# Sztuczne Sieci Neuronowe

Wykład 11
Podsumowanie

wykłady przygotowane wg.
W. Duch, J. Korbicz, L. Rutkowski, R. Tadeusiewicz, “Sieci Neuronowe”, Rozdz. 1.
Biocybernetyka i Inżynieria Medyczna, tom VI,  AOFE, Warszawa 2000.

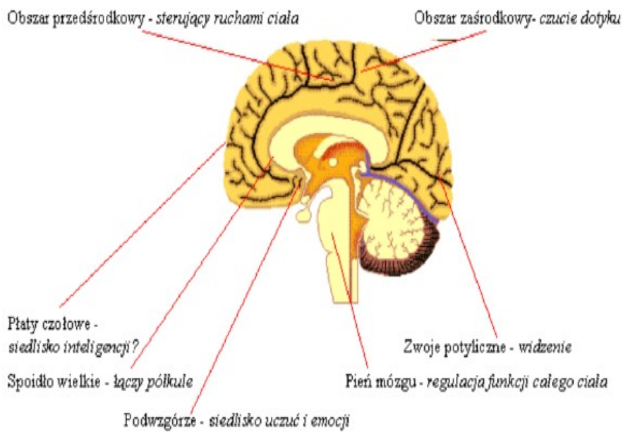
Prof. dr hab. Elżbieta Richter-Wąs

# Liniowe modele

- Przez wiele lat powszechnie stosowaną techniką matematycznego opisywania różnych obiektów i procesów było **modelowanie liniowe**. Dla tego typu modeli dobrze dopracowane/znane są strategie optymalizacji  przy ich budowie.
- Często jednak nie ma podstaw do stosowania **aproksymacji liniowej** dla danego problemu, modele liniowe się nie sprawdzają prowadząc do zbyt szybko wyciąganych wniosków o “niemożności” matematycznego opisu danego systemu.

# Ludzki mózg: niedościgły wzór

Bardzo interesująca jest własność sieci neuronowych, wynikająca z faktu że stanowią one (w jakimś zakresie) probe naśladownictwa działania ludzkiego mózgu.




**Rys. 4.** Widok rozmieszczenia elementów mózgu człowieka w przekroju podłużnym

- Mózg ludzki: objętość 1,4 l., pow. 2000cm2, masa 1,5 kg
- Kora mózgowa: grubość 3 mm, 1010-1011 komórek nerwowych, liczba połączeń (synaps) 1014-1015
- Impulsy komórek nerwowych: częstotliwość 1-100Hz,

czas trwania 1-2 ms, napięcie 100mV

- Szybkość pracy mózgu: 1015 połączeń x 100Hz = 1017 op./s

# Sieci SSN jako nielinowe modele

- **Sieci neuronowe** są bardzo wyrafinowaną techniką modelowania, zdolną do odwzorowywania nadzwyczaj złożonych funkcji. Mają **charakter nieliniowy**, co istotnie wzbogaca możliwość ich zastosowań. To jest jedna z wielu obecnie rozwijanych wyrafinowanych technik!
- Odwołanie się do **modeli tworzonych przy pomocy sieci neuronowych** może być najszybszym i najwygodniejszym rozwiązaniem problemu.
  SSN umożliwiają również kontrole nad złożonym problemem wielowymiarowości, który przy innych podejściach znacząco utrudnia próby modelowania funkcji nieliniowych z dużą ilością zmiennych

# Sieci SSN jako nielinowe modele

- Sieci neuronowe w praktyce same konstruują potrzebne użytkownikowi modele, ponieważ automatycznie uczą się na podanych przez niego przykładach.
  - **użytkownik sieci gromadzi reprezentatywne dane**
  - **uruchamia algorytm uczenia, który ma na celu wytworzenie w pamięci sieci potrzebnej struktury (modelu)**
  - **wyuczona sieć realizuje wszystkie potrzebne funkcje związane z eksploatacją wytworzonego modelu.**

# Sieci SSN jako nieliniowe modele

- Użytkownik potrzebuje pewnej (głównie empirycznej) wiedzy dotyczącej sposobu wyboru i przygotowania danych uczących, musi dokonać właściwego wyboru architektury sieci neuronowej, umieć zinterpretować wyniki... ale poziom wiedzy teoretycznej niezbędnej do skutecznego zbudowania modelu jest przy stosowaniu sieci neuronowych znacznie niższy niż w przypadku stosowania tradycyjnych metod statystycznych.

# Mózg człowieka jako prototyp sieci neuronowej

Można przyjąć że sama dziedzina zaistniała dopiero wraz z wydaniem historycznej pracy

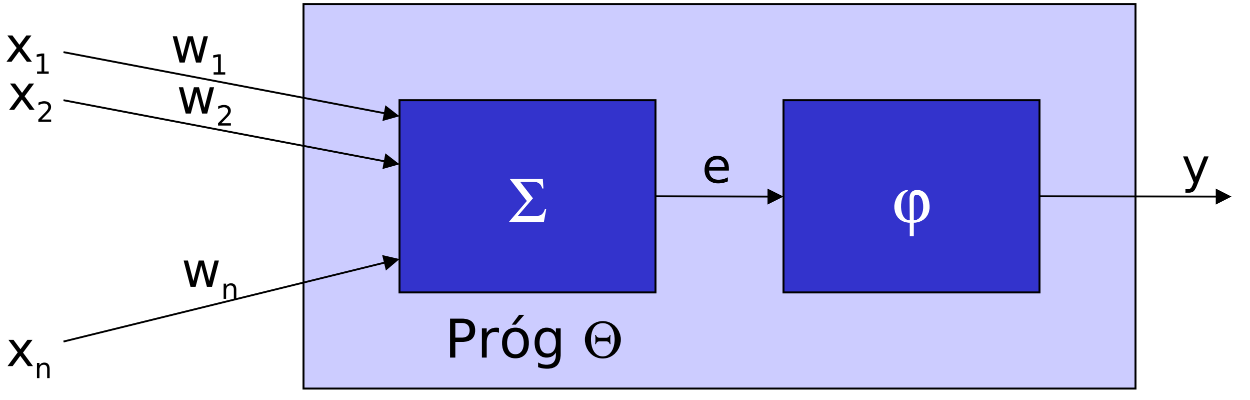
W. S. McCulloch, W. Pitts, *A logical calculus of the ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics, No 5, 1943, pp. 115-133.


**Rys. 2.** Ludzki mózg - pierwowzór i niedościgły ideał dla badaczy sieci neuronowych

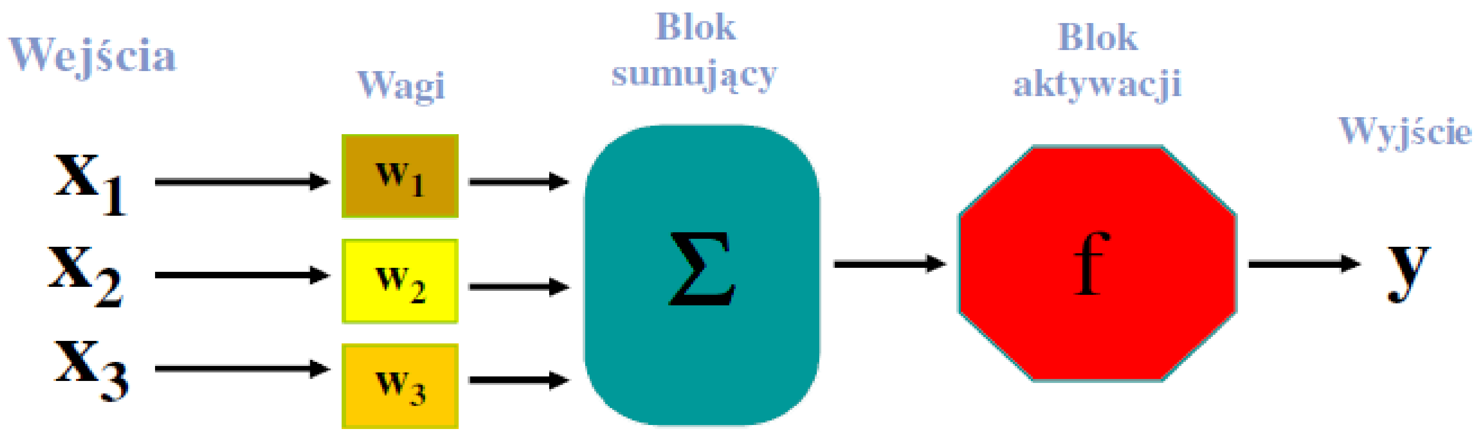
w której po raz pierwszy pokuszono się o **matematyczny opis komórki nerwowej** i powiązanie tego opisu z problemem **przetwarzania danych**.

# Struktura sztucznego neuronu

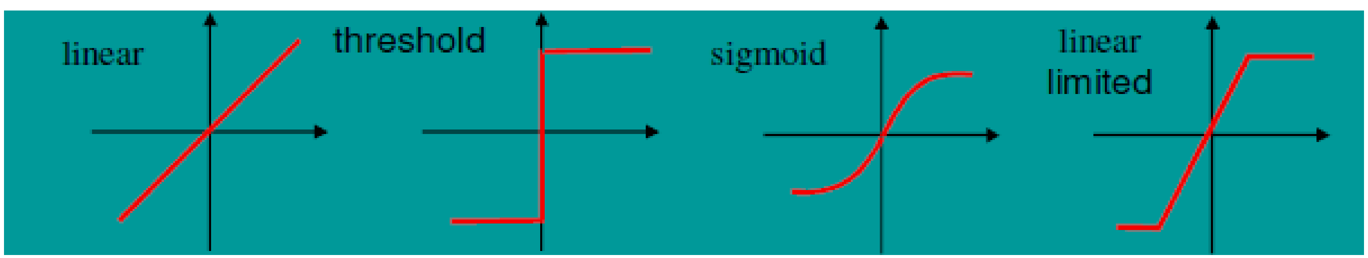
- Jest bardzo interesujące i wręcz intrygujące że sztuczne sieci neuronowe mogą osiągać tak bardzo znaczące rezultaty praktyczne, korzystając z niezwykle uproszczonego modelu neuronu.
- Schematu polega na tym że neuron jedynie wyznacza ważoną sumę swoich wejść i przechodzi w stan pobudzenia wtedy gdy łączny sygnał wejściowy przekroczy pewien ustalony poziom progowy.



