Wprowadzenie
Matematyczny model . . .
Sieci rekurencyjne
Modele samoorganizacji

Wykład 9: Sztuczne sieci neuronowe

Nguyen Hung Son
son@mimuw.edu.pl

Streszczenie

Wprowadzenie
Matematyczny model . . .
Sieci rekurencyjne
Modele samoorganizacji

Home Page
Title Page
Page 2 of 38
Go Back
Full Screen
Close
Quit

1. Wprowadzenie

"I cannot articulate enough to express my dislike to people who think that understanding spoils your experience... How would they know?"

Marvin Minsky

1.1. Historia “sztucznych sieci neuronowych”:

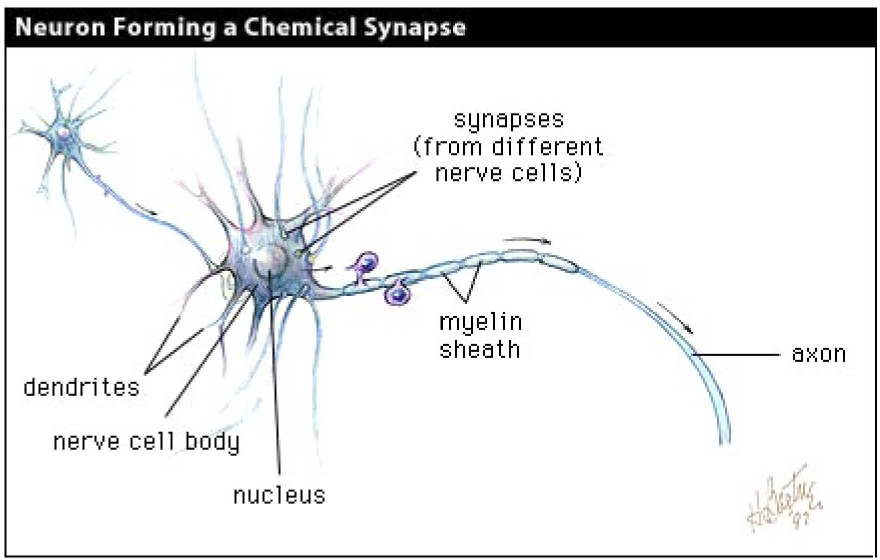
- Dawno, dawno temu: powstały neurony ...
- 1868 J. C. Maxwell: zbadał mechanizm sprzężenia zwrotnego;
- 1942 Weiner et al.: stworzył “cybernetykę”;
- 1943 McCulloch-Pitts: Neurony
- 1949 Hebb: wskazał mechanizm biologiczny;
- 1958-1962: Perceptron Rosenblatta: “Perceptron Separated Input from Output at Computer Level.”
- 1960 Widrow and Hoff: Sieci ADALINE / MADALINE
- 1969 Minsky and Papert: Krytyczna frustracja do sieci neuronowych: “Limited Advancement”
- 1972 Amari, Anderson, Grossberg: Badanie nad “Self Organizing Network”, Symulacja modelu pamięci asocjacyjnej u człowieka.
- 1974 Werbos: Propagacja wsteczna;
- 1982 John Hopfield: sieć rekurencyjna;
- 1986 McLelland and Rumelhart: Popularyzowali metodę propagacji wstecznej w swojej książce;
- 1990 Mead: integracja sprzętowa i zastosowanie sieci w obronie i przemyśle.

Wprowadzenie
Matematyczny model . . .
Sieci rekurencyjne
Modele samoorganizacji

1.2. Inspiracje neurofizjologiczne

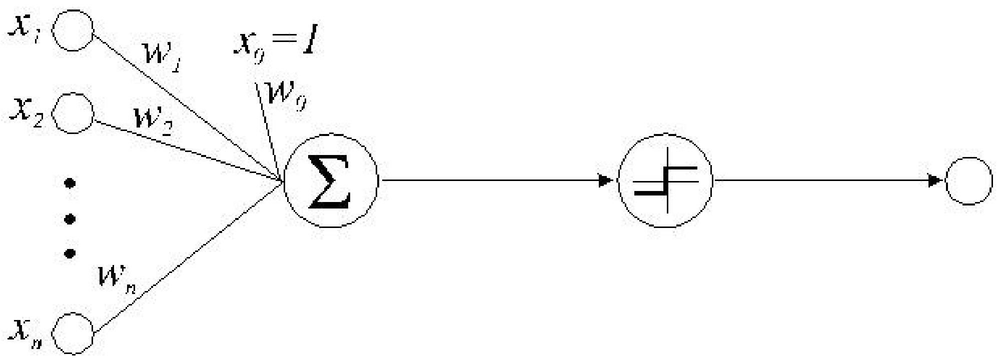
- Mózg ludzki:
  - objętość $\approx$ 1400 cm^3 i powierzchnie 2000 cm^2 (kula o tej samej objętości ma zaledwie 600 cm2).
  - Masa mózgu wynosi 1,5 kg.
  - Są to wartości średnie, ponieważ są w tym zakresie różnice pomiędzy różnymi ludźmi.
  - Dla przykładu mózgi kobiet są z reguły mniejsze niż mężczyzn. Nie udowodniono do tej pory związku między wielkością mózgu a jego intelektualną sprawnością.
- Duże znaczenie dla intelektualnych zdolności ma natomiast kora mózgowa o grubości średnio 3 mm, zawierająca:
  - 10^{10} komórek nerwowych i
  - 10^{12} komórek glejowych.
  - Liczba połączeń miedzy komórkami to około 10^{15}.

- Komórki nerwowe wysyłają i przyjmują impulsy o częstotliwości 1 - 100 Hz, czas trwania 1 - 2 ms, napięciu 100 ms i szybkości propagacji 1 - 100 m/s.
- Szybkość pracy mózgu szacuje się na 10^{18} operacji/s.
- Dla wykonania typowej reakcji mózg wykonuje nie więcej niż 100 elementarnych kroków, ponieważ czas reakcji jest mniejszy niż 300 ms.
- Pojemności kanałów zmysłów można szacować jako: wzrok 100 Mb/s, dotyk 1 Mb/s, słuch 15 Kb/s, węch 1 Kb/s, smak 100 b/s.



1.3. Sztuczna sieć neuronowa

- Sieć neuronowa jest uproszczonym modelem mózgu. Składa się ona z pewnej liczby elementów (neuronów) przetwarzających informacje.
- Neurony są ze sobą powiązane za pomocą połączeń o parametrach (wagach) modyfikowanych w trakcie procesu uczenia.



