Samoorganizacja

Uczenie hebbowskie

Samoorganizacja

- **uczenie nienadzorowane**: wykrywa samoistnie (bez nadzoru) istotne cechy powiązań między sygnałami
- globalne uporządkowanie sieci jest możliwe przez działania samoorganizujące prowadzone **lokalnie** w różnych punktach sieci, niezależnie od siebie
- pewne neurony lub grupy neuronów **współpracują** lub/i **konkurują** ze sobą specjalizując się w wykrywaniu określonych wzorców

Sieci samoorganizujące się

Uczenie konkurencyjne

- reguła Kohonena
- konkurencja WTA, WTM
- kwantyzacja wektorowa, SOM

Uczenie hebbowskie

- reguła Hebba, reguła Oja
- asocjacje i korelacje między sygnałami
- dekompozycja składowych głównych PCA (Principal Component Analysis)
- dekompozycja na składowe niezależne ICA (Independent Component Analysis), separacja źródeł BSS (Blind Source Separation), nielinowe PCA, ...

Reguła Hebba (1949)

„Neurons, that fire together, wire together."

Jeżeli neuron A jest wielokrotnie pobudzany przez neuron B, to staje się on jeszcze bardziej czuły na pobudzenie tego neuronu

- synchroniczne pobudzanie neuronów A i B wzmacnia połączenie synaptyczne między nimi
- brak współpobudzenia nie prowadzi do wzmacniania połączenia

Zmiana wagi połączenia pomiędzy neuronami A i B

$$w_{AB}(k+1) = w_{AB}(k) + \eta y_A(k) y_B(k)$$

Reguła Hebba

Dla neuronu liniowego

$$y = \mathbf{w}^T \mathbf{x}$$

reguła Hebba przybiera postać

$$\Delta \mathbf{w} = \eta y \mathbf{x} = \eta \mathbf{w}^T \mathbf{x} \mathbf{x} = \eta \mathbf{x} \mathbf{x}^T \mathbf{w}$$

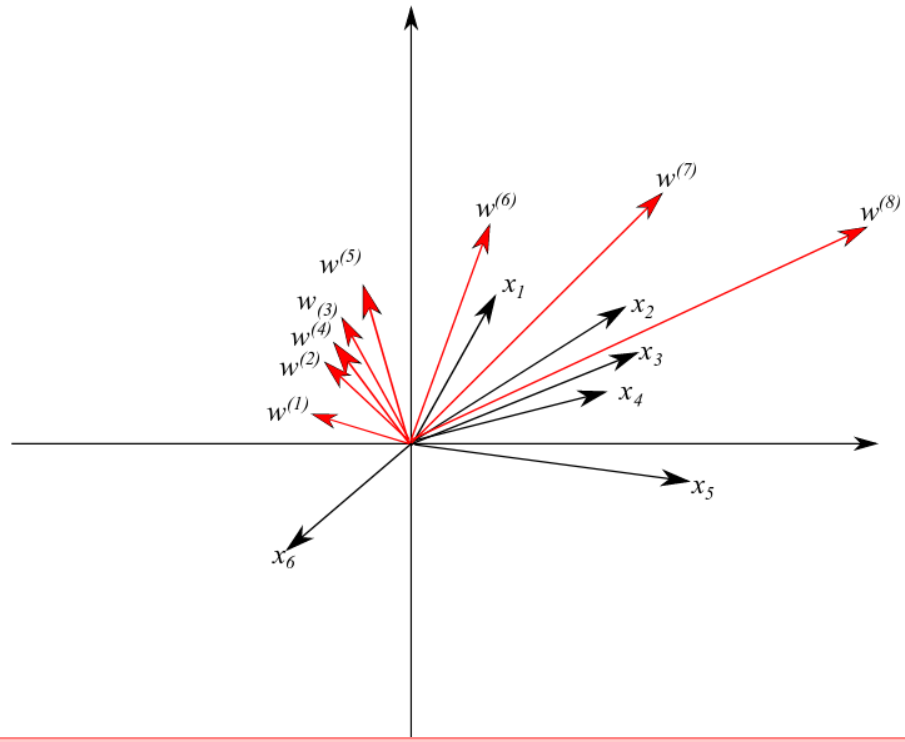
Decydujący wpływ na wartość końcową $\mathbf{w}$ będą miały sygnały zgrupowane (rozciągające się) wzdłuż znalezionego kierunku - maksymalizacja wariancji $\mathbf{w}^T \mathbf{x}$.