# Sztuczny neuron sigmoidalny

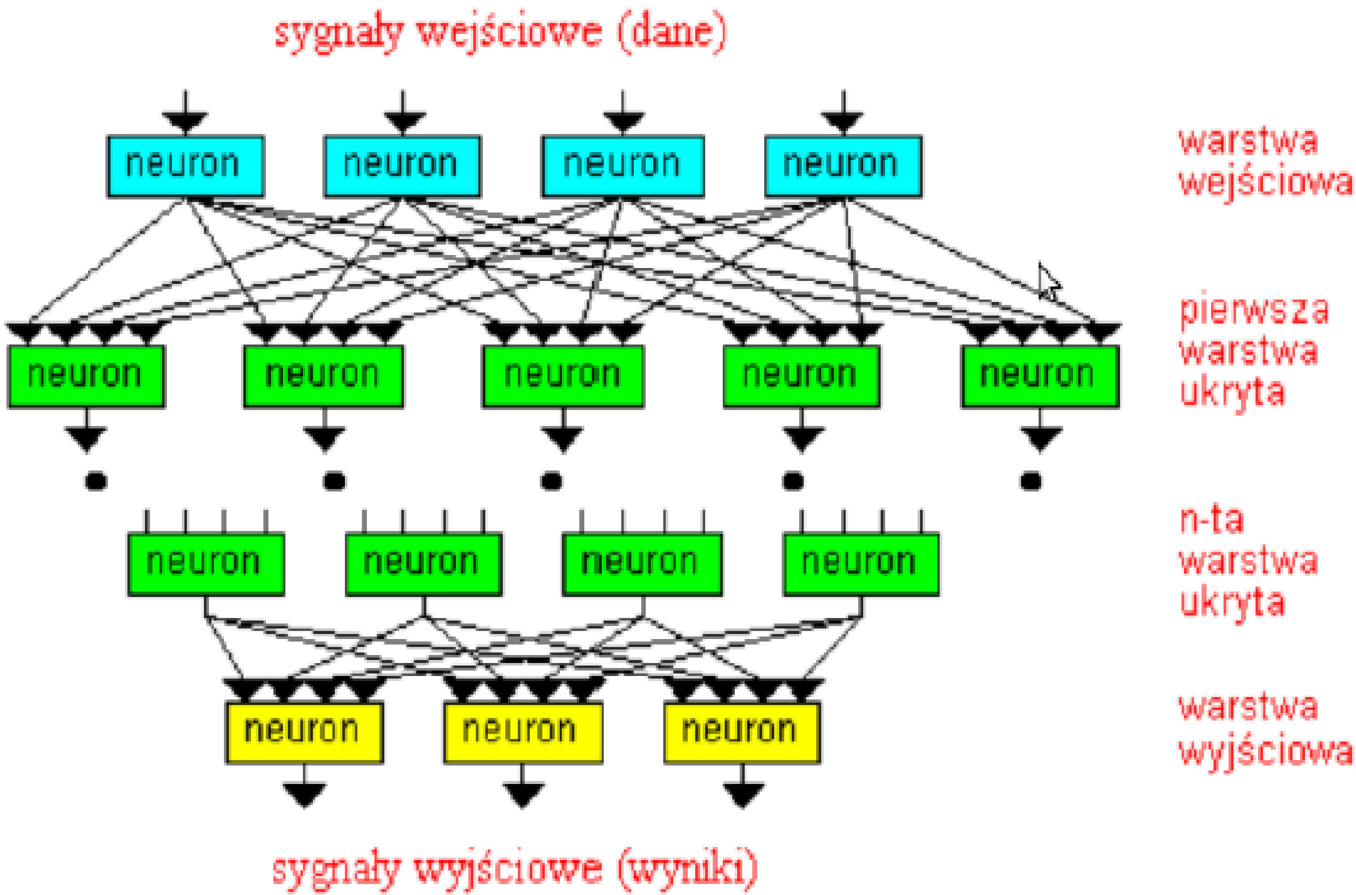



$$y = f\left( \Sigma \left( x_i * w_i \right) \right)$$




**f – funkcja nieliniowa**

# Struktura sztucznej sieci neuronowej



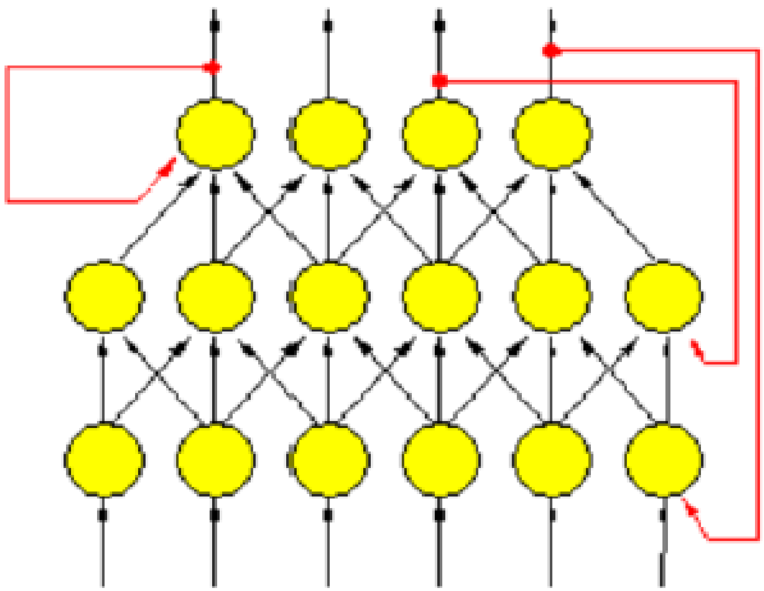

**Rys. 10.** Schematyczna budowa typowej sieci neuronowej

# Struktura sztucznej sieci neuronowej

- Neurony wejściowe, ukryte i wyjściowe musza pozostać wzajemnie połączone co stawia przed twórcą sieci problem wyboru jej struktury.
- Kluczową kwestią przy wyborze struktury sieci jest występowanie lub brak w tej strukturze **sprzężenia zwrotnego**.
- Proste sieci maja **strukturę jednokierunkową** (ang. feedforward): sygnał przepływa w nich tylko w jednym kierunku – od wejść, poprzez kolejne neurony ukryte, osiągając ostatecznie neurony wyjściowe. Strukturę taką charakteryzuje zawsze stabilne zachowanie, co jest jej zaletą.

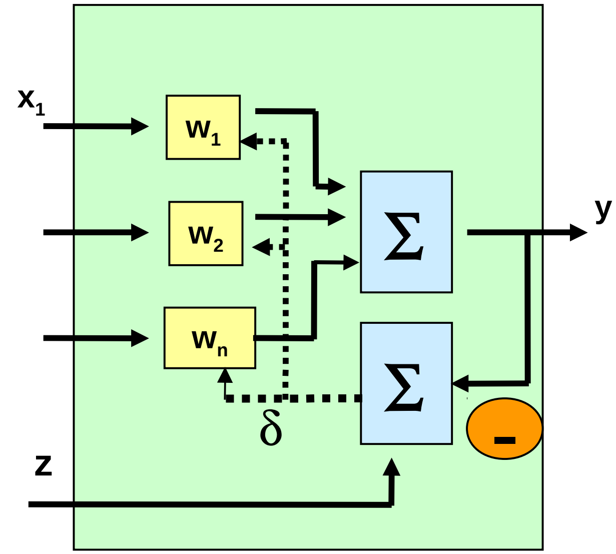
# Struktura sztucznej sieci neuronowej

Sieć może mieć również wbudowane **sprzężenie zwrotne** (tzn. zawiera połączenia powrotne od późniejszych do wcześniejszych neuronów), wówczas może wykonać bardziej skomplikowane obliczenia, w szczególności takie, które maja charakter rekurencyjny



# Uczenie pojedynczego neuronu

Na podstawie ***sygnału błędu*** $\boldsymbol{\delta}$ oraz wektora wejściowego **X** możliwe jest takie skorygowanie wektora wag **W**, by neuron lepiej realizował zadaną funkcję **y = f(X)**. Nowy wektor wag obliczany jest ze wzoru




$$W' = W + \eta\ \delta\ X$$

gdzie $\eta$ jest współczynnikiem liczbowym, decydującym o szybkości uczenia.
Korekta **W** jest tym silniejsza im większy został odnotowany błąd.

# Uczenie “z” lub “bez nauczyciela”

Często stosowaną metodą jest technika uczenia “***bez nauczyciela***”, zwaną ***“unsupervised learning”*** lub ***“hebbian learning”.*** Zasada tego uczenia polega na tym, że waga $\omega_i^{(m)}$, **i-**tego wejścia **m-**tego neuronu wzrasta podczas prezentacji **j-**tego wektora wejściowego $X^{(j)}$ proporcjonalnie do iloczynu **i-**tej składowej sygnału wejściowego tego $x_i^{(j)}$ docierającego do rozważanej synapsy i sygnału wyjściowego rozważanego neuronu.

$$\omega_{\mathbf{i}}^{(\mathbf{m})(\mathbf{j+1})} = \omega_{\mathbf{i}}^{(\mathbf{m})(\mathbf{j})} + \eta\, \mathbf{x}_{\mathbf{i}}^{(\mathbf{j})} \mathbf{y}_{\mathbf{m}}^{(\mathbf{j})}$$

przy czym oczywiście

$$\mathbf{y}_{\mathbf{m}}^{(\mathbf{j})} = \Sigma_{\mathbf{i=1}}^{\mathbf{m}} \omega_{\mathbf{i}}^{(\mathbf{m})(\mathbf{j})} \mathbf{x}_{\mathbf{i}}^{(\mathbf{j})}$$

Wzmocnieniu w sieci ulegają te wagi, które są aktywne (duże $x_i^{(j)}$) w sytuacji gdy “ich” neuron jest pobudzony (duże $y_m^{(j)}$). Tego typu sieć jest zatem ***“autoasocjacyjna”:*** jeśli pewien wzór pobudzeń **X** jest sygnalizowany przez pewne **m-**te wyjście sieci, to w miarę upływu czasu ta sygnalizacja staje się coraz bardziej wyraźna.

# Uczenie z rywalizacją i sieci Kohonena

*Uczenie z rywalizacja* **(competitive learning)**

wprowadził Kohonen przy tworzeniu sieci neuronowych uczących się realizacji ***dowolnych odwzorowań*** **X ⇒ Y**.
Zasada uczenia z rywalizacja jest formalnie identyczna z regułą "instar"

$$\omega_{\mathbf{i}}^{(\mathbf{m}*)(\mathbf{j}+\mathbf{1})} = \omega_{\mathbf{i}}^{(\mathbf{m}*)(\mathbf{j})} + \eta^{(\mathbf{j})}\left(\mathbf{x}_{\mathbf{i}}^{(\mathbf{j})} - \omega_{\mathbf{i}}^{(\mathbf{m}*)(\mathbf{j})}\right)$$

z dwoma dość istotnymi uzupełnieniami.

⇒Wektor wejściowy X jest przed procesem uczenia normalizowany tak, aby ||X|| = 1.

⇒ Index poddawanego treningowi neuronu **m*** nie jest przypadkowy czy arbitralnie wybierany, jest to bowiem ten (i tylko ten) **neuron którego sygnał wyjściowy $y_{m*}^{(j)}$** jest największy. Przy każdorazowym podaniu sygnału wejściowego $X^{(j)}$ neurony rywalizują ze sobą i wygrywa ten, który uzyskał największy sygnał wyjściowy $y_{m*}^{(j)}$. Tylko ten **zwycięski neuron podlega uczeniu**, którego efektem jest jeszcze lepsze dopasowanie wag $W^{(m*)(j+1)}$ do rozpoznawania obiektów podobnych do $X^{(j)}$.