- Większość budowanych sieci ma budowę warstwową.

1.4. Zastosowania sieci

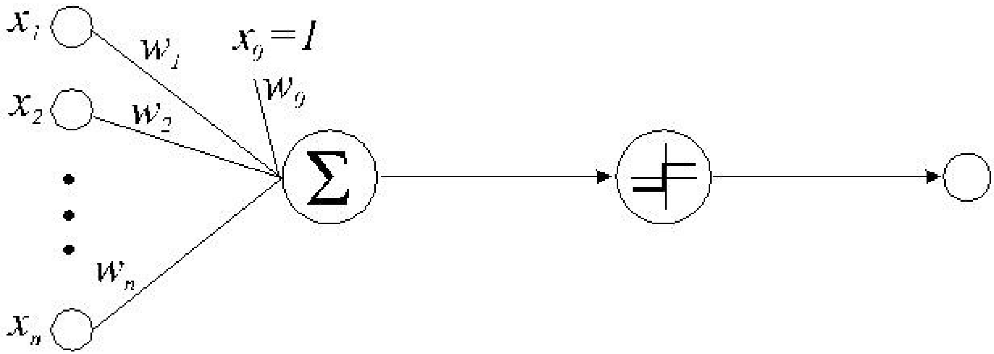
- **Predykacja:** Sieci neuronowe są często wykorzystywane, aby na podstawie pewnych danych wejściowych przewidywała dane wyjściowe. Ważną zaletą jest to, że sieć może nauczyć się przewidywania sygnałów wyjściowych bez jawnego definiowania związku między danymi wejściowymi a wyjściowymi.
- **Klasyfikacja i rozpoznawanie:** Zadanie polega na przewidywaniu identyfikatora klasy, do której dany obiekt należy.
- **Kojarzenie danych:** Klasyczne systemy komputerowe mogą gromadzić duże zbiory danych bez możliwości ich kojarzenia. Sieci neuronowe, dzięki zdolności uczenia się i uogólniania doświadczeń, pozwalają zautomatyzować procesy wnioskowania i pomagają wykrywać istotne powiązania pomiędzy danymi.
- **Analiza danych:** Zadanie polega na znalezieniu związków pomiędzy danymi. Realizacja tego zadania przez sieci neuronowe daje nowe możliwości w zakresie prowadzenia analiz ekonomicznych.
- **Filtracja sygnałów** Dane gospodarcze pochodzące z różnych 1ródeł są zakłócone. Klasyczne metody eliminacji śzumów”pozwalają usunąć zakłócenia o charakterze losowym, lecz nie dają podstaw do eliminacji przekłamań systematycznych.
- **Optymalizacja** Sieci neuronowe - zwłaszcza sieci Hopfielda - dobrze nadają się do optymalizacji decyzji gospodarczych. Doświadczalnie potwierdzono możliwości sieci do rozwiązywania zadań optymalizacji statycznej i dynamicznej. Szczególnie ciekawe jest zastosowanie sieci do optymalizacji kombinatorycznej i zagadnień bardzo trudnych obliczeniowo (również np-zupełnych), które mogą być rozwiązane w krótkim czasie dzięki współbieżnym wyliczaniu przez wiele elementów sieci. Dobrym przykładem jest sieć rozwiązująca klasyczny problem komiwojażera.

1.5. właściwości sieci

- *Adaptowalność i samo-organizacja*: oparte o zasady uczenia adaptacyjnego (podstawa filozoficzna)
- *Nieliniowość* : oferuje możliwości uczenia się aproksymacji funkcji i klasyfikacji.
- *Przetwarzanie równoległe*: sprzętowa realizacja sieci pozwala przetwarzać dużą ilość obliczeń dzięki równoległych połączeń.

Wprowadzenie
Matematyczny model...
Sieci rekurencyjne
Modele samoorganizacji

2. Matematyczny model neuronu



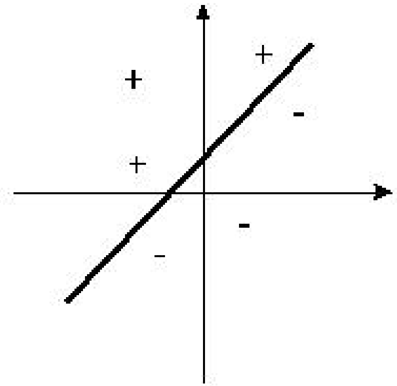
$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{jeśli } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{w p.p.} \end{cases}$$

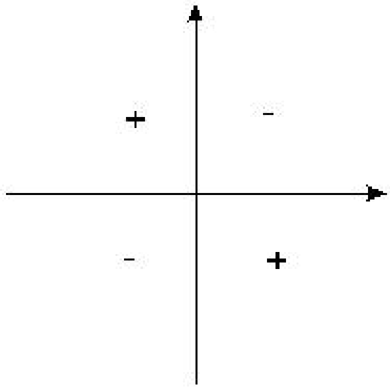
Czasem stosujemy prostsze obliczenia:

$$o(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x} > 0 \\ -1 & \text{otherwise.} \end{cases}$$

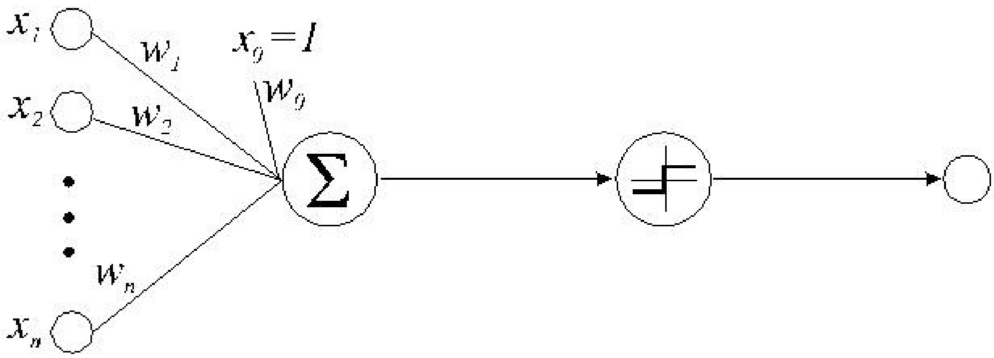
Ogólnie, perceptron realizuje pewną funkcję!