Przy prezentacji wielu próbek uczących wektor wag kumuluje wartości macierzy autokorelacji $\mathbf{R}$ wektorów $\mathbf{x}$

$$\Delta \mathbf{w} = \eta \mathbf{R} \mathbf{w} \quad \text{gdzie} \quad \mathbf{R} = E[\mathbf{x} \mathbf{x}^T]$$

Interpretacja geometryczna reguły Hebba

Wektor wag przemieszcza się w kierunku odpowiadającym największemu rozrzutowi danych (maleje kąt względem kierunku największej wariancji danych)



Trening prowadzi do nieograniczonego wzrostu wag (norma $\|\mathbf{w}\|$ rośnie w każdym kroku)

Modyfikacje reguły Hebba

Zapobieganie nieograniczonemu wzrostowi wag:

- sztywne granice

$$w_{min} < w_i < w_{max}$$

- miękkie granice

$$\Delta w_i = \eta x_i y (w_i - w_{min})(w_{max} - w_i)$$

- normalizacja wag po każdym kroku uczenia
- dodanie czynnika normalizującego wagi, np. reguła Oji (1982)

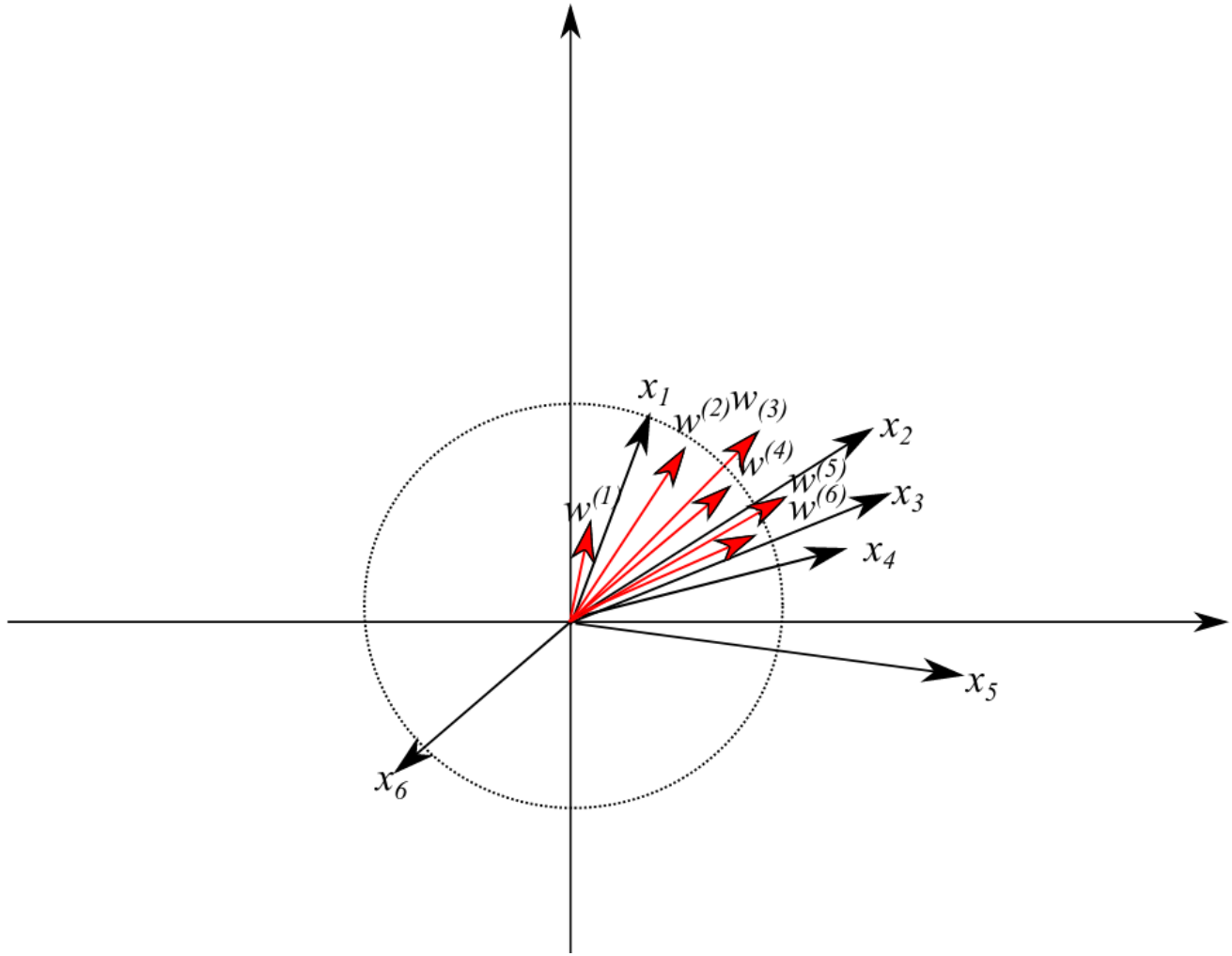
$$\Delta w_i = \eta y \left(x_i - y w_i\right)$$