# Uczenie z rywalizacją i sieci Kohonena

Reguła uczenia Kohonena bywa często wzbogacana o dodatkowy element związany z topologią uczącej się sieci. Neurony w sieci są ***uporządkowane***, można więc wprowadzić pojęcie ***sąsiedztwa***. Uogólniona metoda samoorganizującej się sieci Kohonena polega na tym, że uczeniu podlega nie tylko neuron **m*** wygrywający w konkurencji z innymi neuronami sieci, ale także neurony które z nim sąsiadują.
Formalnie regule można zapisać wzorem:

$$\omega_{\mathbf{i}}^{(\mathbf{m}*)(\mathbf{j}+\mathbf{1})} = \omega_{\mathbf{i}}^{(\mathbf{m}*)(\mathbf{j})} + \eta^{(\mathbf{j})}\mathbf{h}(\mathbf{m},\mathbf{m}*)(\mathbf{x}_{\mathbf{i}}^{(\mathbf{j})} - \omega_{\mathbf{i}}^{(\mathbf{m})(\mathbf{j})})$$

formuła uczenia może być zapisana w formie:

$$\omega_i^{(m^*)(j+1)} = \omega_i^{(m^*)(j)} + \eta^{(j)} x_i^{(j)} \left( 2\, y_m^{(j)} - 1 \right)$$

# Uczenie z rywalizacją i sieci Kohonena

Funkcjonowanie powyższego wzoru w istotny sposób oparto na fakcie, że $\mathbf{y_m^{(j)} \in \{0,1\}}$.

Wzór ten nazywamy *reguła Hebb/Anti-Hebb*.

Funkcje **h(m,m*)** można definiować na wiele różnych sposobów, na przykład:

$$h(m,m^*) = \begin{cases} 1 \quad \text{dla} \quad m=m^* \\ 0.5 \;\text{dla}\; |m-m^*| = 1 \\ 0 \quad \text{dla} \quad |m-m^*| > 1 \end{cases}$$

$$h(m,m^*) = 1/\rho(m,m^*)$$

$$h(m,m^*) = \exp\left(-[\rho(m,m^*)]^2\right)$$

# Uczenie z forsowaniem

Omawiane dotychczas techniki **uczenia "bez nauczyciela"** mają bardzo interesującą odmianę polegającą na wykorzystaniu przytoczonych powyżej metod wówczas kiedy wektor wymaganych wartości sygnałów wyjściowych sieci $Z^{(j)}$ jest znany .

Wszystkie wymienione powyżej metody uczenia dadzą się łatwo zastosować poprzez zamianę **y** przez stosowne **z**. Takie uczenie ma charakter ***"forsowania"*** poprawnych rozwiązań bez względu na to co robi sieć.

# Uczenie z forsowaniem

Wyróżnić możemy następujące metody:

- ***metoda autoasocjacji***:

$$\omega_{\mathbf{i}}^{(\mathbf{m})(\mathbf{j+1})} = \omega_{\mathbf{i}}^{(\mathbf{m})(\mathbf{j})} + \eta^{(\mathbf{j})} \mathbf{z}_{\mathbf{m}}^{(\mathbf{j})}$$

- ***metoda przyrostowej autoasocjacji***:

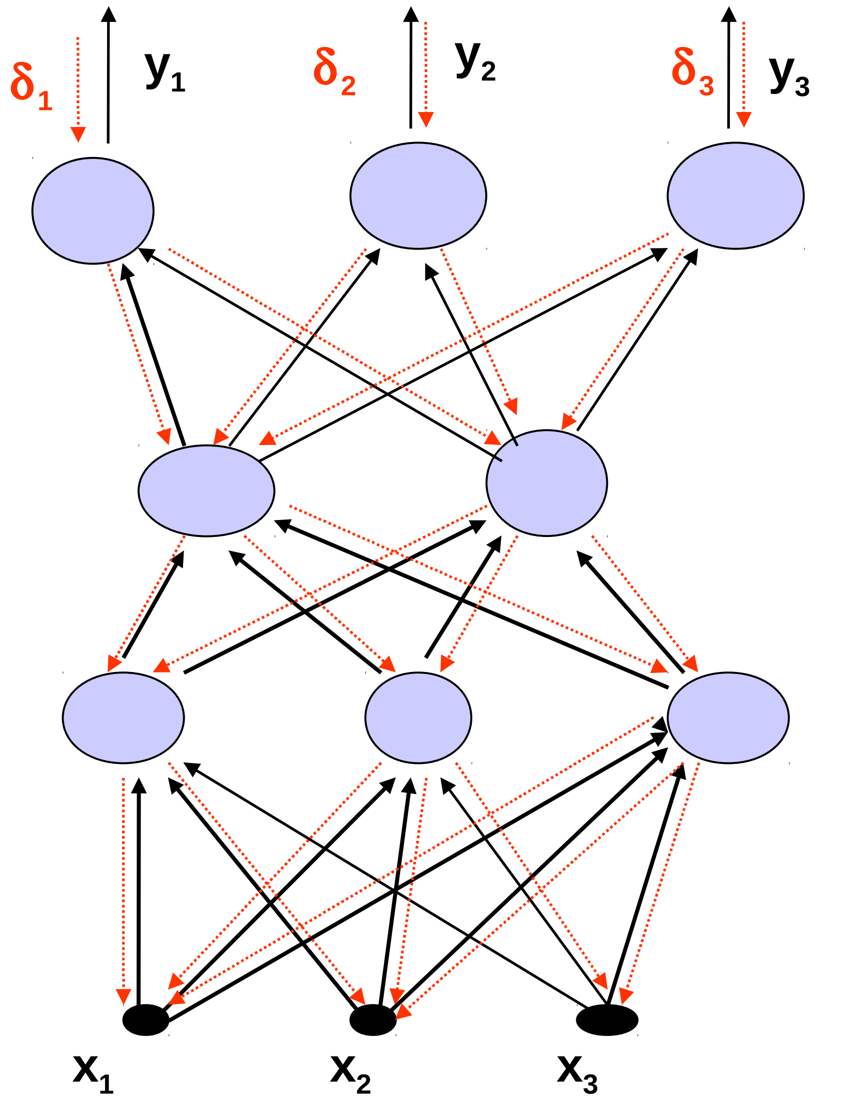
$$\omega_{\mathbf{i}}^{(\mathbf{m})(\mathbf{j+1})} = \omega_{\mathbf{i}}^{(\mathbf{m})(\mathbf{j})} + \eta^{(\mathbf{j})} [(\mathbf{x}_{\mathbf{i}}^{(\mathbf{j})} - \mathbf{x}_{\mathbf{i}}^{(\mathbf{j-1})})(\mathbf{z}_{\mathbf{m}}^{(\mathbf{j})} - \mathbf{z}_{\mathbf{m}}^{(\mathbf{j-1})})]$$

- ***metoda zbliżania wektora wag do wektora odpowiedzi***:

$$\omega_{\mathbf{i}}^{(\mathbf{m})(\mathbf{j+1})} = \omega_{\mathbf{i}}^{(\mathbf{m})(\mathbf{j})} + \eta^{(\mathbf{j})} (\mathbf{z}_{\mathbf{m}}^{(\mathbf{j})} - \omega_{\mathbf{m}}^{(\mathbf{j})})$$

Wybór jednej z różnych możliwości podyktowany musi być ocena ich przydatności w konkretnym zadaniu.  Brak jest tutaj konkretnej teorii, konieczne są eksperymenty i poszukiwania oparte na badaniach empirycznych.

# Uczenie nieliniowej sieci wielowarstwowej




Propagacja sygnału

Propagacja błędu

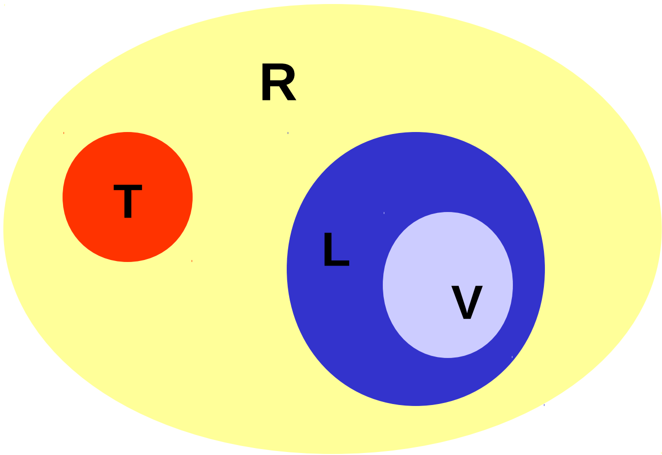
# Zdolności uogólniania sieci neuronowej

Podstawową cechą sieci neuronowej jest jej zdolność do uogólniania, a więc generowania właściwego rozwiązania dla danych, które nie pojawiły się w zestawie danych uczących.

➔ Sieć zostaje poddana uczeniu na zbiorze **L** z bieżącym sprawdzeniem stopnia uczenia na zbiorze **V**.

➔ Zdolność odtworzenia zbioru **L** przez sieć jest miarą **zdolności zapamiętania** danych uczących

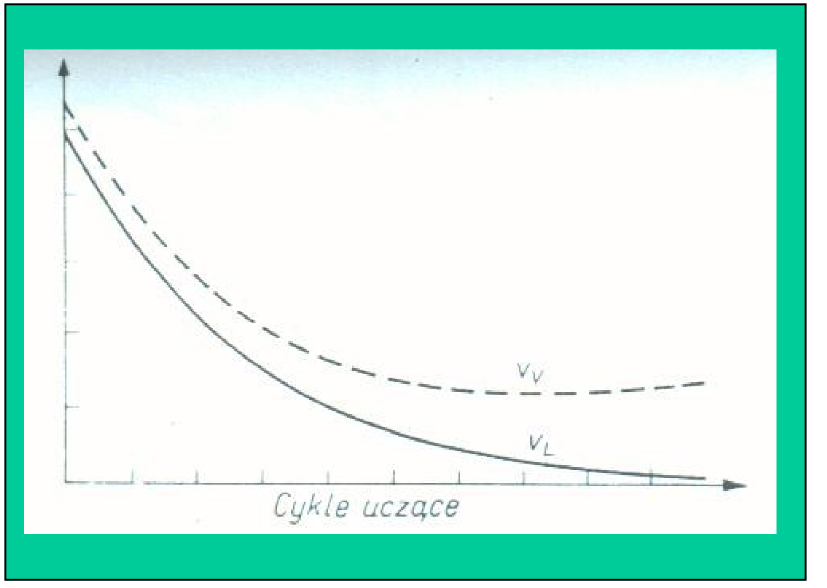
➔ Zdolność do generowania właściwych rozwiązań dla danych należących do zbioru **T**, na których sieć nigdy nie była trenowana, jest miarą **zdolności uogólniania**. (Zakłada się że dane tworzące zarówno zbiór L jak i zbiór T są typowymi reprezentantami zbioru danych)




R – zbiór danych wejściowych
T - zbiór testujący (testing)
L - zbiór uczący (learning)
V - zbiór danych sprawdzających (validation)