Perceptron ma dość ograniczone możliwości. Z powodzeniem może reprezentować funkcję OR, AND, ale już nie poradzi sobie z funkcją XOR. Ogólnie funkcje reprezentowane przez perceptron to takie, które rozdzielają liniowo obszar danych:





2.1. funkcja aktywacji:



- Funkcja progowa (threshold function) $sgn(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases}$
- Funkcja sigmoidalna:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

  Własności: $\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$

- Funkcja tangensu hiperbolicznego:

$$th(x) = \frac{1}{2}[1 + \tanh(x)]$$

Wprowadzenie
Matematyczny model . . .
Sieci rekurencyjne
Modele samoorganizacji

Home Page
Title Page
Page 12 of 38
Go Back
Full Screen
Close
Quit

3. Uczenie perceptronu

3.1. Problem klasyfikacji

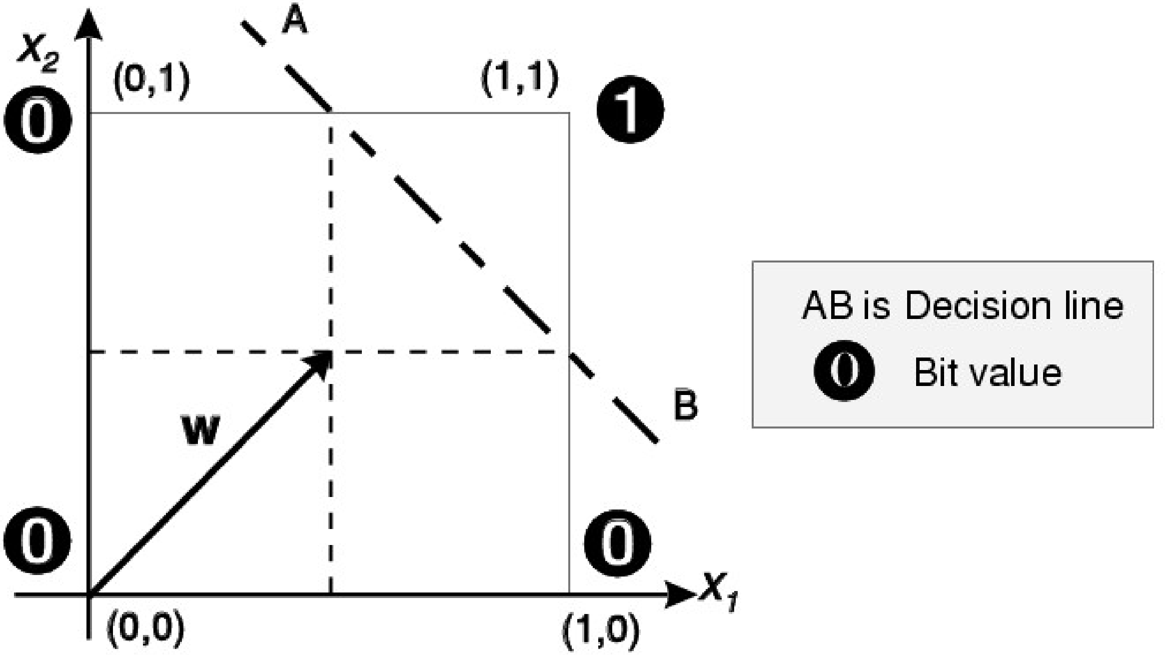
Przykłady:

- Klasyfikacja obrazów
- Predykcja finansowa
- Sterowanie
- funkcje boolowskie:

x_1	x_2	AND
0	0	0
1	0	0
0	1	0
1	1	1

Problem uczenia się perceptronu: Znaleźć wartości wag dla perceptronu, które definiują funkcję podobną do wzorców.

Przykład funkcja AND:



AND gate with $\mathbf{w} = (1/2, 1/2, 3/4)$

3.2. Reguły uczenia się:

$$w_i \leftarrow w_i + \Delta w_i$$

gdzie

$$\Delta w_i = \eta(t - o)x_i$$

Oznaczenia:

- $t = c(\vec{x})$ wartość funkcji celu (wzorca)
- o jest wartością wyjściową neuronu
- η jest małą stałą, np. $= 0.1$, zwaną stałą uczenia się

Można wykazać teoretyczną zbieżność algorytmu uczenia się w sytuacji:

- Dane treningowe są liniowo separowalne
- η jest dostatecznie mała.

Wprowadzenie
Matematyczny model . . .
Sieci rekurencyjne
Modele samoorganizacji
Home Page
Title Page
Page 15 of 38
Go Back
Full Screen
Close
Quit

Wyjaśnienia reguł uczenia perceptronu: rozważmy najprostszy przypadek

$$o = w_0 + w_1 x_1 + \cdots + w_n x_n$$

Szukajmy w_i, które minimalizują funkcję błędu:

$$E[\vec{w}] \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

gdzie D jest zbiorem przykładów treningowych.
Metoda gradientu:

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n} \right]$$

Reguły gradientu:

$$\Delta\vec{w} = -\eta \nabla E[\vec{w}]$$

i.e.,

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

$$\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_d (t_d - o_d)^2 \\
&= \frac{1}{2} \sum_d \frac{\partial}{\partial w_i} (t_d - o_d)^2 \\
&= \frac{1}{2} \sum_d 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) \\
&= \sum_d (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - \vec{w} \cdot \vec{x_d}) \\
\frac{\partial E}{\partial w_i} &= \sum_d (t_d - o_d)(-x_{i,d})
\end{aligned}$$

Wprowadzenie
Matematyczny model . . .
Sieci rekurencyjne
Modele samoorganizacji

Home Page
Title Page
Page 17 of 38
Go Back
Full Screen
Close
Quit

3.3. Gradient Descent

Gradient-Descent$(training\_examples, \eta)$

Each training example is a pair of the form $\langle \vec{x}, t \rangle$, where $\vec{x}$ is the vector of input values, and t is the target output value. η is the learning rate (e.g., .05).

