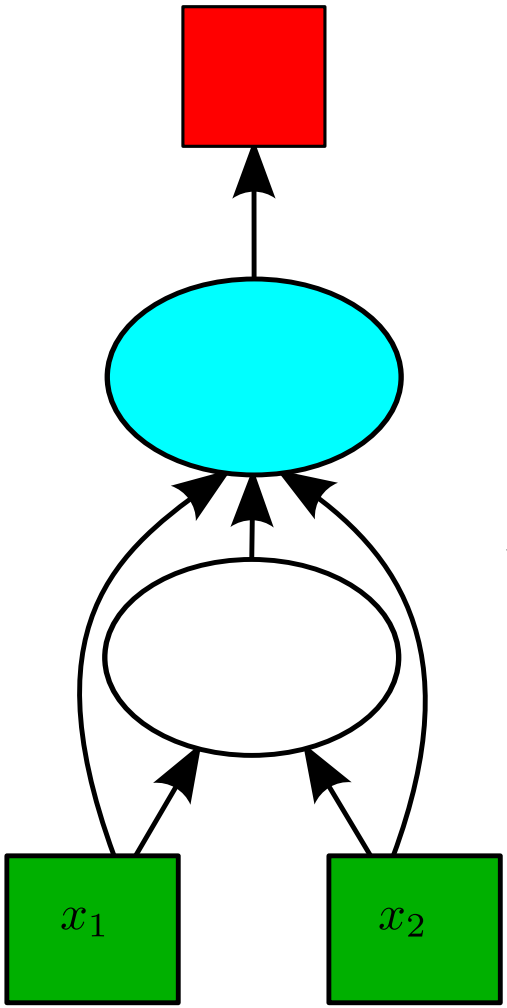


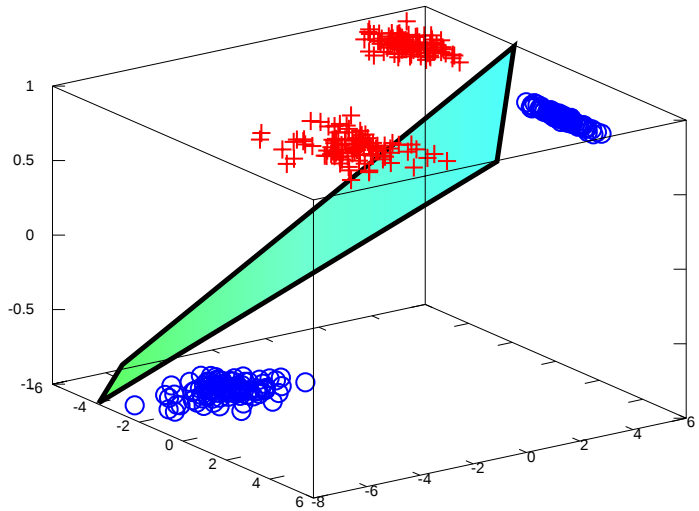
Przykład 3/4





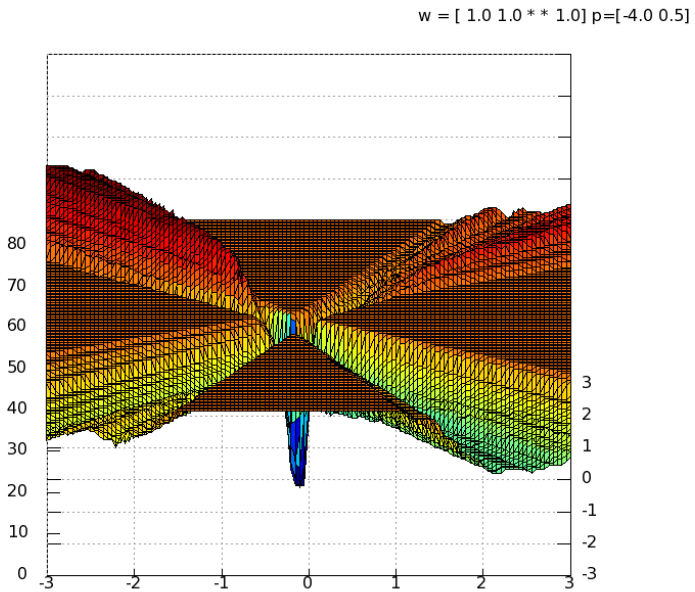
Przykład 4/4



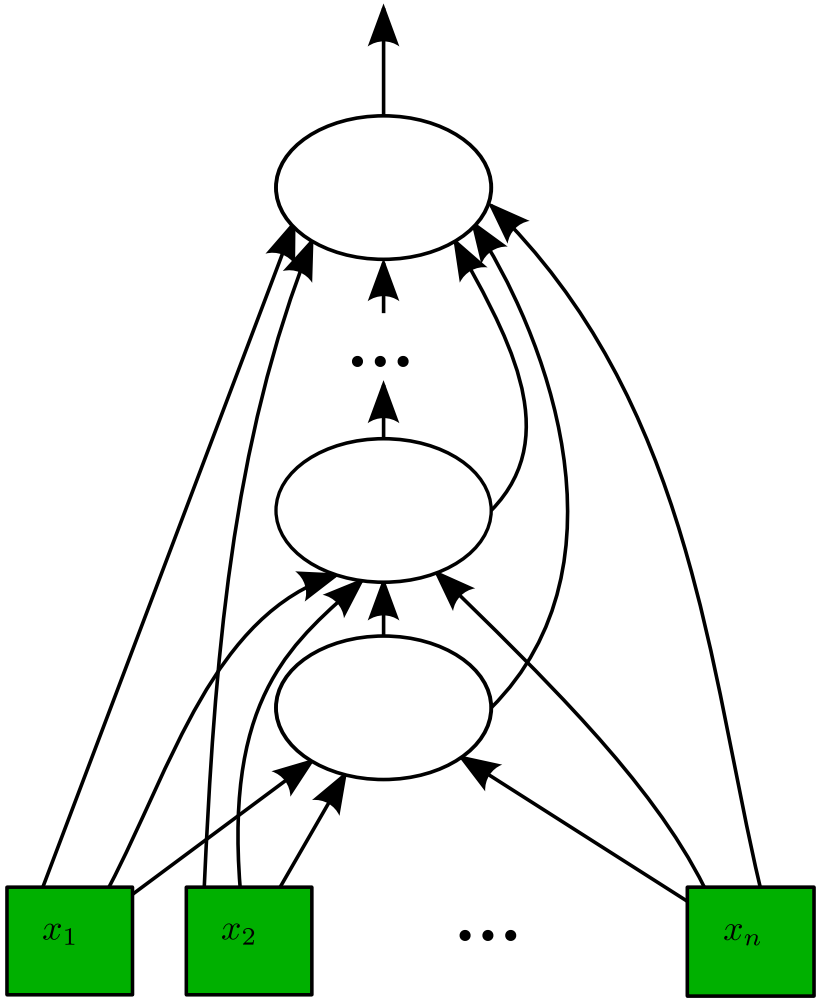


Przestrzeń wag

click



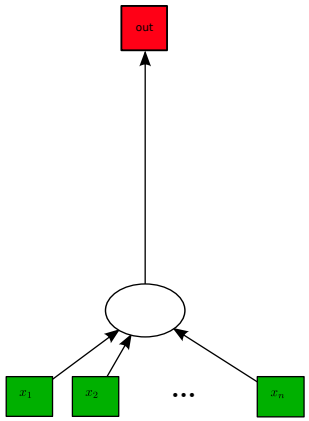
Algorytm piramidalny



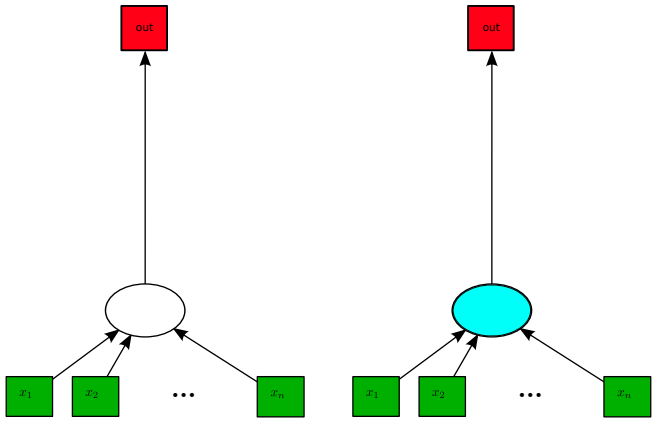
Algorytm piramidalny

1. Rozpocznij od sieci składającej się z pojedynczego perceptronu z progową funkcją aktywacji,
2. Naucz jedyny preceptron algorytmem kieszonkowym (z zapadką),
3. Powtarzaj aż do uzyskania zadowalającego rezultatu (tj. zadowalający poziom klasyfikacji, limit czasowy itp.)
   - Na szczyt wieży dodaj kolejny neuron. Jego wejściami będą dane uczące $E_1, .., E_n$ **oraz** wyjście **wszystkich leżących niżej neuronów**,
   - Naucz szczytowy neuron algorytmem kieszonkowym (z zapadką), za $n+1, ..., n+l$-te wejścia przyjmij wyniki z neuronów niższych,
   - Jako wynik całej sieci zwracany będzie wynik nowo-dodanego perceptronu,
4. Zwróć wynikową sieć.