# Cykla uczące i błąd veryfikacji

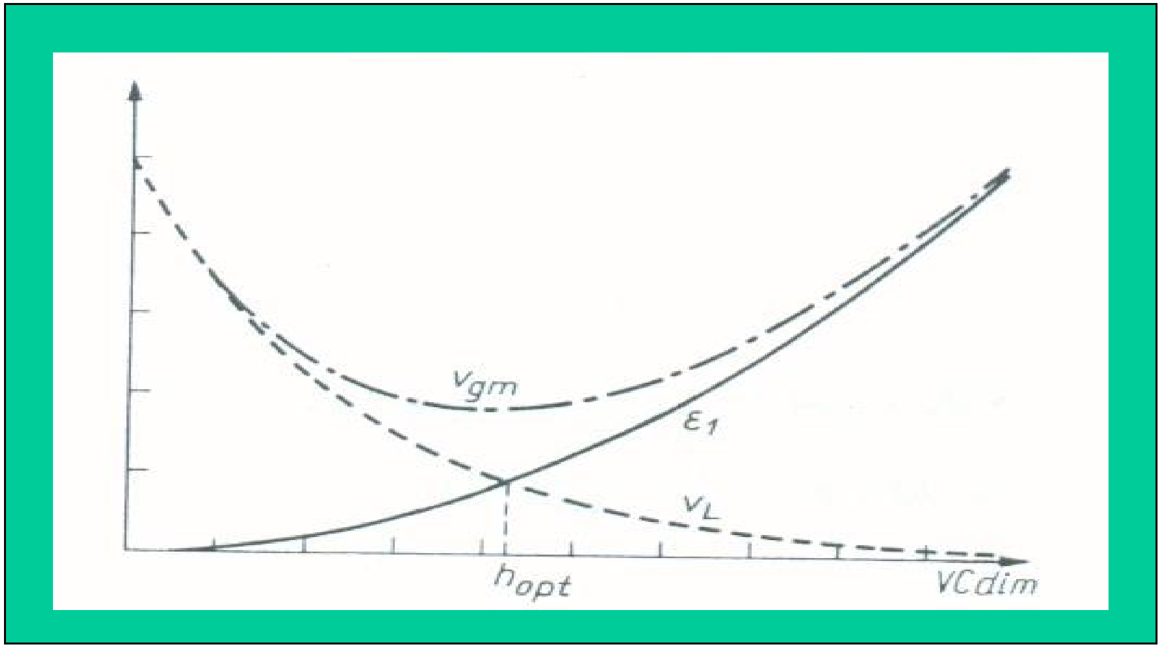
W ogólnym przypadku wraz z upływem czasu uczenia błąd uczenia **$v_L$(W)** maleje i błąd testowania **$v_V$(W)** również (przy ustalonej wartości liczby próbek uczących **p** oraz miary ***VCdim***).




Od pewnego momentu błąd veryfikacji pozostaje stały, natomiast błąd uczenia nadal maleje. W ostatnich fazach procesu uczenia nieregularności w danych odbiegające od cech charakterystycznych danego procesu zaczynają odgrywać role i powodują wzrost błędu testowania.

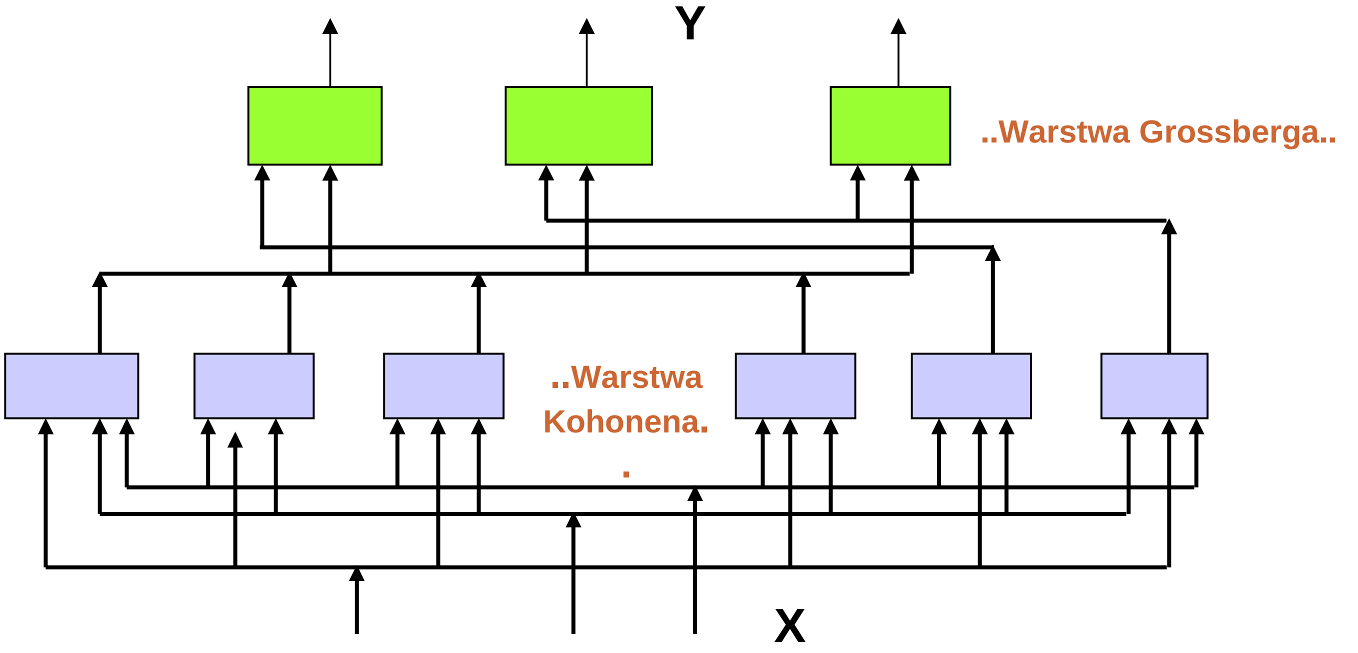
# Zdolności uogólniania sieci neuronowej

Przy stałej liczbie próbek p i wzrastającej wartości miary *VCdim* błąd uczenia $v_L(W)$ maleje monotonicznie, a przedział ufności $\varepsilon_1$ rośnie. W efekcie maksymalny błąd uogólniania osiąga minimum. Zakres *VCdim* < $h_{opt}$ odpowiada nadmiarowości danych bieżących względem aktualnej wartości *VCdim. Zakres VCdim* > $h_{opt}$ odpowiada zbyt malej liczbie danych uczących przy aktualnej wartości *VCdim.*

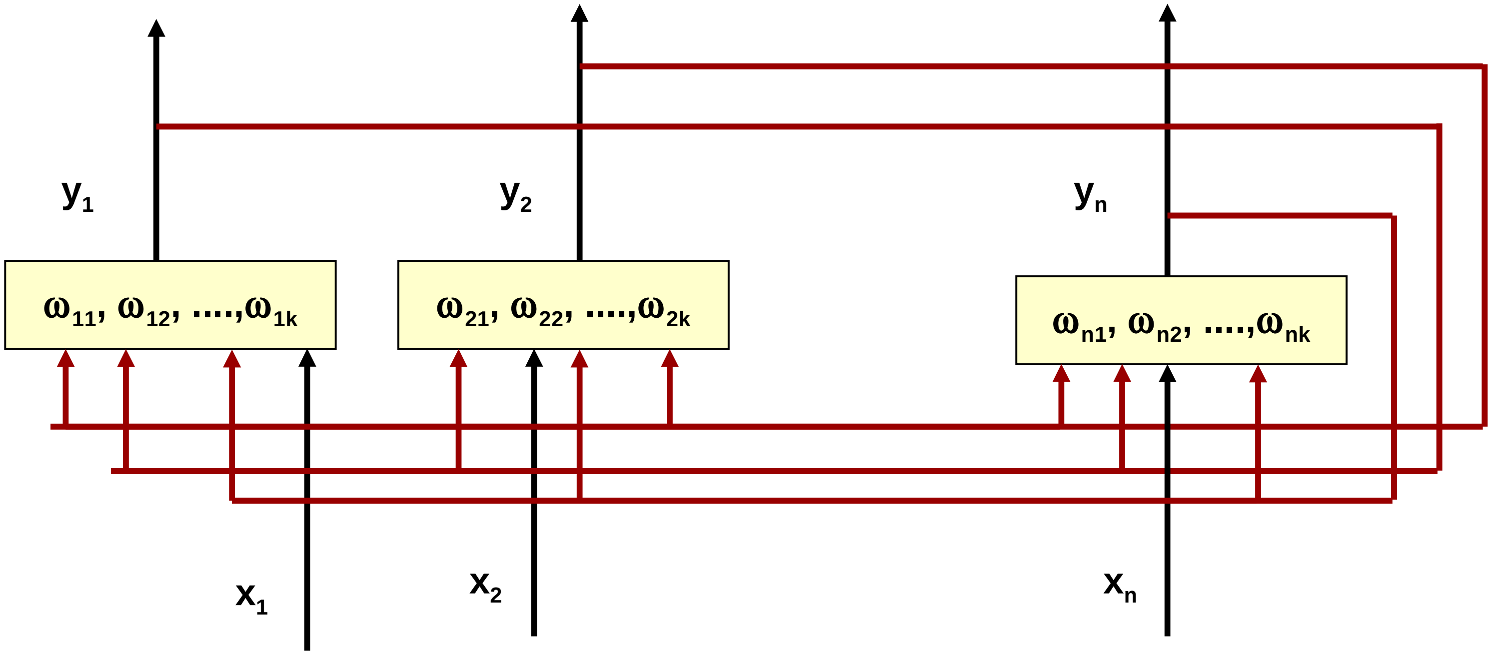


# Sieci CP (Counter Propagation)

Sieć **CP** właściwie nie jest oryginalną propozycją, lecz stanowi **kompilację sieci Kohonena i sieci Grossberga**. Zestawienie tych sieci w strukturze sieci **CP** wprowadziło istotnie nową jakość – sieć stosunkowo szybko się uczącą i mającą (potencjalnie) nieograniczony zakres możliwych odwzorowań pomiędzy sygnałem wejściowym **X** i wyjściowym **Y**.



# Sprzężenie zwrotne jako nowa jakość w strukturach sieci




Sieć o takim schemacie nazywa się ***siecią autoasocjacyjną***. W ramach tego sprzężenia każdy neuron jest połączony jednym z wejść także ze swoim własnym wyjściem, zatem zasada autoasocjacyjnosci odnosi się także do pojedynczych neuronów. **Każdy** neuron sieci ma także kontakt z pewnym, odpowiadającym mu sygnałem wejściowym $x_m^{(j)}$, zatem zaciera się tu podział na warstwę wejściową i pozostałe warstwy sieci.  W związku z tym neurony sieci Hopfielda ***nie tworzą*** wcale ***wyraźnie wydzielonych warstw***, mogą być rozważane w dowolnej topologii.

# Natura procesów w sieci Hopfielda

Uproszczenie to ma następująca interpretację:

→ w chwili początkowej (**j=0**) do neuronów sieci (wszystkich lub wybranych) doprowadza się sygnały wejściowe $\mathbf{x_m^{(0)} \neq 0}$.

→ w wyniku, na wyjściach neuronów sieci wytwarza się zestaw sygnałów wyjściowych $\mathbf{Y^{(1)}}$.