- Initialize each w_i to some small random value
- Until the termination condition is met, Do
  - Initialize each Δw_i to zero.
  - For each $\langle \vec{x}, t \rangle$ in $training\_examples$, Do
    - Input the instance $\vec{x}$ to the unit and compute the output o
    - For each linear unit weight w_i, Do

$$\Delta w_i \leftarrow \Delta w_i + \eta(t - o)x_i$$

  - For each linear unit weight w_i, Do

$$w_i \leftarrow w_i + \Delta w_i$$

3.4. Incremental (Stochastic) Gradient Descent

Batch mode Gradient Descent:
Do until satisfied

1. Compute the gradient $\nabla E_D[\vec{w}]$
2. $\vec{w} \leftarrow \vec{w} - \eta \nabla E_D[\vec{w}]$

Incremental mode Gradient Descent:
Do until satisfied

- For each training example d in D
  1. Compute the gradient $\nabla E_d[\vec{w}]$
  2. $\vec{w} \leftarrow \vec{w} - \eta \nabla E_d[\vec{w}]$

$$E_D[\vec{w}] \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

$$E_d[\vec{w}] \equiv \frac{1}{2} (t_d - o_d)^2$$

Incremental Gradient Descent can approximate *Batch Gradient Descent* arbitrarily closely if η made small enough

3.5. Podsumowania

Reguły uczenia perceptronu gwarantuje sukces wtedy, gdy:

- Przykłady treningowe są liniowo separowane
- Dostatecznie mały jest wsp. uczenia się η

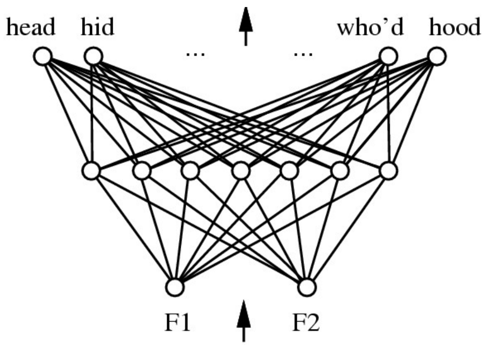
Reguły uczenia się perceptronu stosuje zasadę maksymalnego spadku gradientu:

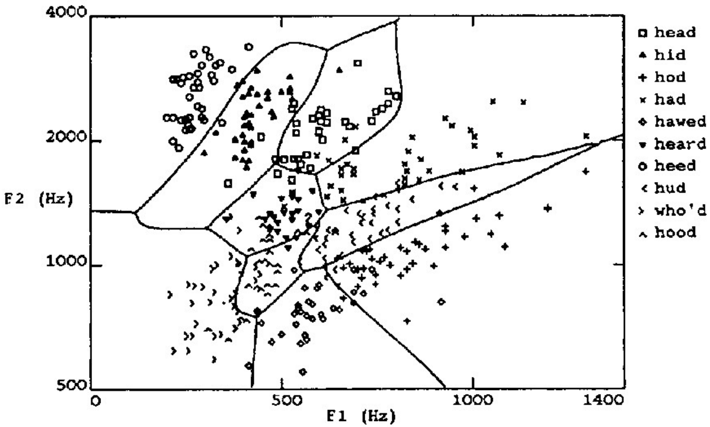
- Gwarantuje zbieżność do rozwiązań o (lokalnym) minimalnym błędzie
- Przy dostatecznie małym wsp. uczenia się η
- Działa nawet na danych szumiących.
- Działa nawet wtedy, gdy dane nie są liniowo separowane.

Wprowadzenie
Matematyczny model . . .
Sieci rekurencyjne
Modele samoorganizacji

Home Page
Title Page
Page 20 of 38
Go Back
Full Screen
Close
Quit

4. Sieci wielowarstwowe





Wprowadzenie
Matematyczny model...
Sieci rekurencyjne
Modele samoorganizacji

Home Page
Title Page
Page 21 of 38
Go Back
Full Screen
Close
Quit

4.1. perceptronu z sigmoidalną funkcją aktywacji

$$
\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 = \frac{1}{2} \sum_d \frac{\partial}{\partial w_i} (t_d - o_d)^2 \\
&= \frac{1}{2} \sum_d 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) = \sum_d (t_d - o_d) \left(- \frac{\partial o_d}{\partial w_i} \right) \\
&= - \sum_d (t_d - o_d) \frac{\partial o_d}{\partial net_d} \frac{\partial net_d}{\partial w_i}
\end{aligned}
$$

Wiemy, że

$$
\frac{\partial o_d}{\partial net_d} = \frac{\partial \sigma(net_d)}{\partial net_d} = o_d(1 - o_d)
$$

$$
\frac{\partial net_d}{\partial w_i} = \frac{\partial (\vec{w} \cdot \vec{x}_d)}{\partial w_i} = x_{i,d}
$$

Więc

$$
\frac{\partial E}{\partial w_i} = - \sum_{d \in D} (t_d - o_d) o_d (1 - o_d) x_{i,d}
$$

Wprowadzenie
Matematyczny model . . .
Sieci rekurencyjne
Modele samoorganizacji

4.2. Backpropagation Algorithm

Initialize all weights to small random numbers.
Until satisfied, Do

- For each training example, Do
  1. Input the training example to the network and compute the network outputs
  2. For each output unit k
  $$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$
  3. For each hidden unit h
  $$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{h,k}\delta_k$$
  4. Update each network weight $w_{i,j}$
  $$w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$$
  where
  $$\Delta w_{i,j} = \eta \delta_j x_{i,j}$$

- Gradient descent over entire *network* weight vector
- Easily generalized to arbitrary directed graphs
- Will find a local, not necessarily global error minimum
  - In practice, often works well (can run multiple times)
- Often include weight *momentum* α

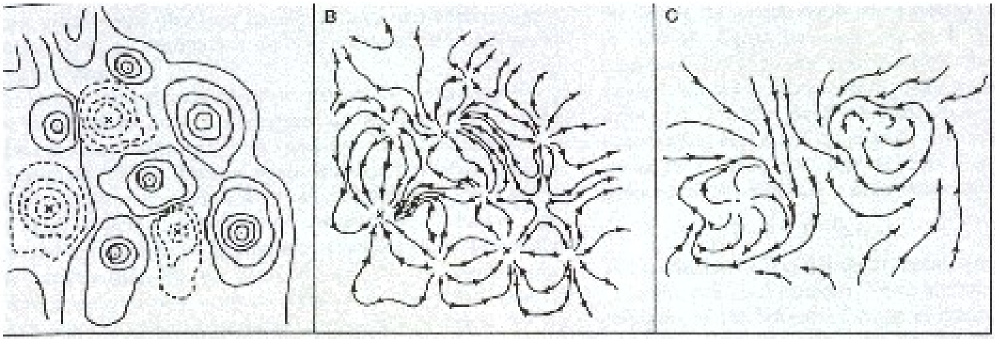
$$\Delta w_{i,j}(n) = \eta \delta_j x_{i,j} + \alpha \Delta w_{i,j}(n-1)$$