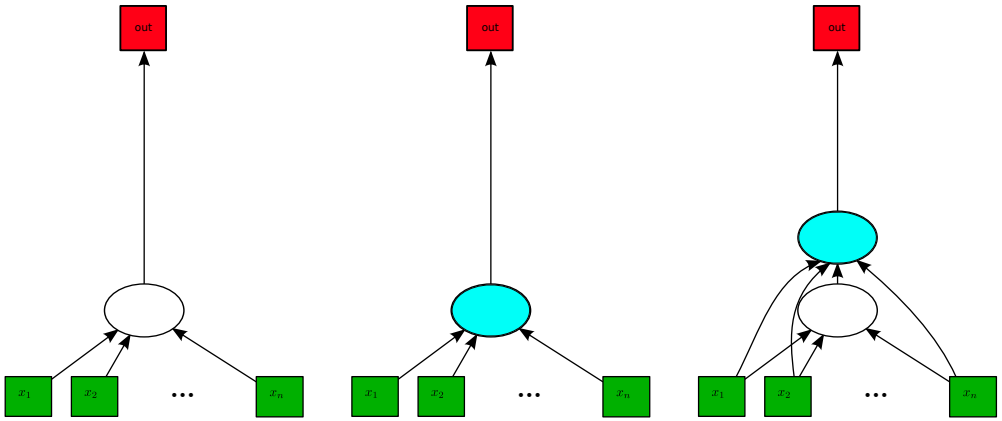
Algorytm piramidalny



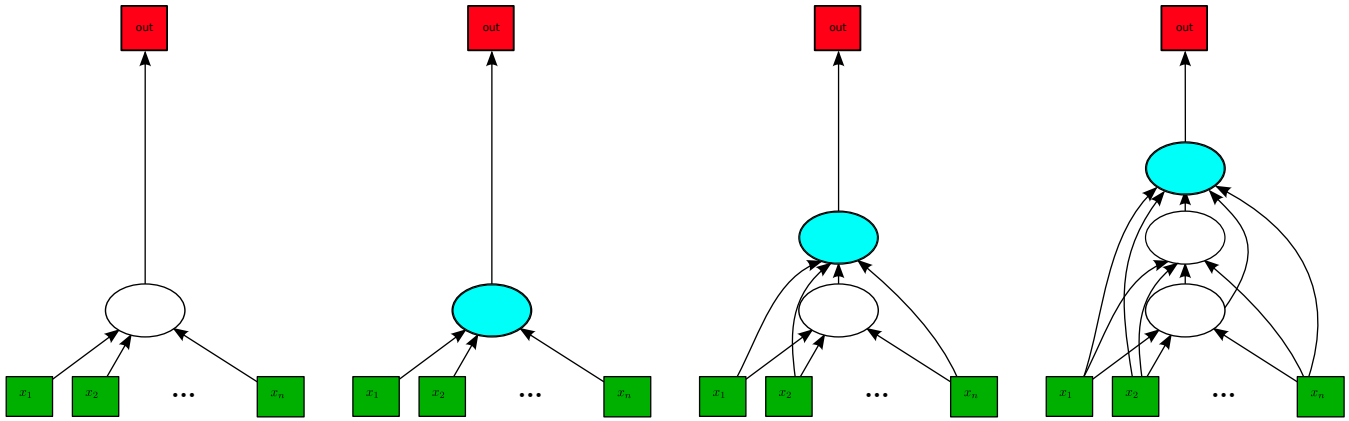
Algorytm piramidalny



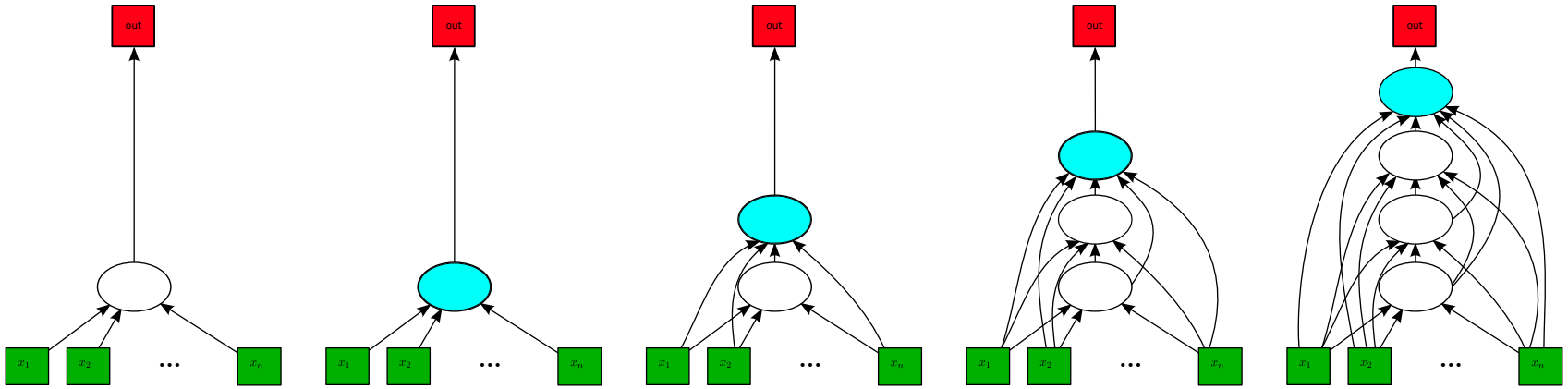
Algorytm piramidalny



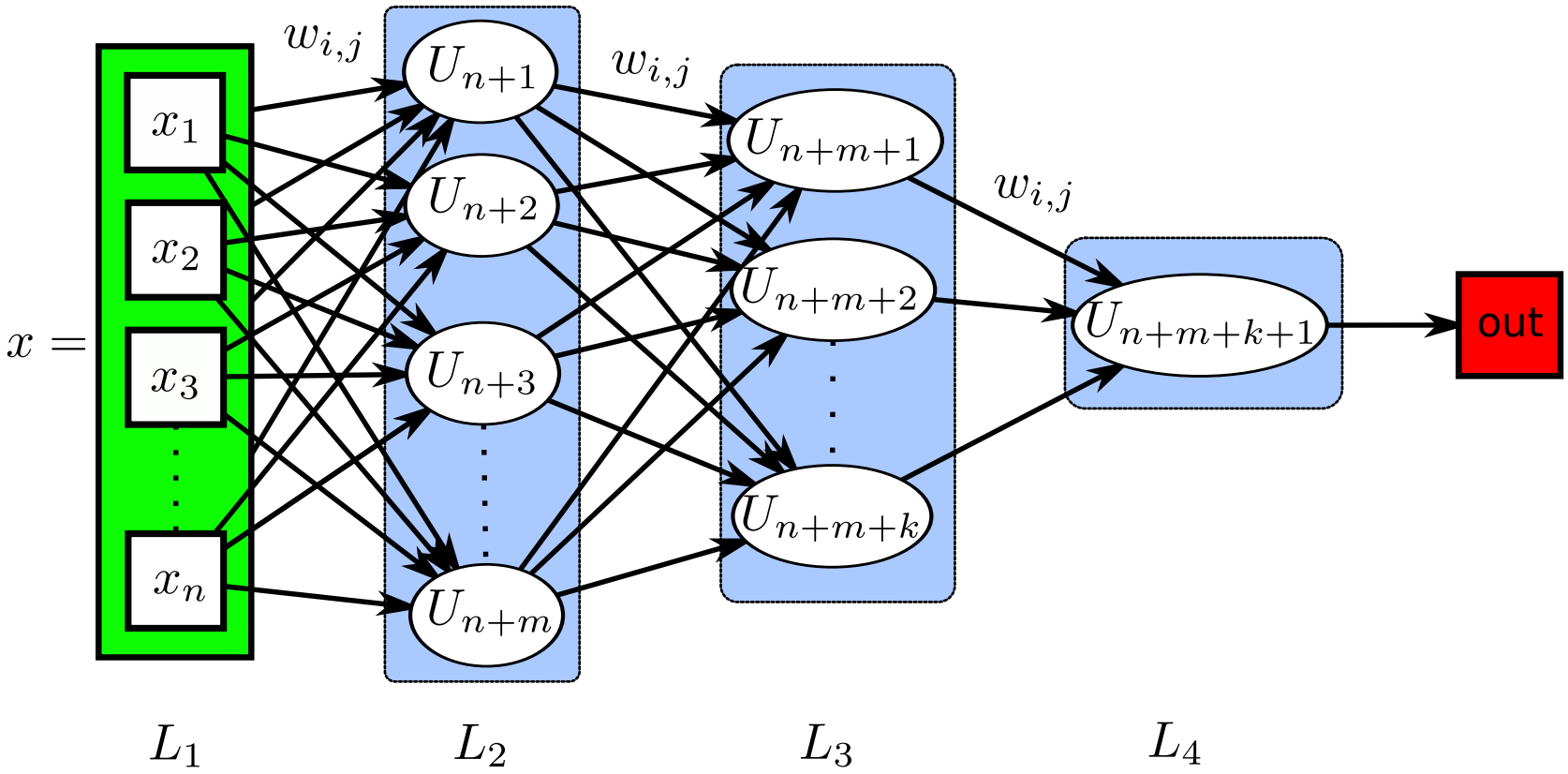
Algorytm piramidalny



Algorytm piramidalny



Algorytm kafelkowy



Algorytm kafelkowy

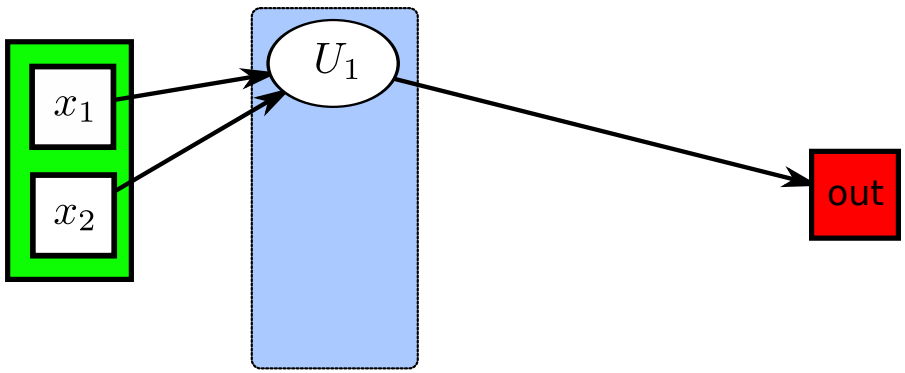
- Rozpocznij od sieci składającej się z warstwy wejściowej $L = 1$,
- Dodaj do sieci $L + 1$ kafel, na razie składający się tylko z jednego neuronu. Dodany neuron za wejścia przyjmuje wszystkie wyjścia z kafla L. Za wyjście sieci przyjmij wyjście z nowego neuronu,
- Naucz dodaną jednostkę algorytmem kieszonkowym z zapadką. Jeżeli sieć klasyfikuje poprawnie wszystkie przykłady, to zakończ zwracając wynikową sieć,
- Jeżeli nie, to dodaj do $L + 1$-go kafla kolejny neuron. Naucz go algorytmem kieszonkowym z zapadką, ale tylko na takim zbiorze, że:
  - poprzednie neurony w $L + 1$ kaflu dają na tych przykładach tę samą kombinację odpowiedzi (tj. z punktu widzenia sieci jest to ta sama klasa),
  - oczekujemy, że mają być to przykłady z różnych klas,
  - spośród wszystkich podzbiorów spełniających dwa powyższe warunki wybrany powinien być najliczniejszy.

Algorytm kafelkowy cd.

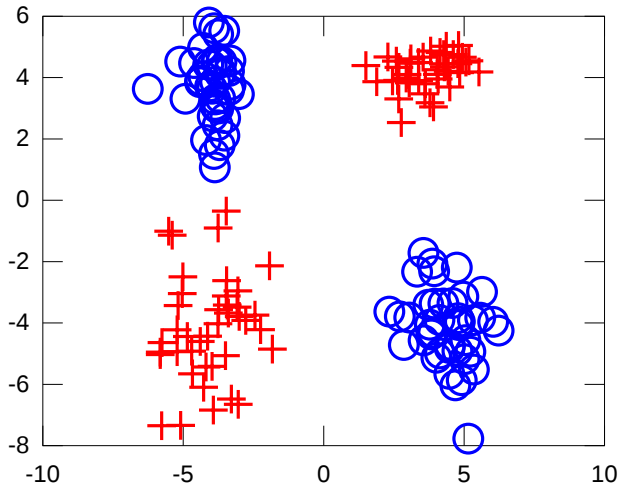
- Jeżeli kafel poprawnie klasyfikuje wszystkie przykłady (tj. różnym kategoriom przypisuje różne zestawy odpowiedzi), to wróć do 2 (dodaj nowy kafel), jeżeli nie to wróć do 4 (dodaj nową jednostkę w tym samym kaflu),
- Zwróć wynikową sieć (oczekujemy odpowiedzi binarnej, więc ostatni kafel powinien liczyć tylko jedną jednostkę).