→ sygnały wejściowe zostają odłączone i aż do końca symulacji nie uczestniczą w obliczeniach ($\mathbf{x_m^{(j)} \equiv 0}$)

→ w sieci zaczyna rozwijać się pewien proces, polegający na wyznaczaniu kolejnych wartości $\mathbf{Y^{(j+1)} = \Xi ( Y^{(j)} )}$

# Natura procesów w sieci Hopfielda

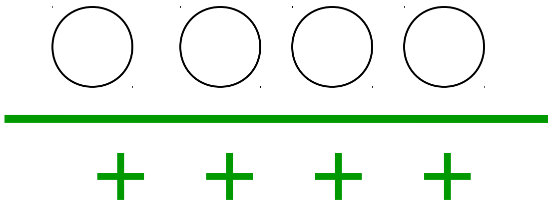
Proces wyznaczamy przez kolejne wartości

**$Y^{(1)}$, $Y^{(2)}$, $Y^{(3)}$, ....... , $Y^{(j-1)}$, $Y^{(j)}$, $Y^{(j+1)}$, .....**

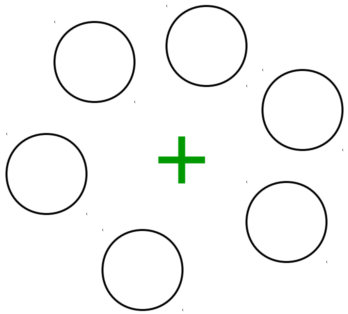
można obserwować w przestrzeni stanu, do której należą wszystkie wektory sygnałów wyjściowych z elementów sieci $Y^{(j)}$.
W tej przestrzeni możliwe są wszystkie znane procesy, jakie związane są z realizacją nieliniowej rekurencyjnej zależności $Y^{(j+1)} = \Xi ( Y^{(j)} )$
→ **stabilizowanie się przebiegów** i ich zbieżność do określonych wartości **$Y^*$**
→ **pojawianie się oscylacji** wartości $Y^{(j)}$ i związanych z nimi ***cykli*** oraz ***orbit*** w przestrzeni $Y$
→ **pojawianie się *przebiegów rozbieżnych***, wreszcie można w takim systemie przewidzieć możliwość pojawienia się ***chaosu***.
O wyborze jednej z tych możliwości decyduje zestaw współczynników wagowych $\omega_i^{(m)}$.

# Bazowe funkcje radialne

## *Ilustracja podziału przestrzeni danych*



siec sigmoidalna



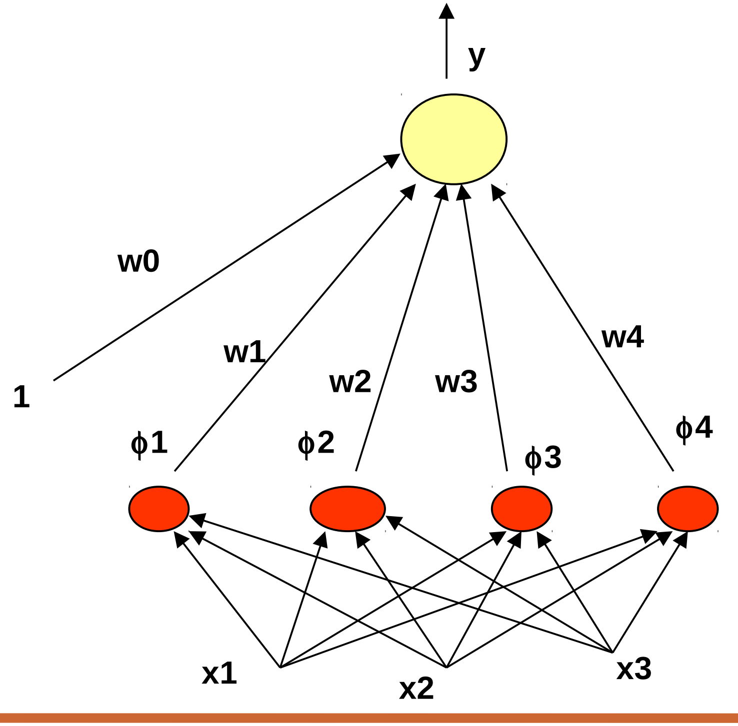
siec radialna

Stanowi to naturalne uzupełnienie neuronu sigmoidalnego, umożliwiające w wypadku wystąpienia naturalnej kołowej symetrii danych wydatne zmniejszenie liczby neuronów potrzebnych do realizacji zadania klasyfikacyjnego.

# Sieć RBF

***Sieć RBF***

Ma ona strukturę dwuwarstwową, warstwa ukryta realizuje odwzorowanie nieliniowe realizowane przez neurony radialnej funkcji bazowej. Neuron wyjściowy jest liniowy, a jego rolą jest sumowanie wagowe sygnałów pochodzących od neuronów warstwy ukrytej.

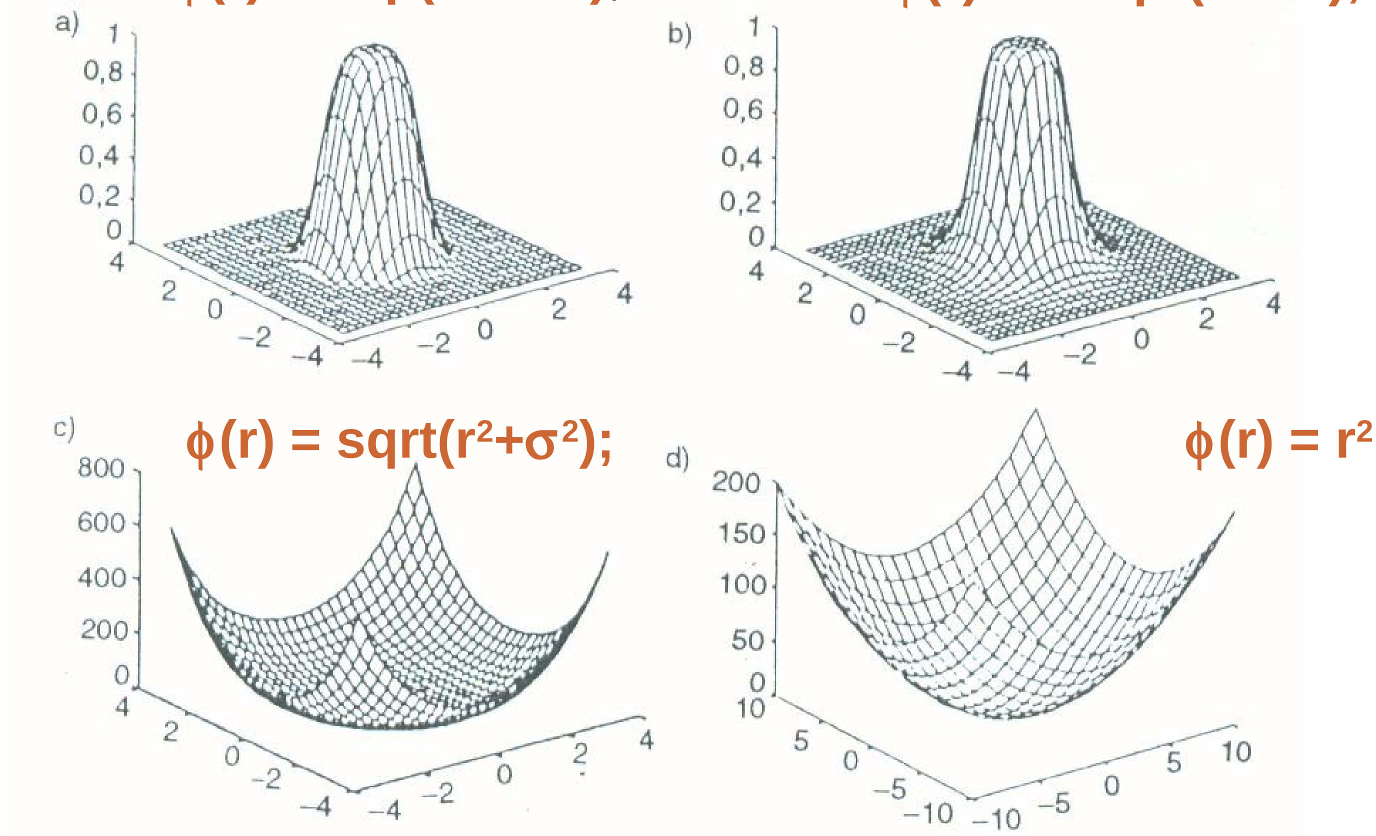



Ogólna postać
sieci radialnej RBF

# Sieć neuronowa radialna

$\phi(r) = \exp(-r^2/2\sigma^2)$, $\phi(r) = 1/\mathrm{sqrt}(r^2+\sigma^2)$,

$\phi(r) = \mathrm{sqrt}(r^2+\sigma^2)$; $\phi(r) = r^2$




Wykresy funkcji bazowych: a) funkcja Gaussa; b) funkcja Hardy'ego; c) funkcja wielomianowa; d) funkcja potęgowa.

# Metody uczenia

Problem uczenia sieci przy wybranym typie radialnej funkcji bazowej składa się z dwu etapów:

$\rightarrow$ doboru centrów i parametrów kształtu funkcji bazowych
$\rightarrow$ doboru wag neuronów warstwy wyjściowej

Podstawowa trudność to etap pierwszy.

Stosuje się najczęściej:

- $\rightarrow$ **wybór losowy**,
- $\rightarrow$ **samoorganizujący się proces podziału na klastry**
- $\rightarrow$ **uczenie pod nadzorem**.