Reguła Oji

$$\Delta \mathbf{w} = \eta \left(y\mathbf{x} - \alpha y^2 \mathbf{w} \right)$$

- dodanie wyrazu zanikania wagi proporcjonalnego do y^2
- adaptacja prowadzi do unormowania wag $\|\mathbf{w}\| = \frac{1}{\sqrt{\alpha}}$
- wektor wag zbliża się do wektora własnego macierzy kowariancji $E[\mathbf{X}\mathbf{X}^T]$ o największej wartości własnej
- wektor wag leży wzdłuż kierunku maksymalizującego wartość oczekiwaną y^2 (wariancja wartości wyjściowej neuronu)

Interpretacja geometryczna reguły Oji



Własności reguły Oji

Dla wyuczonego wektora wag wartość oczekiwana zmian dla wszystkich wektorów uczących

$$E[\Delta \mathbf{w}] = 0$$

stąd

$$E[\Delta \mathbf{w}] = \eta E[y(\mathbf{x} - y\mathbf{w})] = \eta E[(\mathbf{w}^T\mathbf{x})\mathbf{x} - y^2\mathbf{w}]$$
$$= \eta \left(E[\mathbf{x}\mathbf{x}^T]\mathbf{w} - E[y^2]\mathbf{w}\right) = 0$$

Macierz kowariancji $E[\mathbf{x}\mathbf{x}^T] = \mathbf{R}_x$.
Oznaczmy $\lambda = E[y^2]$

Reguła Oji prowadzi do wyznaczenia wektora własnego $\mathbf{w}$ macierzy kowariancji $\mathbf{R}_x$

$$\mathbf{R}_x\mathbf{w} = \lambda\mathbf{w}$$

Normalizacja wag

$$\lambda = E[y^2] = E[(\mathbf{w}^T\mathbf{x})(\mathbf{w}^T\mathbf{x})] = \mathbf{w}^T E[\mathbf{x}\mathbf{x}^T]\mathbf{w} = \mathbf{w}^T\mathbf{R}_x\mathbf{w}$$

ponieważ

$$\mathbf{R}_x\mathbf{w} = \lambda\mathbf{w}$$

stąd

$$\lambda = \mathbf{w}^T\lambda\mathbf{w} \quad \Rightarrow \quad \|\mathbf{w}\| = 1$$

Dodatkowy czynnik ogranicza wartość wag.

Analiza składowych głównych PCA

Transformacja liniowa

$$\mathbf{y} = \mathbf{W}\mathbf{x}$$

gdzie

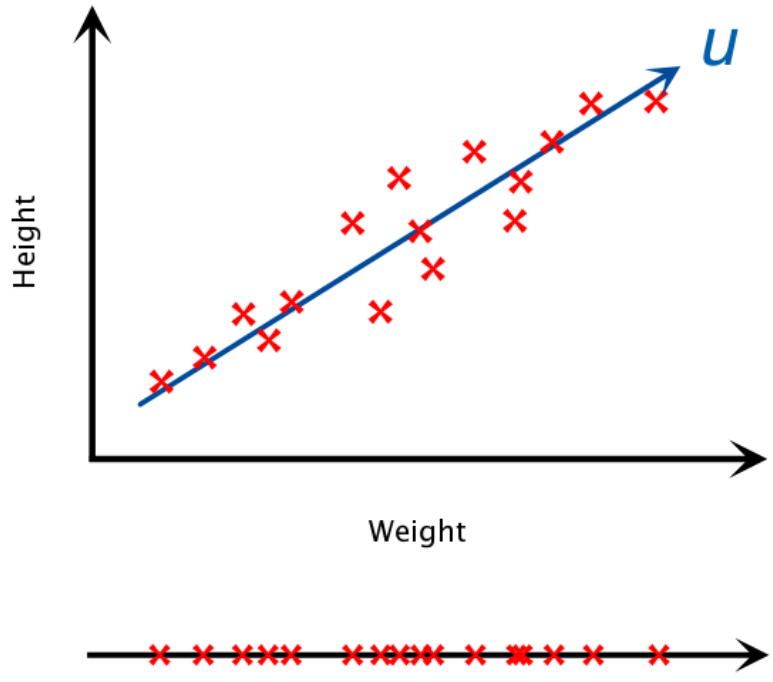
$$\mathbf{x} \in \mathbb{R}^N \qquad \mathbf{y} \in \mathbb{R}^K \quad \mathbf{W} \in \mathbb{R}^{K \times N} \qquad K < N$$