Przykład 1/8

Sieć

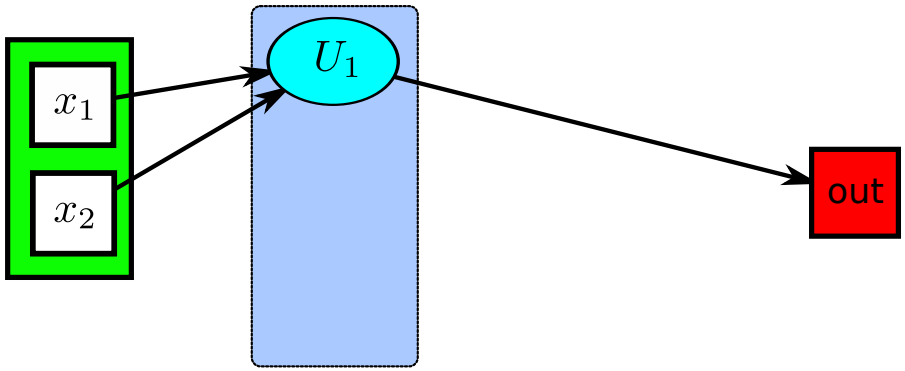


Dane

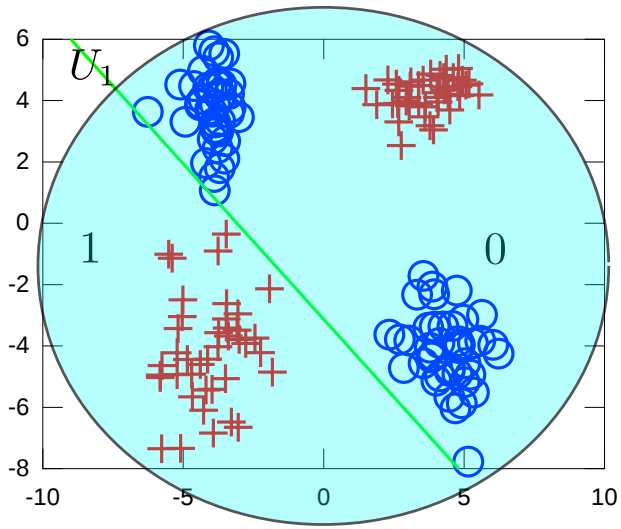


Przykład 2/8

Sieć

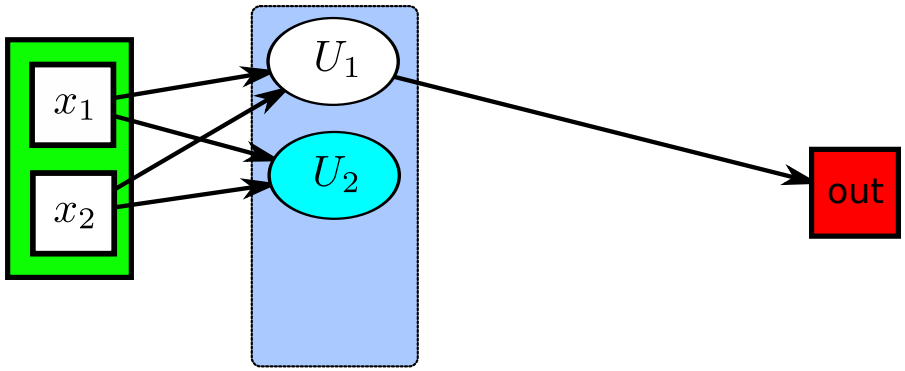


Dane

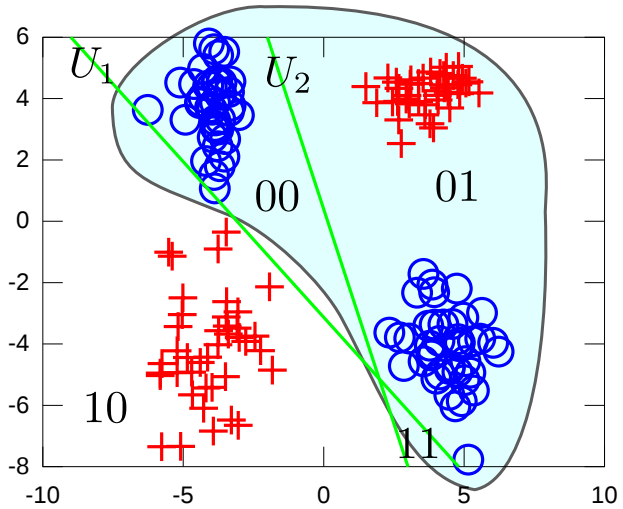


Przykład 3/8

Sieć

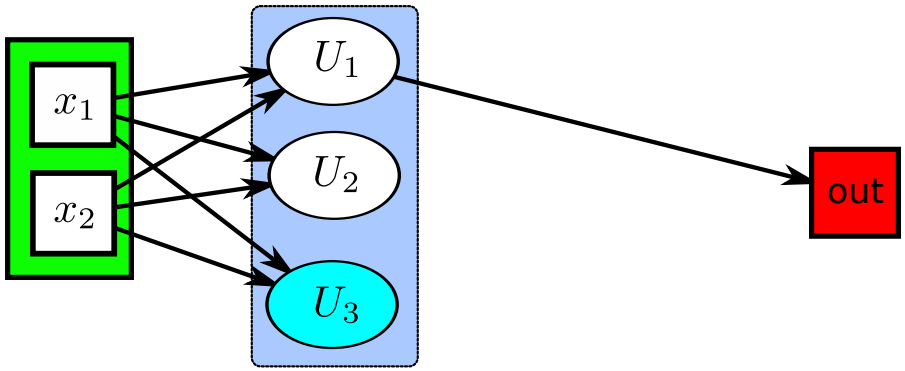


Dane

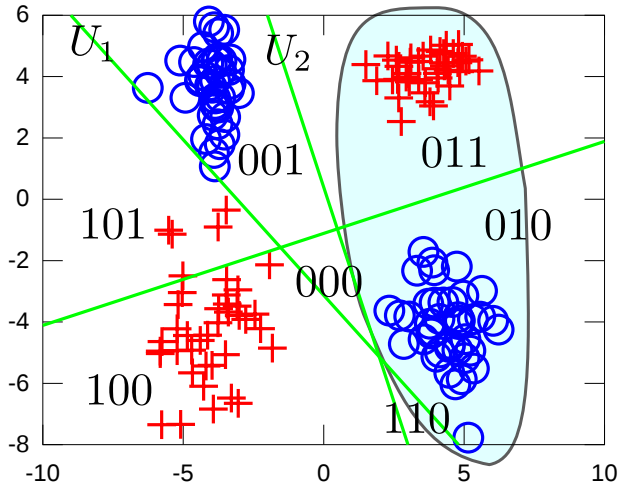


Przykład 4/8

Sieć

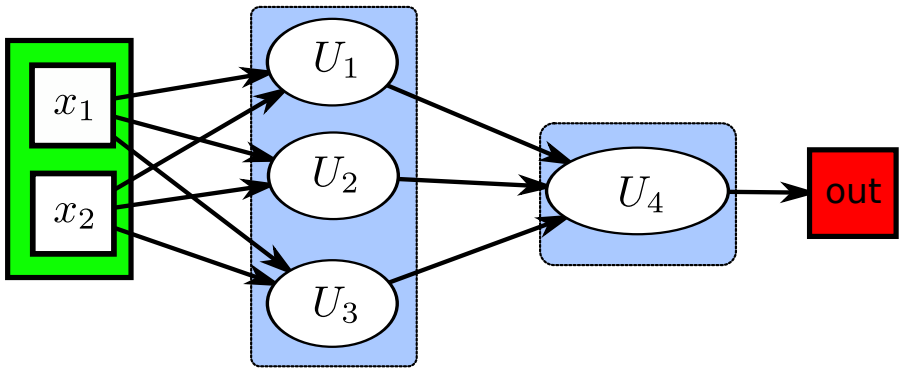


Dane