# Sieć radialna a sieć sigmoidalna

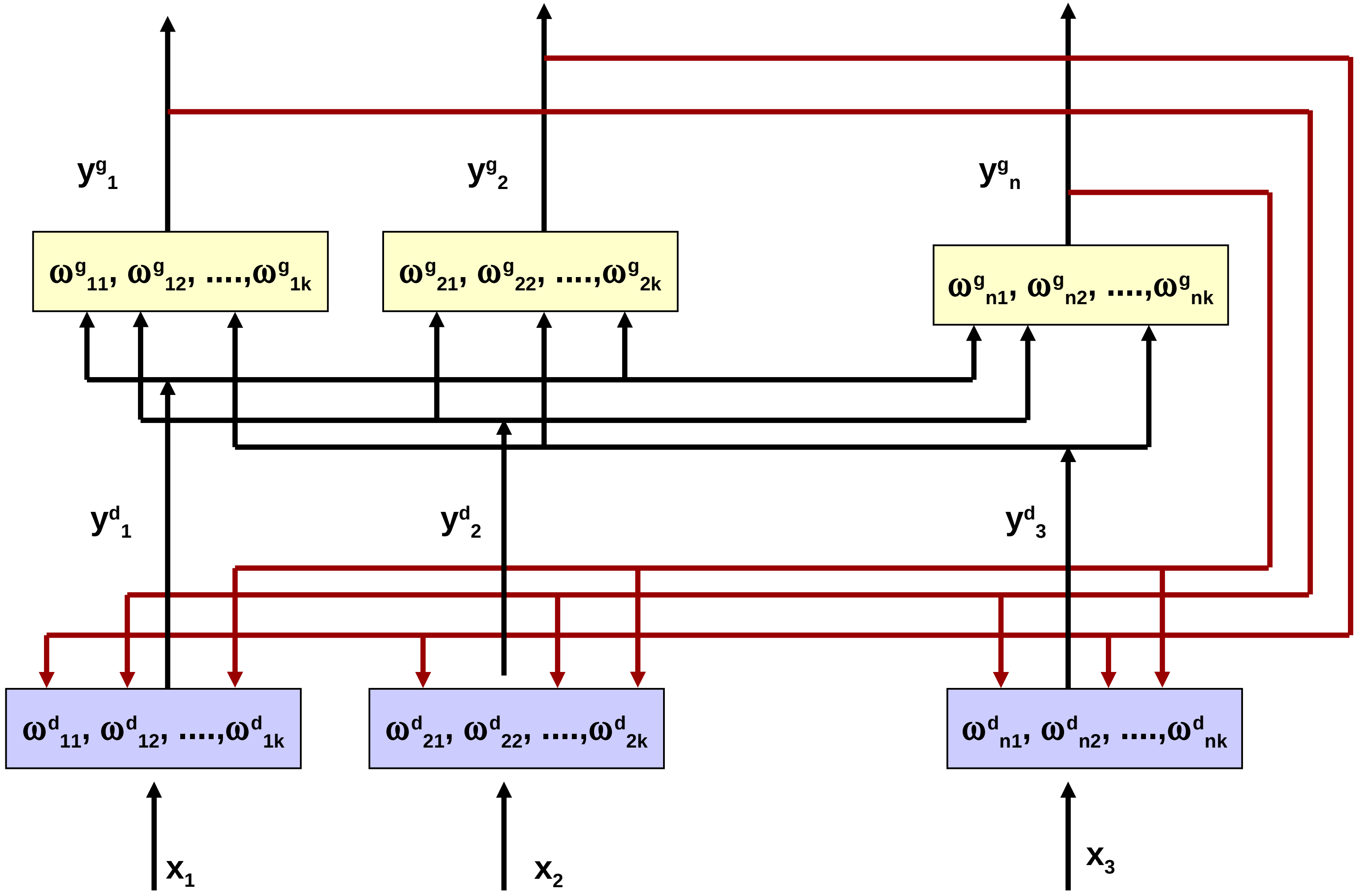
***Sieć sigmoidalna:***

Działanie funkcji rozciąga się od określonego punktu w przestrzeni aż do nieskończoności, reprezentuje aproksymację globalną funkcji zadanej. Nie ma niemożności fizycznego powiązania obszaru aktywności neuronu z odpowiednim obszarem danych uczących, trudności z określeniem optymalnego punktu startowego z procesie uczenia.

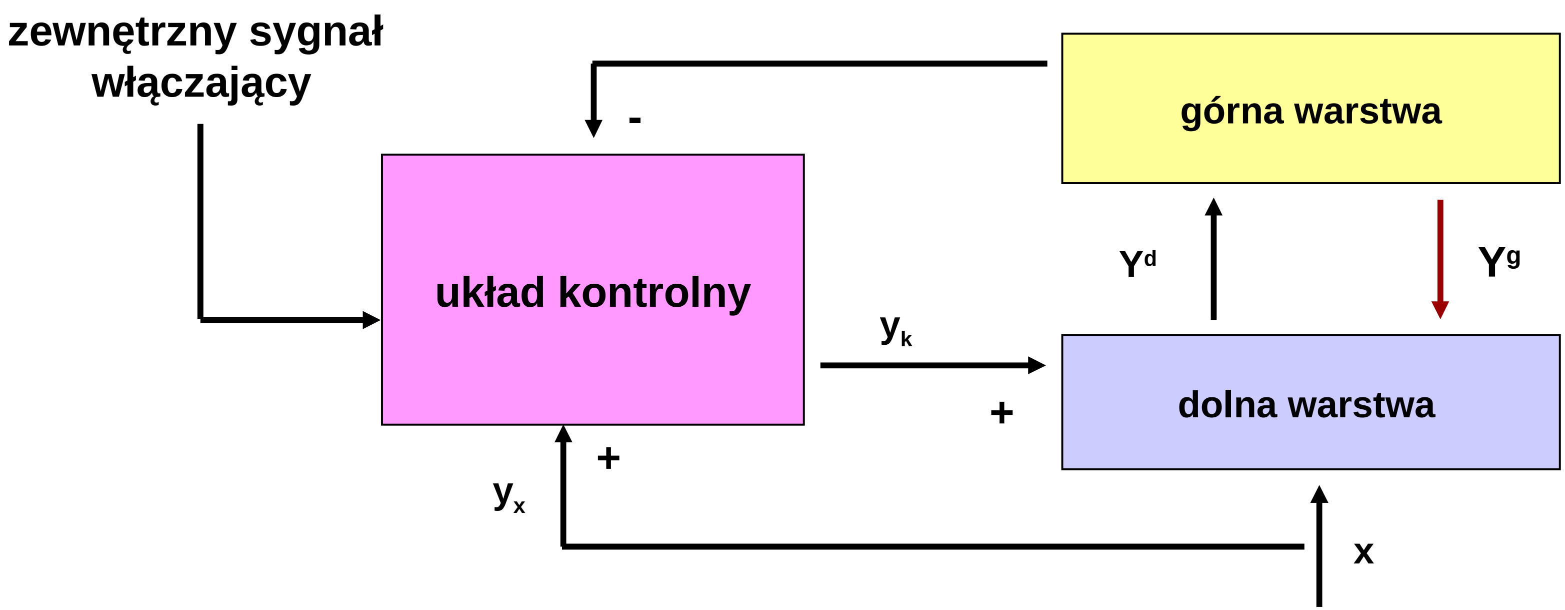
***Sieć radialna:***

Bazuje na funkcjach mających wartość niezerową jedynie w określonej przestrzeni tylko wokół centrów, realizuje aproksymację typu lokalnego, której zasięg działania jest bardzo ograniczony. Można się spodziewać że zdolności do uogólniania są gorsze niż dla sieci sigmoidalnych. Łatwość powiązania parametrów funkcji bazowych z fizycznym rozmieszczeniem danych w obszarze parametrów. Łatwość uzyskania dobrych wartości startowych w procesie uczenia pod nadzorem.

# Schemat sieci ART



# Układ kontrolny sieci ART

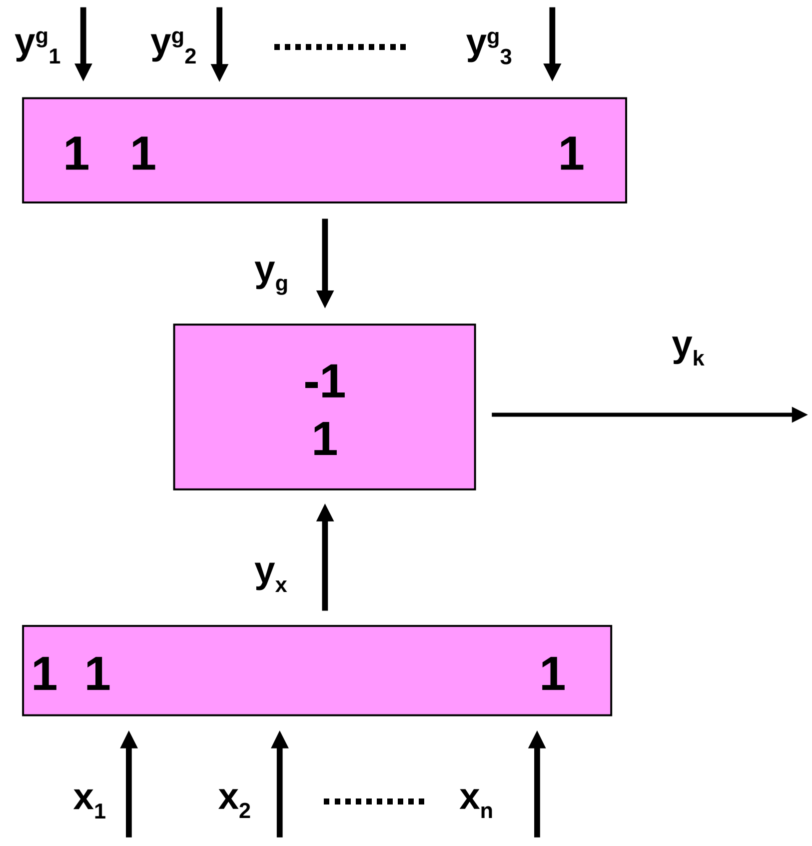



Działanie układu kontrolnego polega na tym, że jego sygnał dodatkowo pobudza (albo zwiększa czułość) neuronów dolnej warstwy. Bez tego dodatkowego sygnału neurony te nie są w stanie reagować na sygnały górnej warstwy $Y^g$ i pętla "halucynacji" ulega przerwaniu.

# Układ kontrolny sieci ART

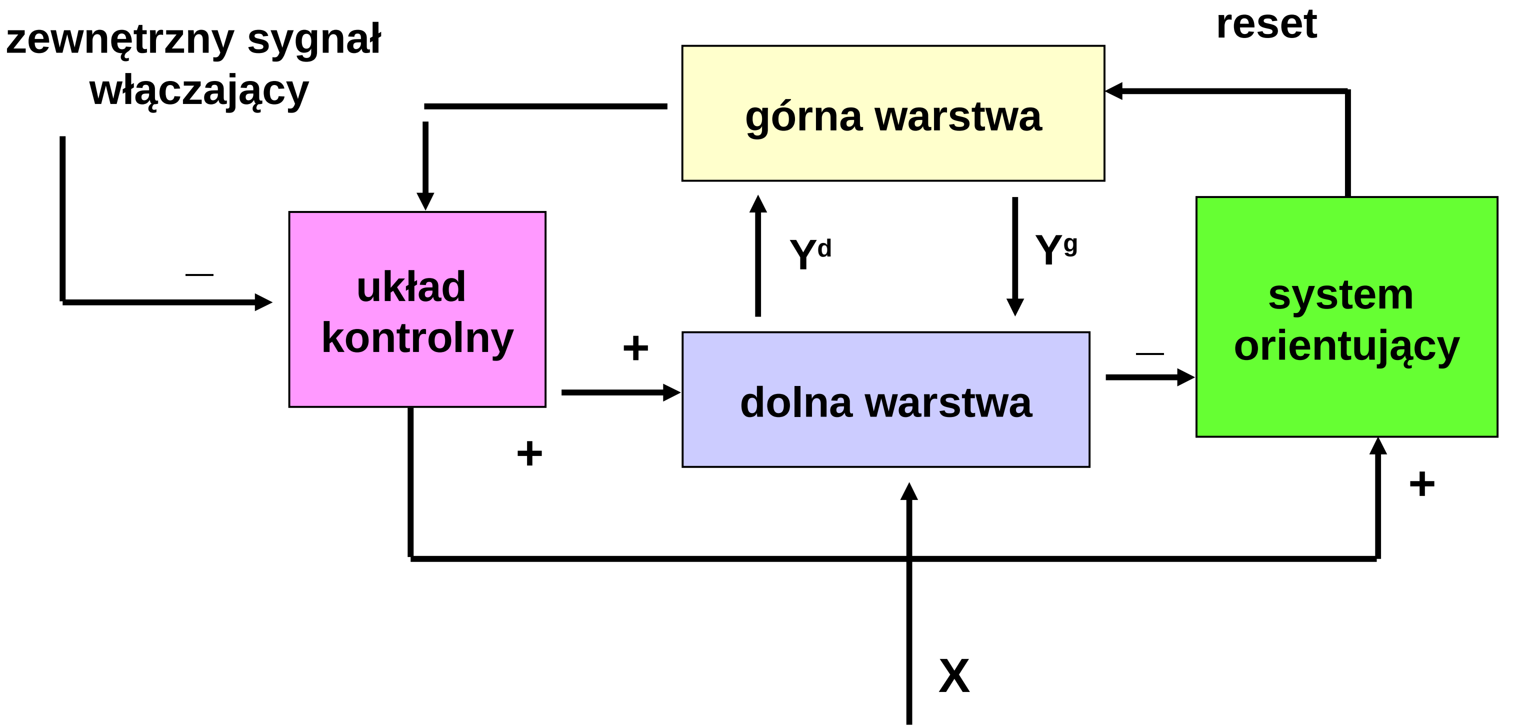
Strukturę układu kontrolnego można przedstawić następująco:

Sygnał $y_k$ doprowadzony jest do wszystkich elementów dolnej (wejściowej) warstwy sieci i współuczestniczy w kształtowaniu ich sygnałów.