- Minimizes error over *training* examples
  - Will it generalize well to subsequent examples?
- Training can take thousands of iterations $\rightarrow$ slow!
- Using network after training is very fast

Wprowadzenie
Matematyczny model . . .
Sieci rekurencyjne
Modele samoorganizacji

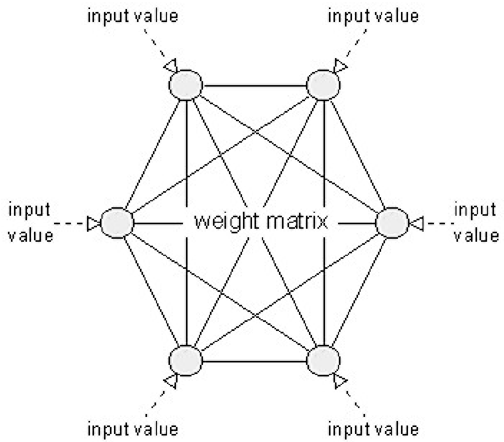
5. Sieci rekurencyjne

Idea:



Wprowadzenie
Matematyczny model . . .
Sieci rekurencyjne
Modele samoorganizacji

5.1. Sieci Hopfielda



- Zbadane przez Hopfielda;
- Zwane również sieciami auto-asocjacyjnymi;
- Są to jedno-warstwowe sieci z pełnym połączeniem typu “każdy z każdym”;
- Każdy neuron ma bipolarne wartość wejść i wyjść;

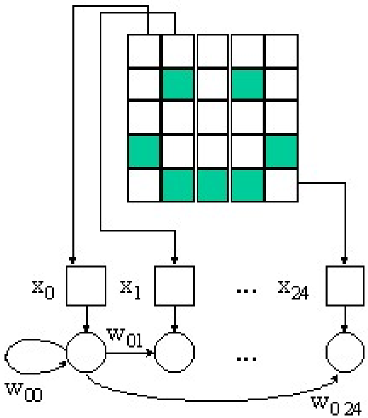
5.2. Operacje w sieciach Hopfielda:

- Zainicjuj wartości początkowe do sieci;
- Czekaj aż sieć się ustabilizuje;
  - To działa w czasie dyskretnym: $t_1, t_2, ..., t_N$;
  - wartości neuronów w chwili t_n zależy od wartości neuronów w chwili t_{n-1}:

$$u_i(t_n) = \sum_j w_{ij} y_j(t_{n-1}) + x_i$$

$$y_i(t_n) = \begin{cases} 1 & \text{jeśli } u_i(t_n) > T_i \\ y_i(t_{n-1}) & \text{gdy } u_i(t_n) = T_i \\ -1 & \text{jeśli } u_i(t_n) < T_i \end{cases}$$

  - Istnieją modele synchroniczne i asynchroniczne;
- Odczytaj wartości neuronów jako wynik obliczeń;



5.3. Ustalenie wartości wag:

UWAGA: wagi się nie zmieniają w procesie uczenia się;
Maciecz wag musi spełniać warunki:

- symetryczność:

$$w_{ij} = w_{ji}$$

- zerowa przekątna:

$$w_{ii} = 0$$

Zwykle chcemy, aby sieć ustabilizowała w jednym z wektorów: $\vec{v_1}, ..., \vec{v_P}$, gdzie

$$v_p = (x_1^p, ...x_n^p)$$

Aby to zapewnić możemy stosować reguły Hebba:

$$w_{ij} = \sum_{p=1}^{P} x_i^p x_j^p$$

Wprowadzenie
Matematyczny model . . .
Sieci rekurencyjne
Modele samoorganizacji

5.4. Dlaczego to działa?

- Charakterystyczną własnością dla bieżącego stanu sieci Hopfielda jest funkcja energii:

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_i x_j - \sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} T_i y_i$$

- Hopfield pokazał, że funkcja energii jest nierosnąca;

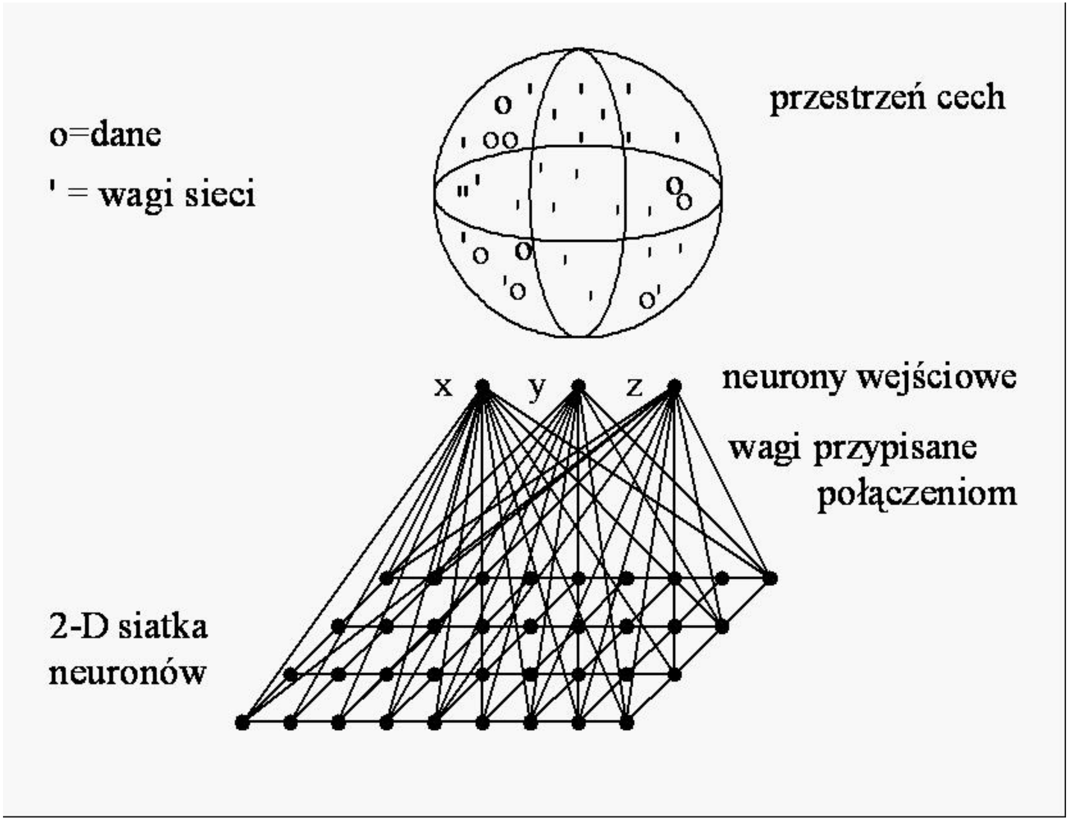
Wprowadzenie
Matematyczny model . . .
Sieci rekurencyjne
Modele samoorganizacji