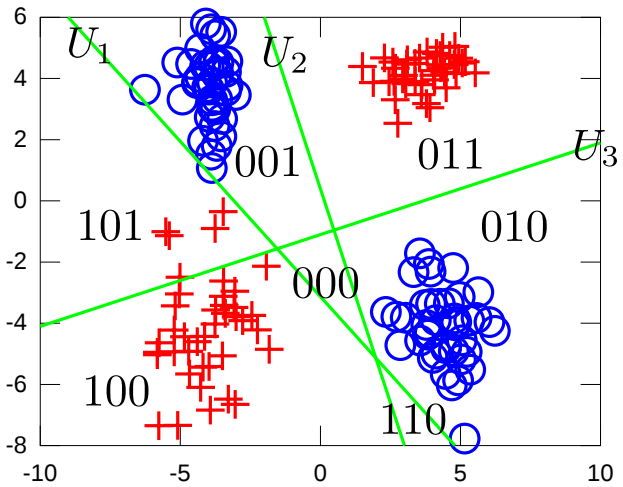


Przykład 5/8

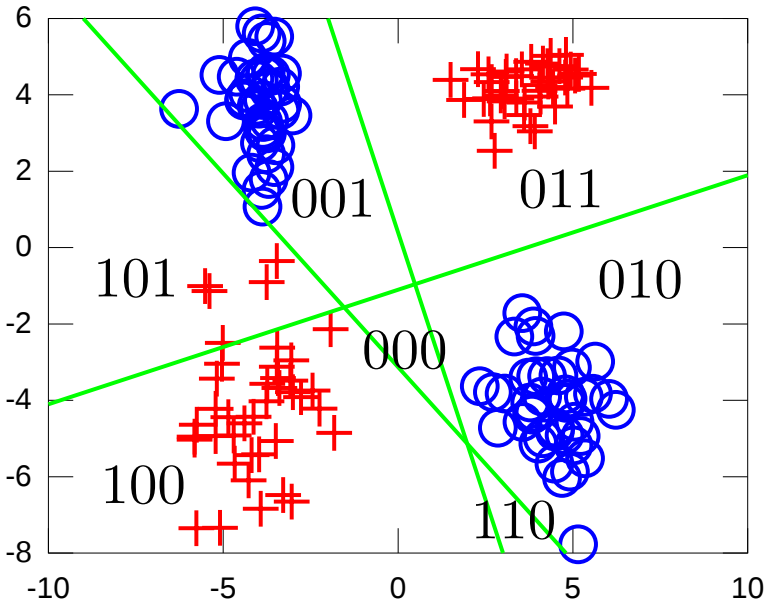
Sieć



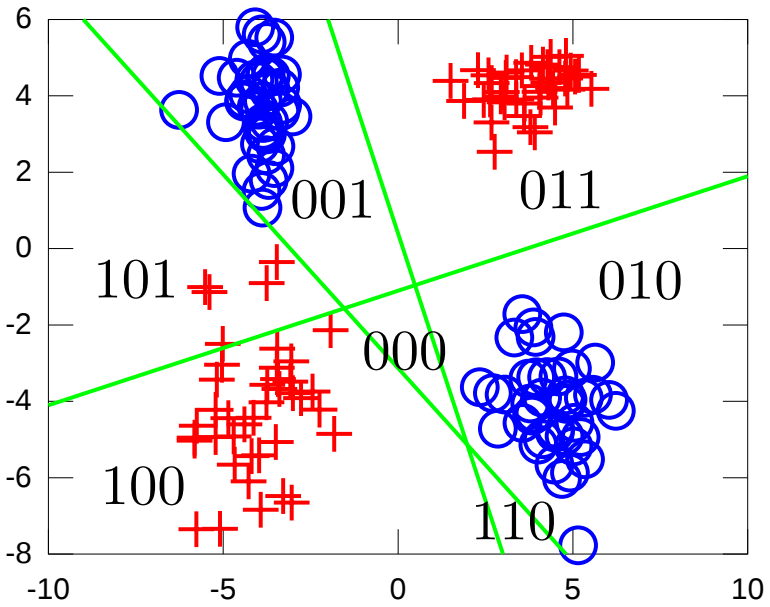
Dane

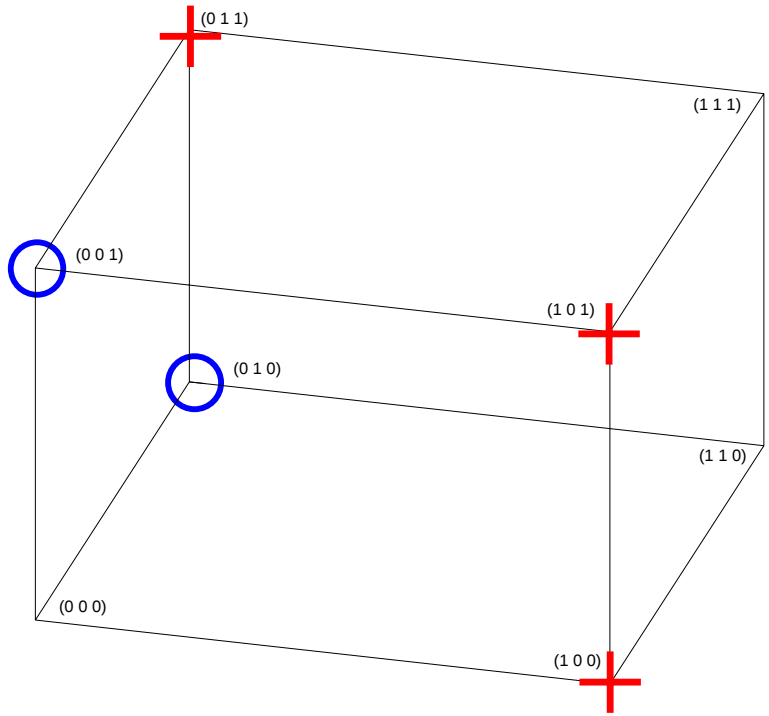


Przykład 6/8



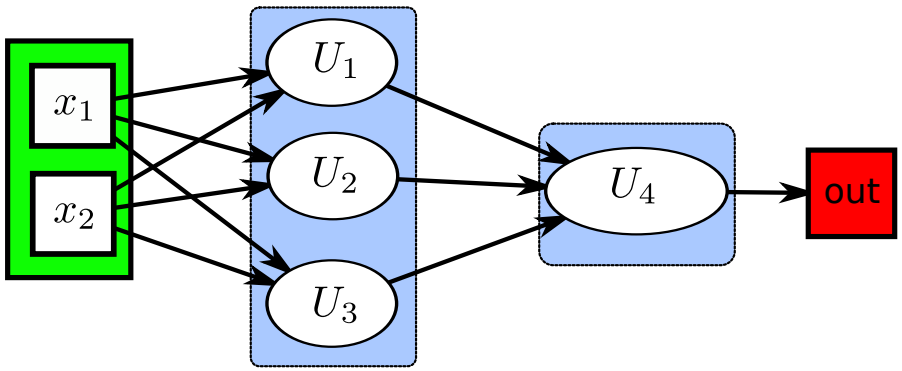
Przykład 6/8


6
4
2
0
-2
-4
-6
-8
-10
-5
0
5
10
001
011
101
010
000
100
110

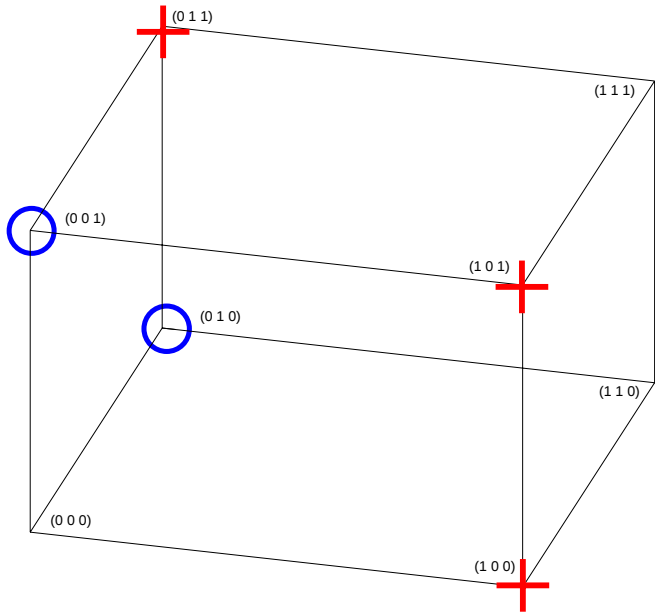

(0 1 1)
(1 1 1)
(0 0 1)
(1 0 1)
(0 1 0)
(1 1 0)
(0 0 0)
(1 0 0)