# Układ orientujący

Innym oryginalnym elementem sieci ART jest system ***orientujący*** (orienting system), którego celem jest ***sterowanie precyzją odwzorowania*** poszczególnych kategorii w sieci ART.

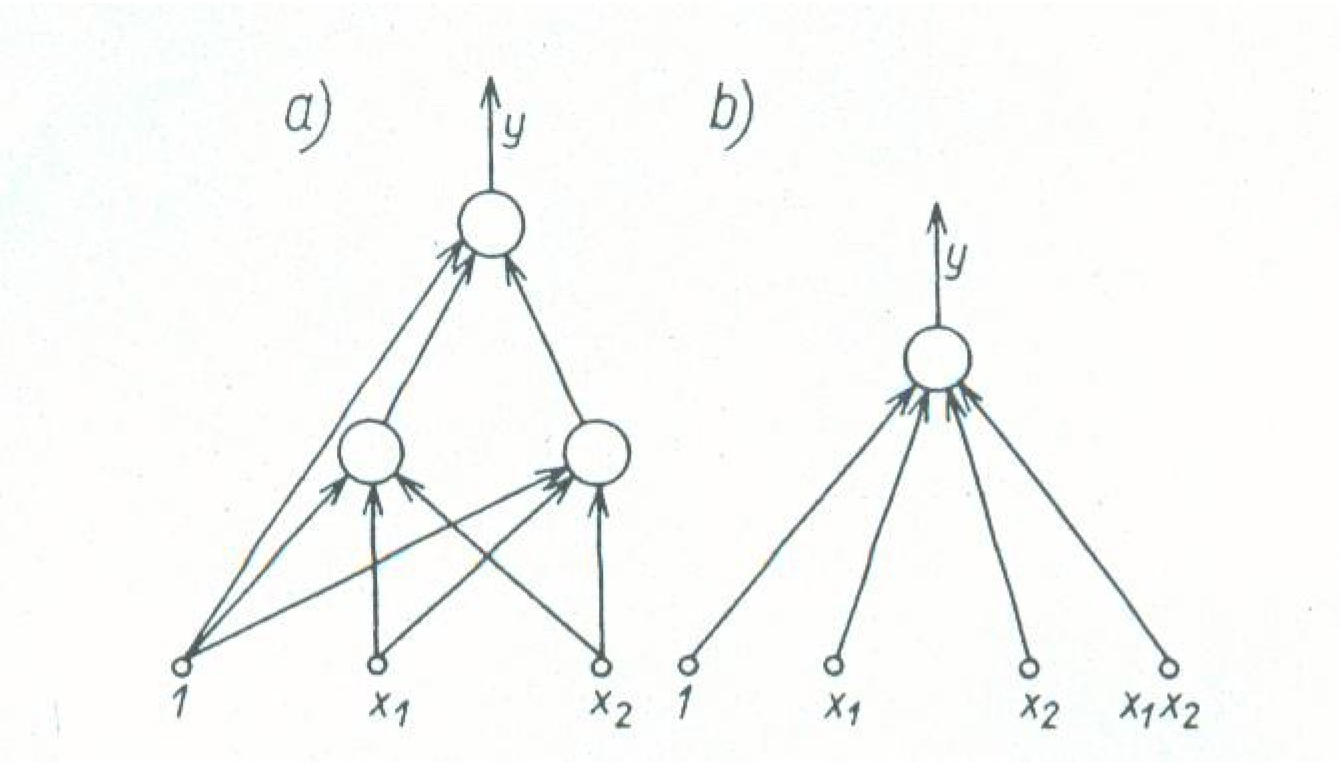


# Sieć Pao

***Sieci bazujące na funkcjach pojedynczych sygnałów $x_i$***

Reprezentacja rozszerzona składa się ze zbioru oryginalnego $\mathbf{x_i}$ oraz zbioru funkcji pojedynczych elementów $\mathbf{x_i}$. Przykładami funkcji rozszerzających może być funkcja wielomianowa, funkcje ortogonalne: **$\sin\pi x$, $\cos\pi x$, $\sin 2\pi x$, $\cos 2\pi x$**, itp. Efektem takiego rozszerzenia jest rzutowanie wzorców wejściowych z przestrzeni N-wymiarowej w przestrzeń o większych wymiarach. Nie wprowadza to nowej informacji ale wzbogaca istniejącą.

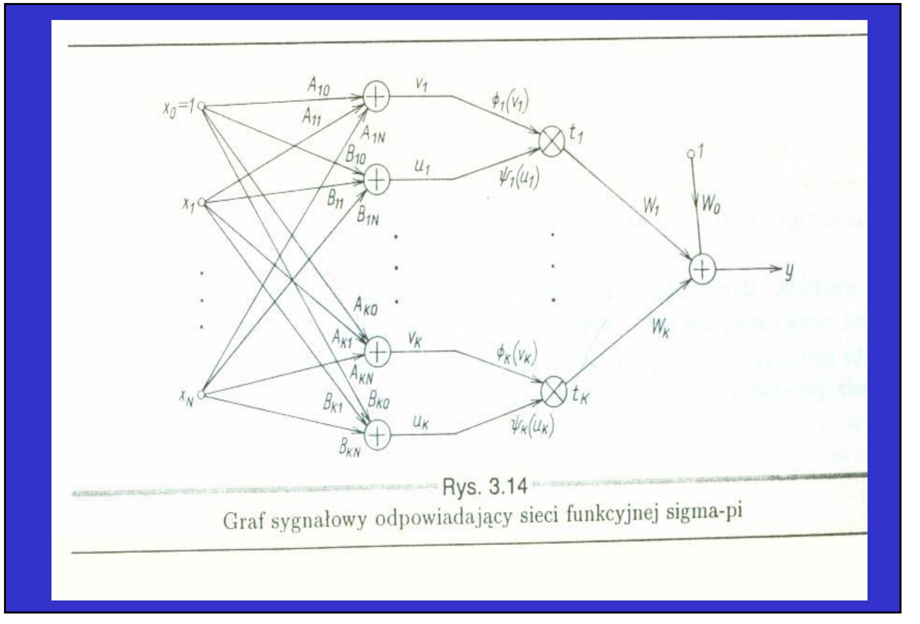
# Sieć Pao



# Sieć sigma-pi

Odmianą sieci neuronowej z rozszerzeniem funkcyjnym jest sieć sigma-pi, wykorzystującą składniki funkcyjne w odwzorowaniu danych wejściowych.

Sieć zawiera 2 warstwy neuronów:
warstwę ukrytą funkcyjną oraz warstwę wyjściową liniową.
Każdy neuron warstwy ukrytej zawiera dwa sumatory, dwie funkcje aktywacji oraz jeden mnożnik.



Rys. 3.14
Graf sygnałowy odpowiadający sieci funkcyjnej sigma-pi

Wprowadzenie połączeń synaptycznych wyższego rzędu do struktury sieci umożliwia uproszczenie i przyspieszenie procesu uczenia z jednoczesnym zwiększeniem zdolności klasyfikacyjnych i aproksymacyjnych sieci.

# Wtrącanie szumu do danych uczących

Metody przestawione poprzednio realizowały zwiększenie zdolności uogólniania poprzez oddziaływanie na samą architekturę sieci. Jest to podstawowa metoda umożliwiająca uzyskanie dobrych właściwości uogólniających sieci.

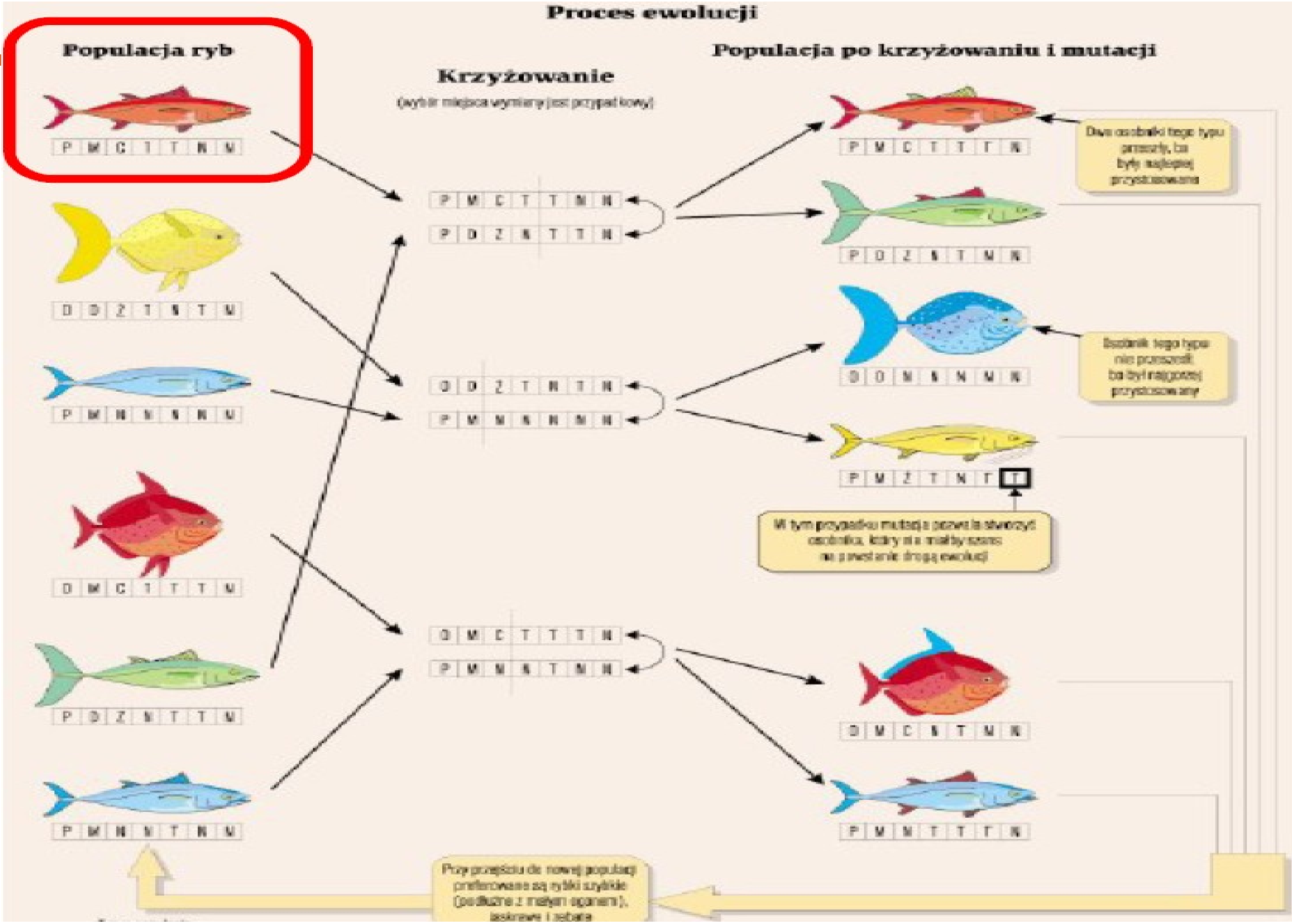
$\rightarrow$ Przy ustalonej minimalnej architekturze sieci jest możliwa dalsza poprawa poprzez odpowiednie przygotowanie zbioru danych uczących. ***Przy dobrze wytrenowanej sieci*** podstawowa jest zasada mówiąca że sygnały wyjściowe sieci powinny być niewrażliwe na zmianę wielkości wejściowych dopóty, dopóki zmiany te są zawarte w pewnych dopuszczalnych granicach przy założeniu, że sieć realizuje odwzorowanie gładkie. Podobne sygnały powinny generować podobne odpowiedzi nawet jeżeli nie wchodziły w skład wzorców uczących.