6. Modele samoorganizacji

- SOM (Self-Organized Maps) - samorganizująca się mapa.
- Mapy topograficzne powstaje przez połączenia lokalne: neuron silnie pobudzany przez pobliskie połączenie, słabo przez odległe, hamowany przez neurony pośrednie.
- Historia: von der Malsburg i Willshaw (1976), uczenie konkurencyjne, mechanizm Hebbowski, wzbudzenie typu Meksykańskiego kapelusza, model układu wzrokowego. Amari (1980) - model ciągłej tkanki neuronowej. Kohonen (1981) - uproszczenie, bez hamowania; dwie fazy - konkurencja i kooperacja.

6.1. Algorytm SOM

Struktura: Neurony są umieszczone (ale nie połączone ze sobą) na siatce 1,2 lub 3-wymiarową. Każdy neuron ma N wag. Neuron i-ty ma wagi $\vec{W}_i(t) = (w_{i1}, \dots, w_{iN})$, a wektor wejściowy ma współrzędne: $(x_1, \dots, x_N)$ (t - czas dyskretny)



Wprowadzenie
Matematyczny model . . .
Sieci rekurencyjne
Modele samoorganizacji

Algorytm :

1. Inicjalizacja: przypadkowe $W_i(0)$ dla wszystkich $i = 1..K$. Definiuj wokół neuronu położonego w miejscu r_c siatki obszar sąsiedztwa $Os(r_c, \sigma(t))$ o promieniu $\sigma(t)$.
2. Oblicz odległości

$$d(X, W_i) = ||X - W_i|| = \sqrt{\sum_j (x_j - w_{ij})^2}$$

3. znajdź neuron

$$c = \mathrm{argmin}_i ||X - W_i||$$

   z wagami W_c najbardziej podobnymi do X (neuron-zwycięzcę).
4. Zmień wagi wszystkich neuronów r_i w sąsiedztwie $Os(r_c, \sigma(t))$

$$W_i(t+1) = W_i(t) + h(r_i, r_c)(X - W_i(t))$$

   where

$$h(r_i, r_c) = h_0(t) \cdot e^{-||r - r_c||^2/\sigma_c^2(t)}$$

5. Powoli zmniejszaj siłę $h_0(t)$ i promień $\sigma(t)$.
6. Iteruj aż ustaną zmiany.

Efekt: podział (tesselacja) na wieloboki Voronoia.

6.2. Własności SOM

- Brak dowodu o zbieżności lub punktach stacjonarnych dla SOM:
- Wyniki analityczne tylko w 1D dla ciągłego czasu, proces Markova: wartości wag wzdłuż prostej porządkują się. Powolna zbieżność: $10^4 - 10^6$ iteracji.
- Sąsiednie neurony kodują sąsiednie obszary, ale niekoniecznie odwrotnie Skręcone konfiguracje przy zbyt szybkiej redukcji sąsiedztwa. Złożoność O(KNn) dla K neuronów i n danych N-wymiarowych: konieczne porównanie wszystkich odległości; niezbyt duże mapy.
- Na komputerach wieloprocesorowych szukanie min z K będzie powolne.
- SOM działa jak metoda klasteryzacji k-średnich jeśli $\sigma = 0$.

6.3. Modyfikacje SOM

- Próba wprowadzenia funkcji błędu (Luttrell; Heskes i Kappen).
- Błąd lokalny neuronu i jest sumą po wszystkich neuronach:

$$E_i(X;t) = \frac{1}{2} \sum_j h(|r_i - r_j|, t) \cdot ||X - W_i(t)||^2$$

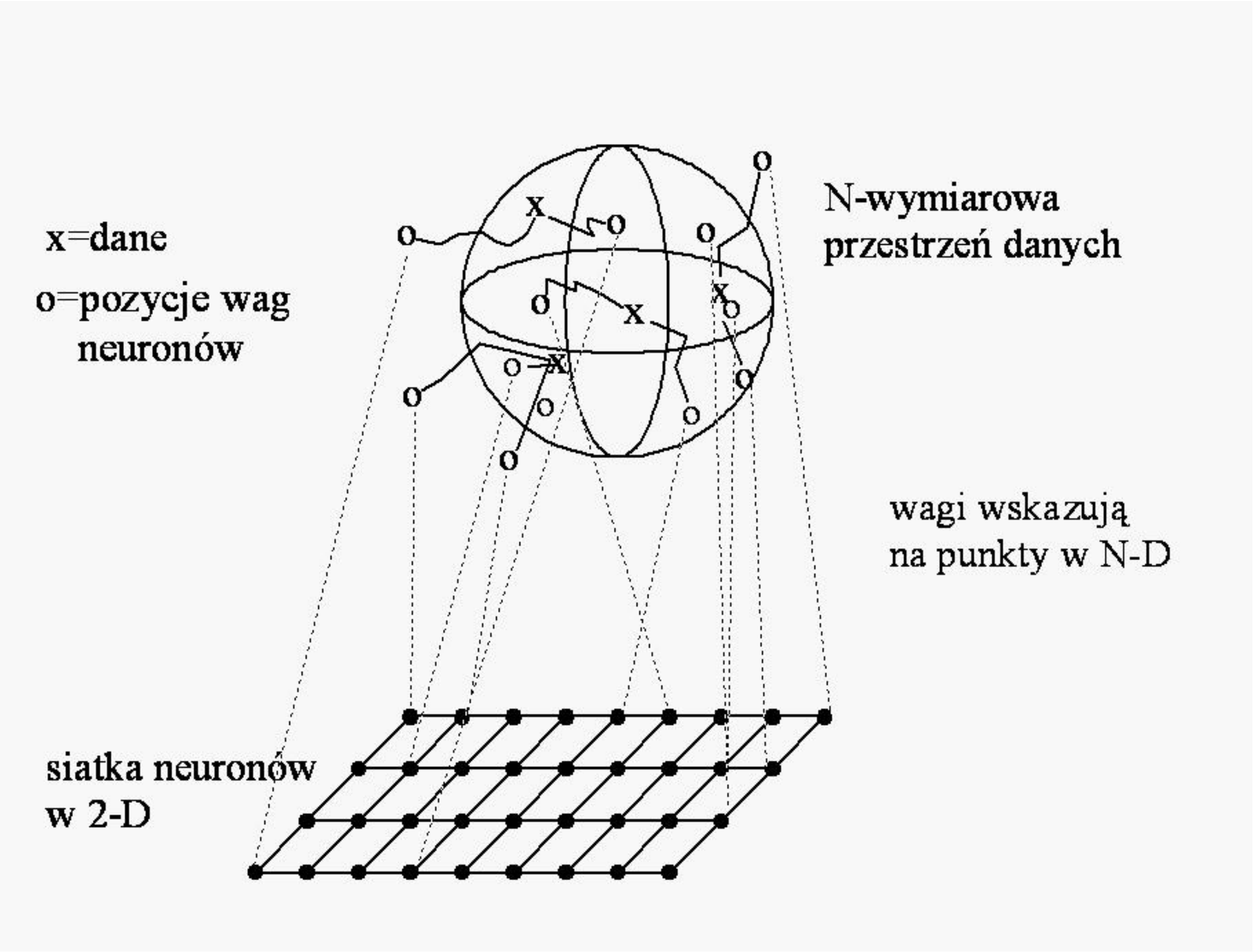
gdzie

$$h(|r_i - r_j|, t) = \exp -\frac{|r_i - r_j|^2}{2\sigma^2(t)}; \quad \sigma(t) = \sigma_0 e^{-2\sigma_0 t/t_{\max}}$$

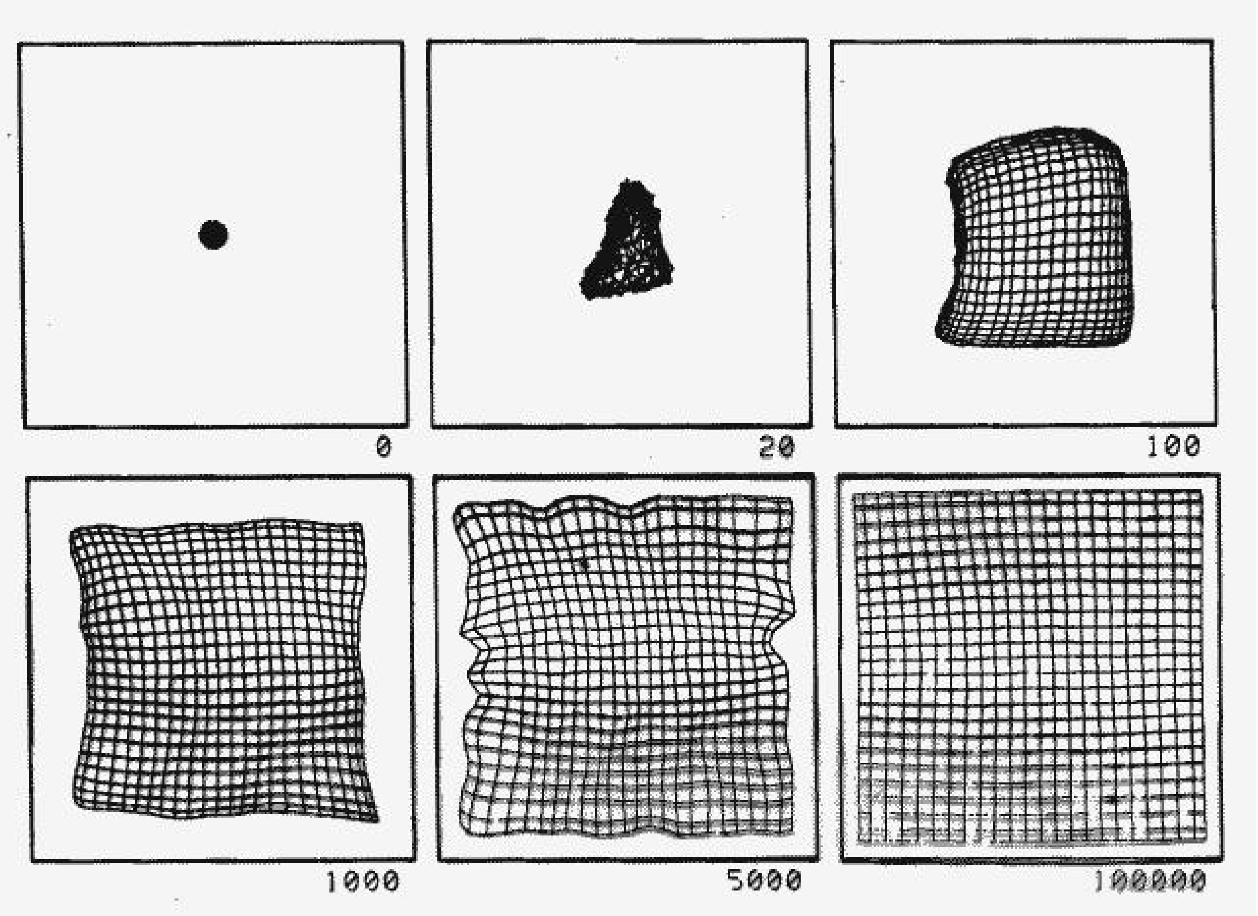
- Neuron-zwycięzca ma najmniejszy błąd lokalny:

$$c = \mathrm{argmin}_i E_i(X;t)$$

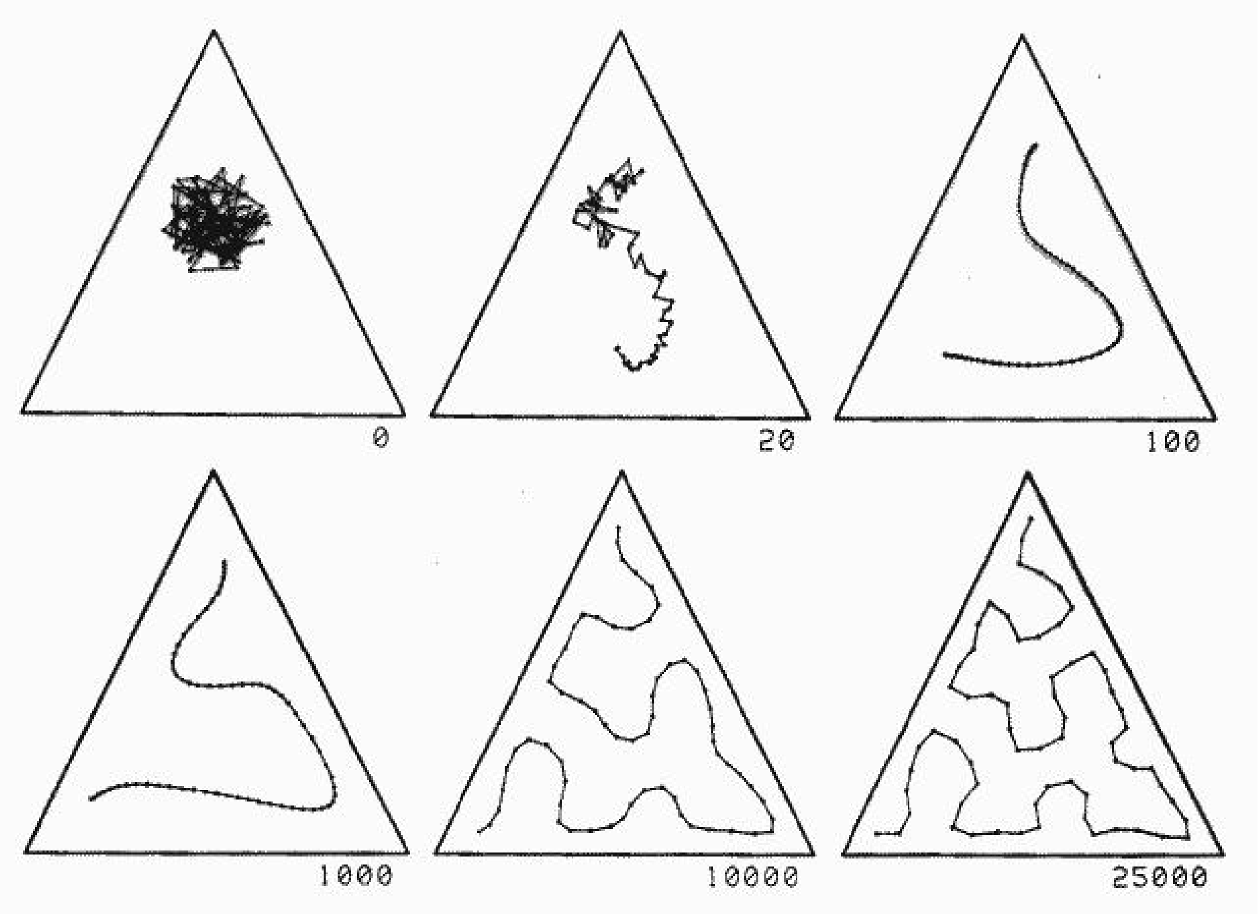
6.4. Uczenie sieci 2D



6.5. Uczenie kwadratu w sieci 2D

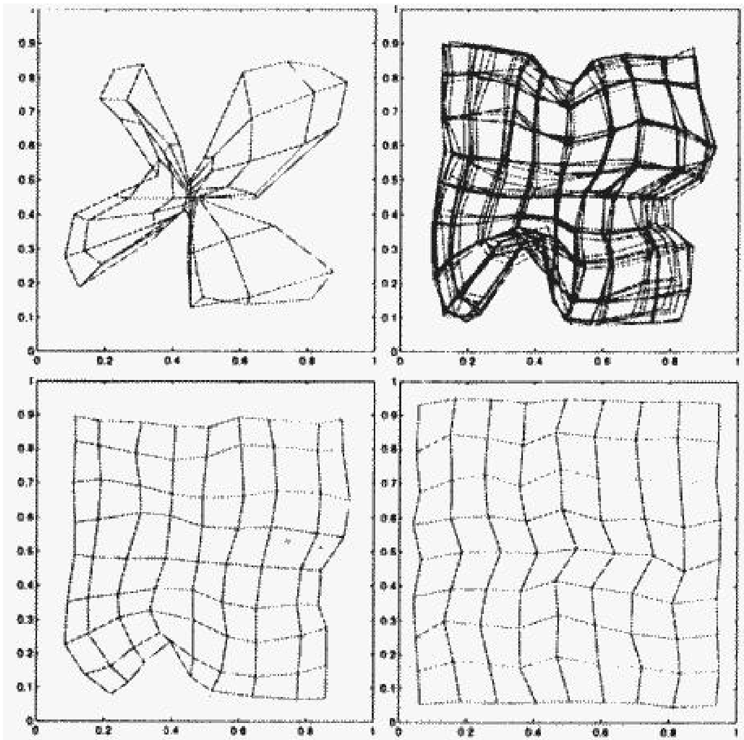


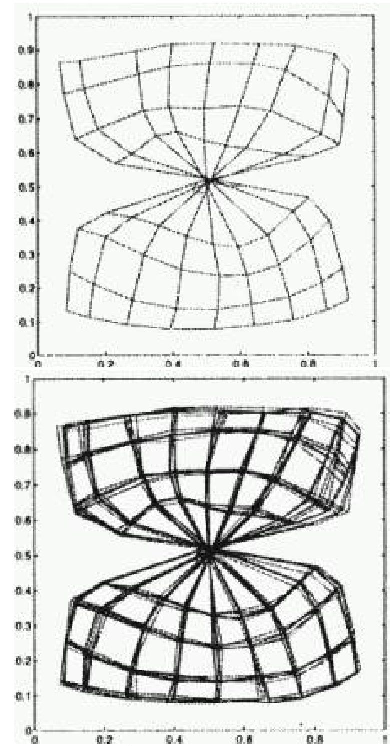
6.6. Uczenie trojkąta w sieci 1D



Tworzy się fraktalne krzywe Peano.

6.7. Zniekształcenie





Początkowe zniekształcenia mogą zniknąć lub pozostać.

Wprowadzenie
Matematyczny model . . .
Sieci rekurencyjne
Modele samoorganizacji
Home Page
Title Page
Page 38 of 38
Go Back
Full Screen
Close
Quit

6.8. WEBSOM websom.hut.fi/websom/