Przykład 7/8

Sieć

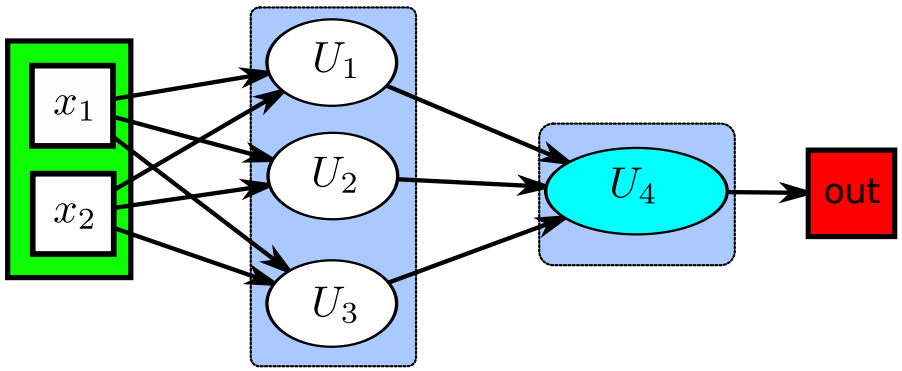


Dane

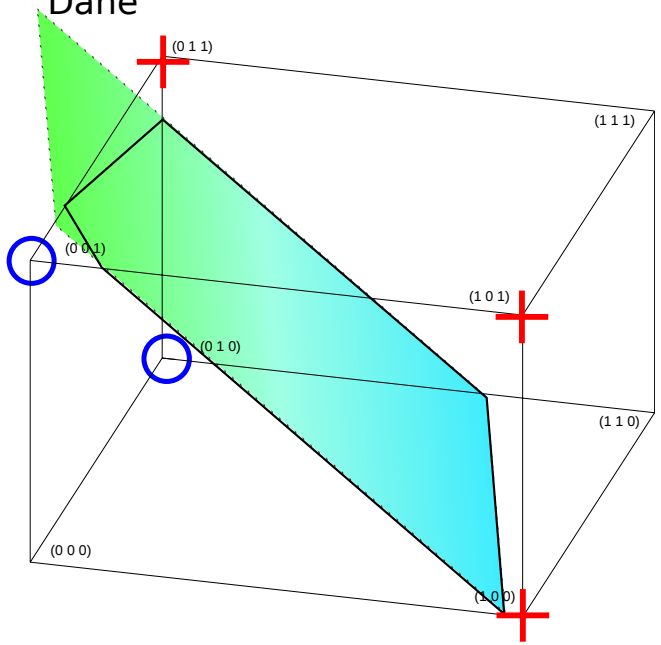


Przykład 8/8

Sieć

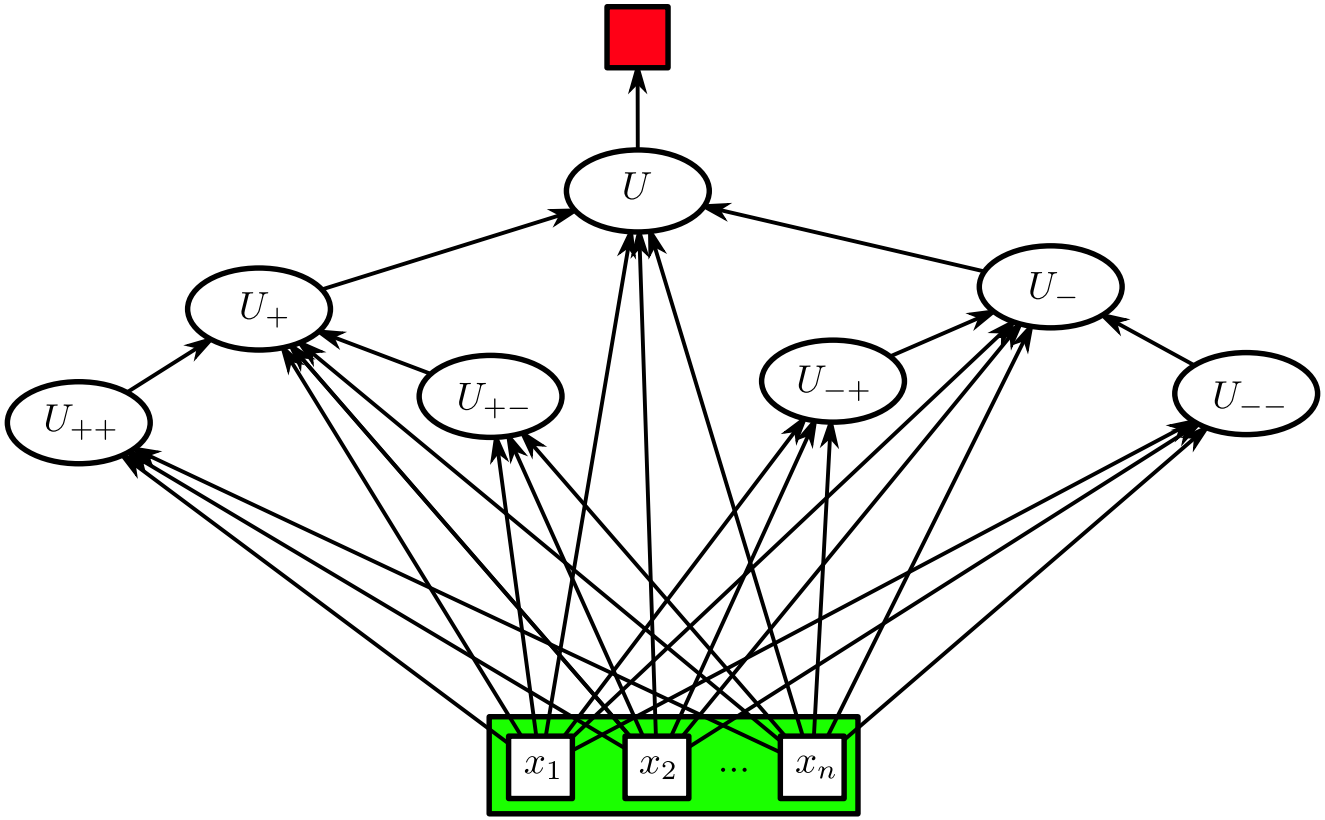


Dane



U_4

Algorytm upstart



Algorytm upstart

- Tworzymy pojedynczą jednostkę U, która widzi wszystkie wejścia. Jej wyjście jest wyjściem całej sieci,
- Odkładamy U na stos wraz ze wszystkimi przykładami uczącymi,
- Dopóki stos jest niepusty, powtarzamy:
  - zdejmujemy ze stosu jednostkę U_i i zbiór stowarzyszonych z nią przykładów uczących,
  - uczymy U_i na jej przykładach algorytmem zapadkowym,
  - jeżeli klasyfikacja U_i jest w pełni zgodna, to rozpocznij następną iterację pętli (continue).

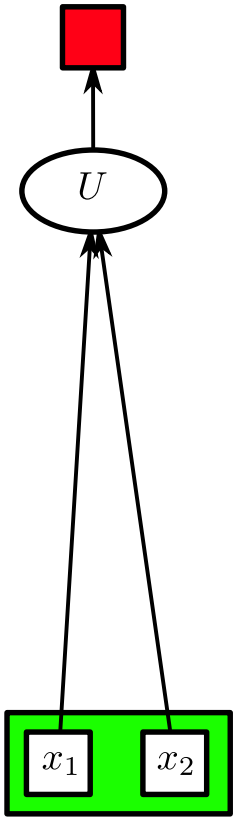
Algorytm upstart cd.

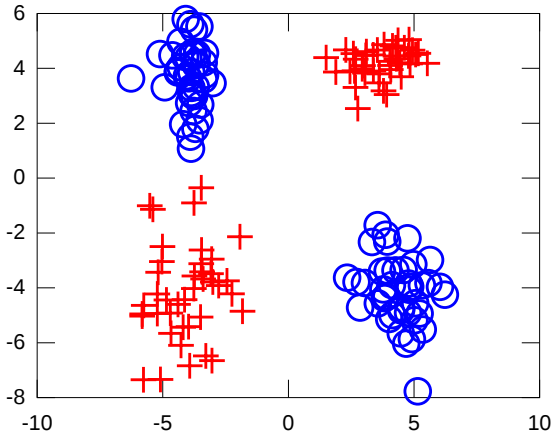
- Dopóki stos jest niepusty, powtarzamy:
  - (...)
  - jeżeli istnieją źle sklasyfikowane przykłady z oczekiwaną odpowiedzią $+1$ dla jednostki U_i, to
    - tworzymy nową jednostkę U_{i+}, jej wejściami są wszystkie wejścia, jej wyjście wchodzi do U_i z dużą wagą dodatnią,
    - odkładamy U_{i+} na stos z następującym zbiorem uczącym: $\left\{E^k : U_i^k = -1, C_{U_i}^k = +1\right\} \cup \left\{E^k : C_{U_i}^k = -1\right\}$, to jest przykłady, które są klasyfikowane przez U_i jako -1, a powinny $+1$ oraz przykłady, która powinny być klasyfikowane przez U_i jako -1, Zbiór uczący dla U_{i+} jest mniejszy od U_i o przykłady dodatnie, które są dobrze klasyfikowane,

Algorytm upstart cd.