# Algorytmy genetyczne

Podstawowy algorytm genetyczny, określany też jako klasyczny, powstał na przełomie lat 60-tych/70-tych, dzięki pracom ***Hollanda***.

Holland zainteresowany cechami naturalnej ewolucji, m.in. faktem że odbywa się ona na chromosomach a nie na żywych istotach, wierzył że odpowiedni program komputerowy może zrealizować algorytm rozwiązywania wielu problemów w sposób naśladujący ewolucję. Rozpoczął więc pracę nad algorytmami symulowanej ewolucji przebiegającej na chromosomach, stanowiących ciąg cyfr binarnych.

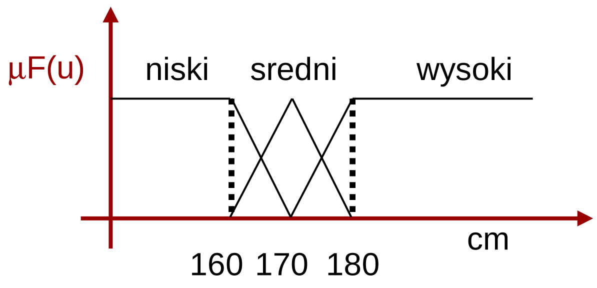
Algorytmy te, wykorzystując analogiczne jak w naturalnej ewolucji **mechanizmy selekcji oraz reprodukcji**, zmierzały do rozwiązania danego problemu poprzez poszukiwanie najlepszego chromosomu na podstawie oceny jego przystosowania.


Proces ewolucji
Populacja ryb
Krzyżowanie
Populacja po krzyżowaniu i mutacji

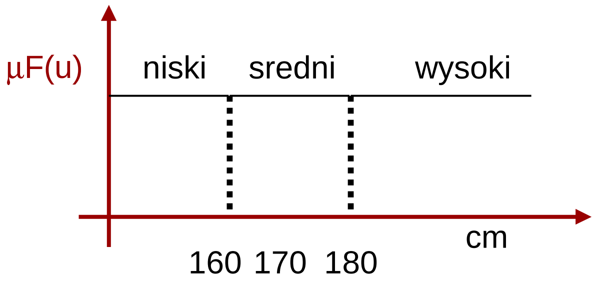
# Podstawowe pojęcia systemów rozmytych

Warunek $\mu F(u) = 1$ oznacza pełną przynależność u do zbioru F, wartość $\mu F(u)=0$ oznacza brak tej przynależności. Wartości pośrednie $\mu F(u)$ wyrażają przynależność częściową u do zbioru F.

W odróżnieniu od ***algebry Boole'a***, która jednoznacznie klasyfikuje dane wejściowe przyporządkowując im w sposób nierozmyty odpowiedni zbiór, w przypadku ***algebry rozmytej*** tej samej wartości zmiennej wejściowej można przyporządkować różne zbiory z określoną wartością funkcji przynależności do zbioru.

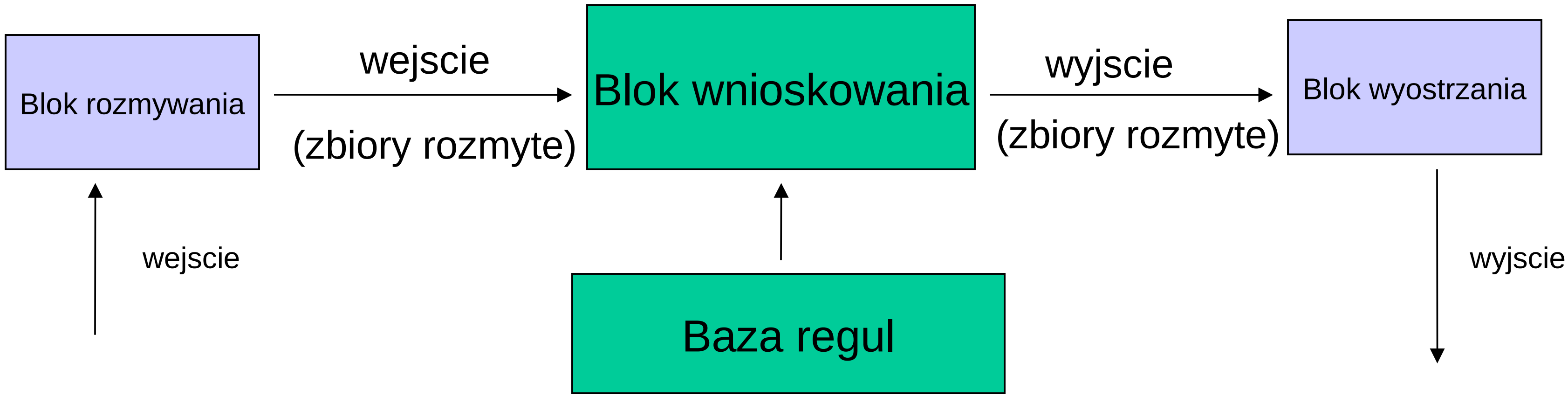



**system rozmyty**




**system przynależności ostrej**

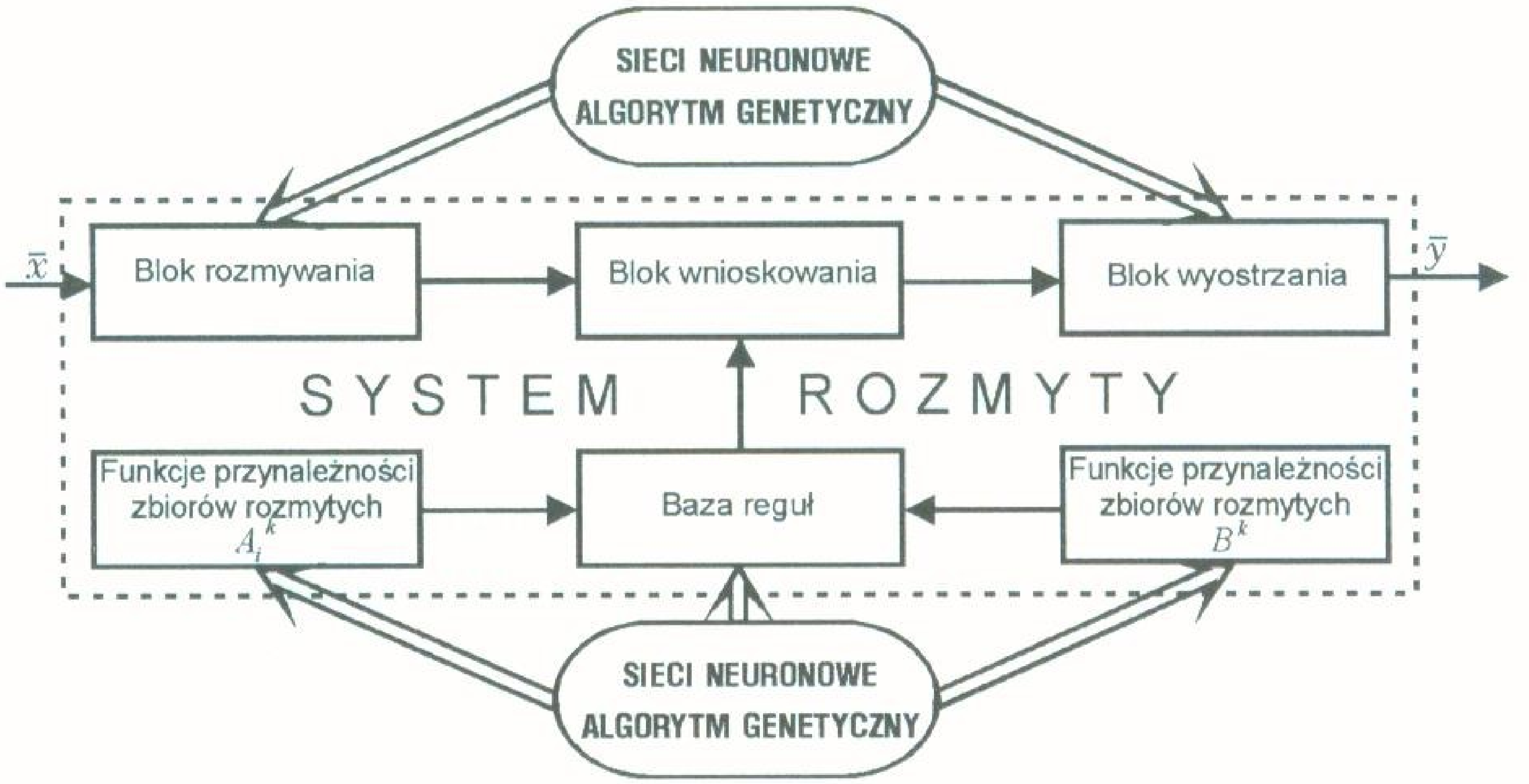
# Systemy rozmyte




Algorytm sterowania jest oparty o zbiory rozmyte i logikę rozmyta. Wprowadzane i wyprowadzane są dane 'ostre' , dodatkowe bloki realizują funkcje "rozmywania" i "wyostrzania".

Blok wnioskowania może być realizowany przy pomocy sieci neuronowych pracujących z wartościami rozmytymi. Takie systemy nazywamy tez *systemami rozmyto-neuronowymi*.

# Inteligentny system obliczeniowy




Rys. 24.1. Ogólna koncepcja inteligentnego systemu obliczeniowego

# Uwagi końcowe

Sztuczne sieci neuronowe maja następujące, podobne do mózgu cechy:

→ zdolności pamięciowe, zwłaszcza pamięci adresowanej kontekstowo, skojarzeniowej

→ umiejętność uczenia się na przykładach

→ umiejętność generalizacji

→ odporność na uszkodzenia sieci

→ umiejętność równoległego przetwarzania informacji

→ umiejętność pracy niealgorytmicznej

→ mogą heurystycznie rozwiązywać problemy

→ mogą poprawnie pracować przy pewnym poziomie uszkodzeń sieci

Przyjmuje się powszechnie, że dzisiejsze sztuczne sieci neuronowe są pozbawione wielu cech odpowiadających wyższym czynnościom mózgowym, jak np. umiejętność abstrakcyjnego myślenia i świadomości.