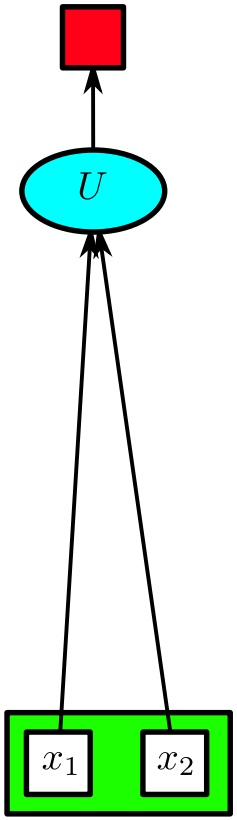
- Dopóki stos jest niepusty, powtarzamy:
  - (...)
  - jeżeli istnieją źle sklasyfikowane przykłady z oczekiwaną odpowiedzią $+1$ dla jednostki U_i, to
    - (...)
  - jeżeli istnieją źle sklasyfikowane przykłady z oczekiwaną odpowiedzią -1 dla U_i, to
    - tworzymy nową jednostkę U_{i-}, jej wejściami są wszystkie wejścia, jej wyjście wchodzi do U_i z dużą wagą dodatnią,
    - odkładamy U_{i-} na stos z następującym zbiorem uczącym: $\left\{E^k : U_i^k = +1, C_{U_i}^k = -1\right\} \cup \left\{E^k : C_{U_i}^k = +1\right\}$, to jest przykłady, które są klasyfikowane przez U_i jako $+1$, a powinny -1 oraz przykłady, która powinny być klasyfikowane przez U_i jako $+1$,
  - zdejmujemy ze stosu następny neuron (continue),
- Zwracamy uzyskaną sieć.

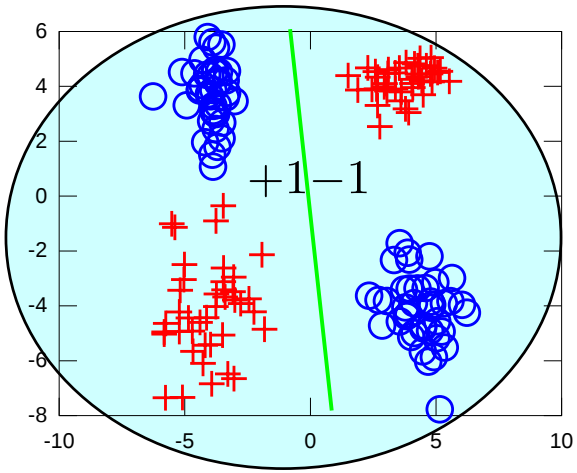
Przykład 1/6



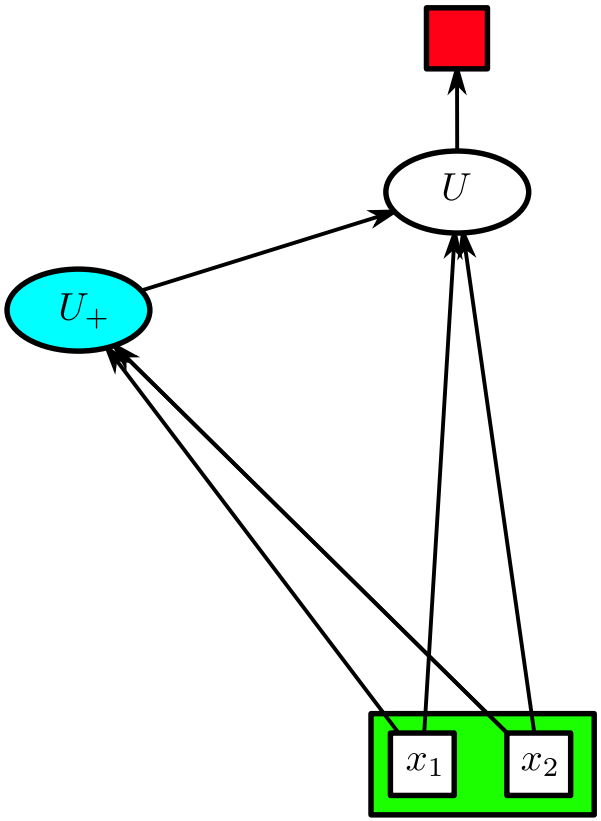


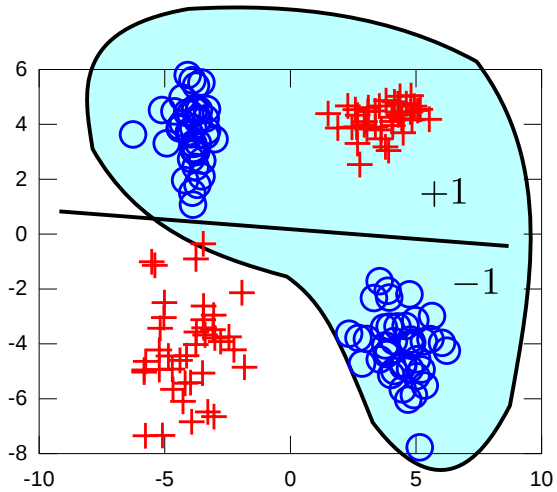
Przykład 2/6



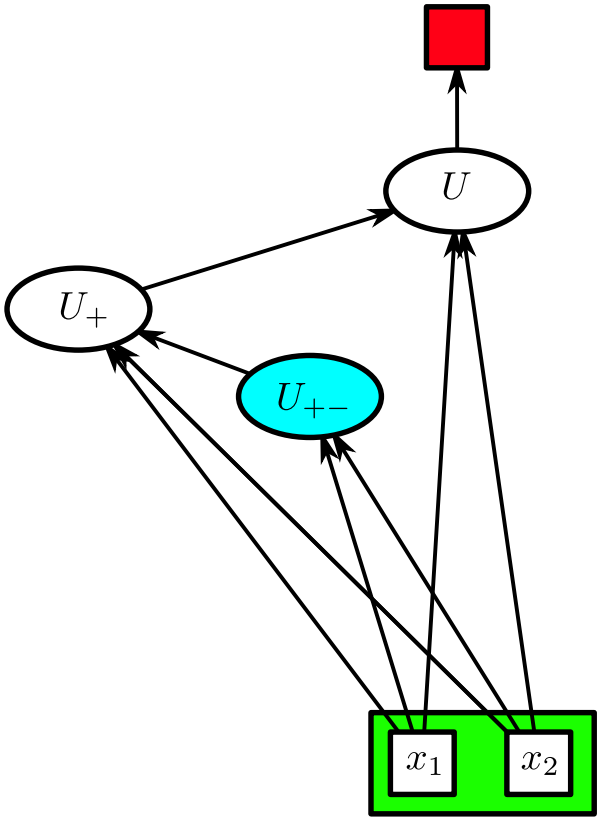


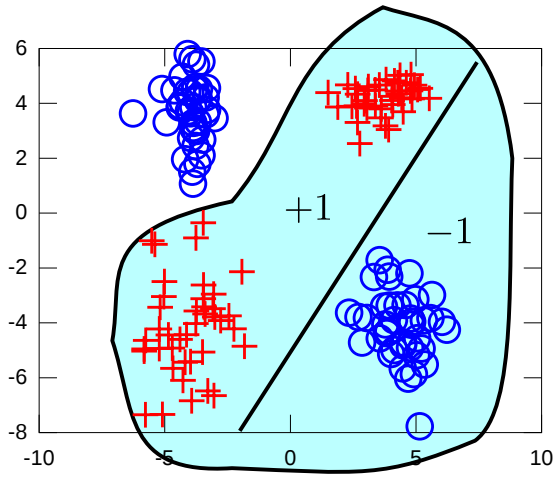
Przykład 3/6



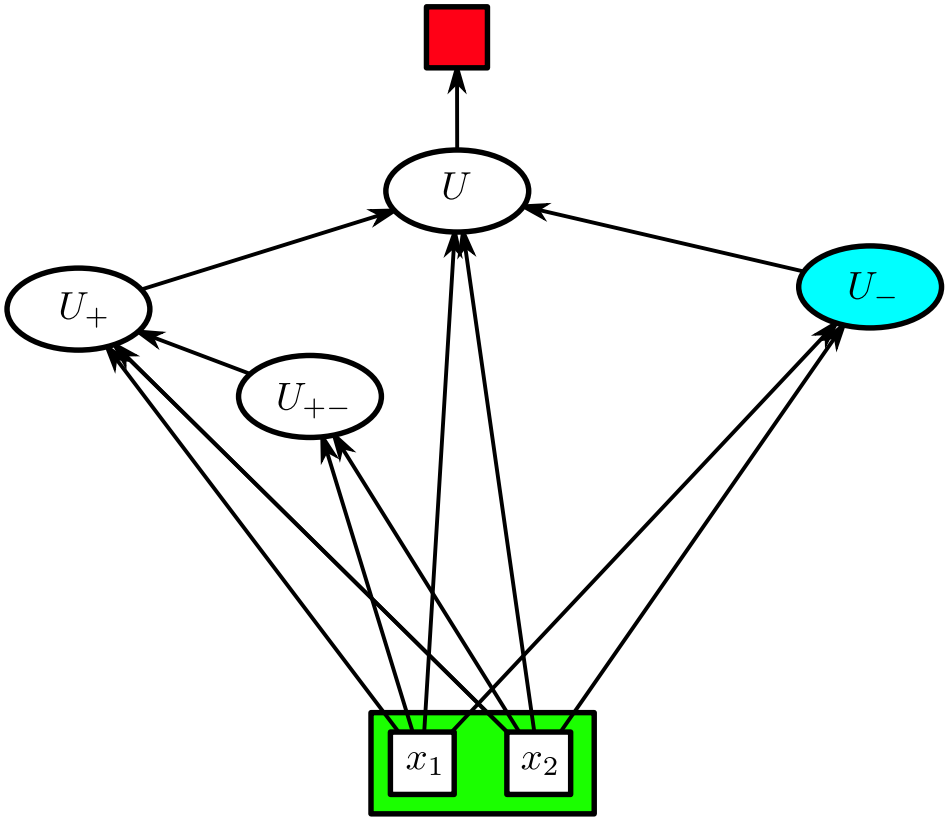


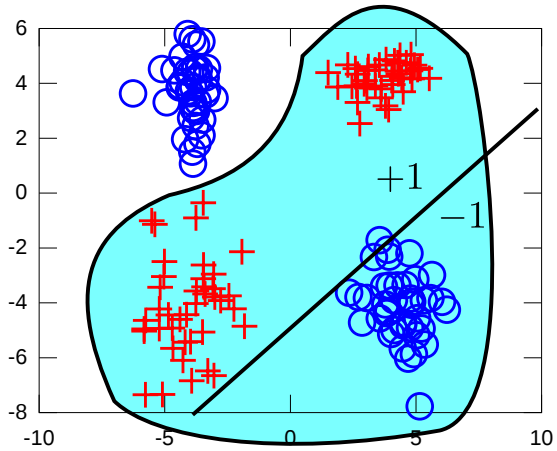
Przykład 4/6



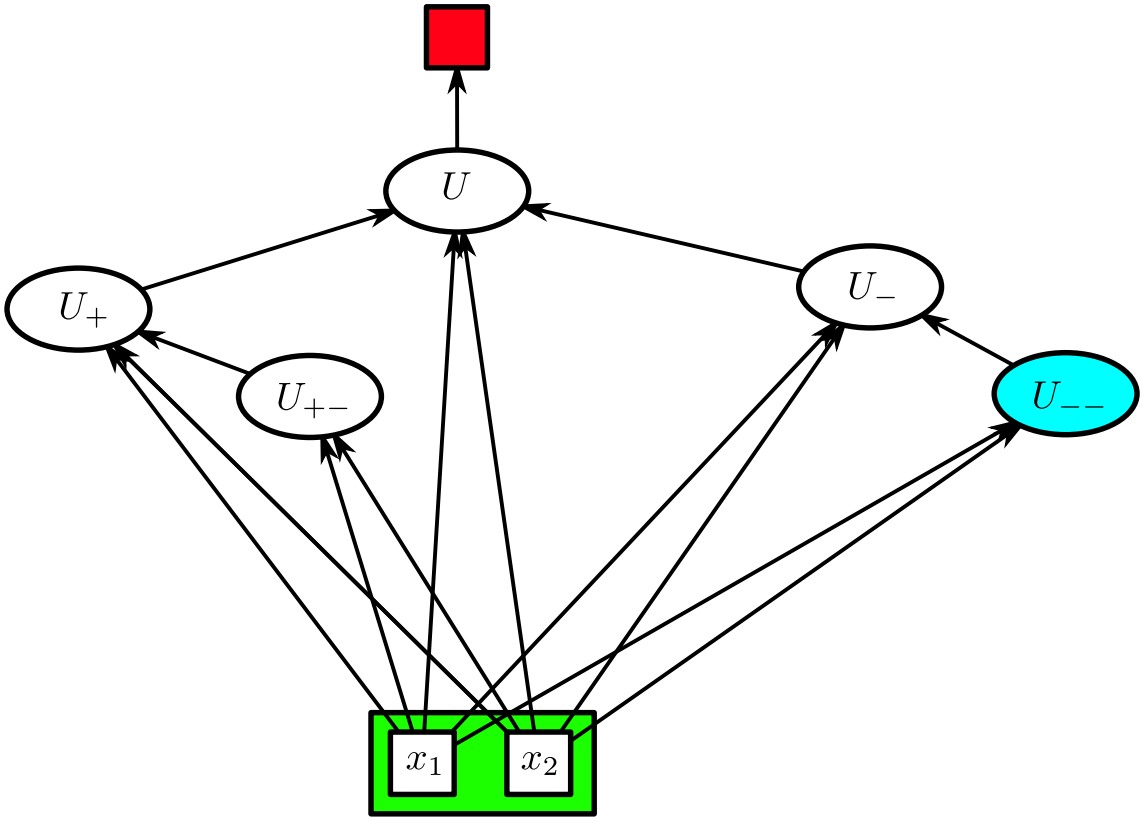


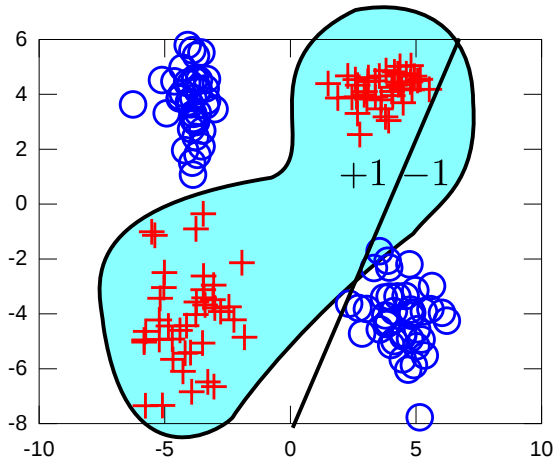
Przykład 5/6





Przykład 6/6





Zadania domowe

- Opisz różnice mięszy siecią wieżową, a piramidalną.
- Która z sieci ma większe możliwości opisu: wieżowa czy piramidalna?
- Skonstruuj sieć warstwową rozwiązującą problem IFF (if and only if).
- Skonstruuj dane jednowymiarowe / dwuwymiarowe, których nie będzie wstanie rozwiązać perceptron, a których nauczy się sieć. Podaj wagi sieci.

Zadania domowe

- Dla sieci wieżowej, piramidalnej lub / i warstwowej wyświetl profil klasyfikacji fragmentu płaszczyzny. Wagi wygeneruj losowo. Przetestuj wykresy dla kilku sieci z różną ilością neuronów / warstw i neuronów w warstwie.
- (*) Skonstruuj dane dwuwymiarowe składające się z trzech przykładów uczących, którego nie będzie wstanie rozwiązać perceptron, a który rozwiąże sieć. Podaj wagi sieci.
- (*) Do tego samego zestawu danych zastosuj różne algorytmy konstrukcyjne. Porównaj uzyskaną jakość uczenia. Porównaj również czas uczenia oraz zużycie pamięci (ilość neuronów i wag).