Redukcja wymiarowości z zachowaniem maksymalnej informacji zawartych w danych.

W PCA kierunek zawierający najwięcej informacji to kierunek o największej wariancji

Projekcja danych na wektor wag

Projekcja danych 2D na kierunek o największej wariancji



Rys: Boedecker, Machine Learning Summer 2015

PCA

Niech $\mathbf{X} \in \mathbb{R}^N$ wektor losowy o zerowej wartości oczekiwanej, wówczas wartość oczekiwana macierzy kowariancji

$$R_x = E[\mathbf{X}\mathbf{X}^T]$$

dla skończonej próbki o liczebności p

$$R_x \approx \frac{1}{p} \sum_{i=1}^{p} \mathbf{x}_i \mathbf{x}_i^T$$

PCA

Wartości własne λ_i oraz wektory własne $\mathbf{w}_i$ macierzy kowariancji

$$\mathbf{R}_x \mathbf{w}_i = \lambda_i \mathbf{w}_i \qquad i = 1, \ldots, N$$

R jest symetryczna i dodatnio półokreślona $\Rightarrow$ wartości własne są rzeczywiste i nieujemne
Niech

$$\lambda_1 > \lambda_2 > \ldots > \lambda_N \geq 0$$

Ograniczając do K pierwszych składowych

$$\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_k]^T$$

otrzymujemy transformację PCA

$$\mathbf{y} = \mathbf{W}\mathbf{x}$$

PCA

Wariancja wzdłuż i-tego kierunku

$$Var(\mathbf{w}_i^T \mathbf{x}) = E[(\mathbf{w}_i^T \mathbf{x} - E[\mathbf{w}_i^T \mathbf{x}])^2] = E[(\mathbf{w}_i^T \mathbf{R}_x \mathbf{w}_i)] = \lambda_i$$

Celem PCA jest znalezienie wektorów $\mathbf{w}_1, \ldots, \mathbf{w}_K$, które maksymalizują wariancję przy zachowaniu ortogonalności

$$\mathbf{w}_i^T \mathbf{w}_j = 0 \quad \text{dla} \quad j \neq i$$

$$\mathbf{w}_i^T \mathbf{w}_i = 1$$

Algorytm PCA

1. Wyśrodkowanie danych dla każdej cechy j

$$x_{ij} \leftarrow x_{ij} - \frac{1}{n}\sum_{j=1}^{n} x_{ij}$$

2. Wyznaczenie macierzy kowariancji

$$\mathbf{R}_x = \frac{1}{n}\mathbf{X}\mathbf{X}^T$$

3. Diagonalizacja za pomocą rozkładu SVD

$$\mathbf{S} = \mathbf{W}^{-1}\mathbf{R}_x\mathbf{W}$$

   $\mathbf{W}$ zawiera wektory własne, $\mathbf{S}$ wartości własne na diagonali

4. Projekcja danych na kierunki wektorów własnych odpowiadającym K największym wartościom własnym

$$\mathbf{y}_i = \mathbf{W}_K\mathbf{x}_i$$

Rekonstrukcja

Rekonstrukcja obrazu wejściowego

$$\hat{\mathbf{x}} = \mathbf{W}_k^T \mathbf{y}$$

PCA minimalizuje błąd rekonstrukcji

$$E_r = E[\|\mathbf{x} - \hat{\mathbf{x}}\|^2]$$

dla k składowych PCA

$$E_r = \sum_{i=k+1}^{N} \lambda_i$$

Minimalizacja E_r odpowiada maksymalizacji $\sum_{i=1}^{K} \lambda_i$

PCA

- Gdy zmienne są skorelowane to znajomość tylko części zmiennych pozwala określić pozostałe
- Względny wkład poszczególnych składowych

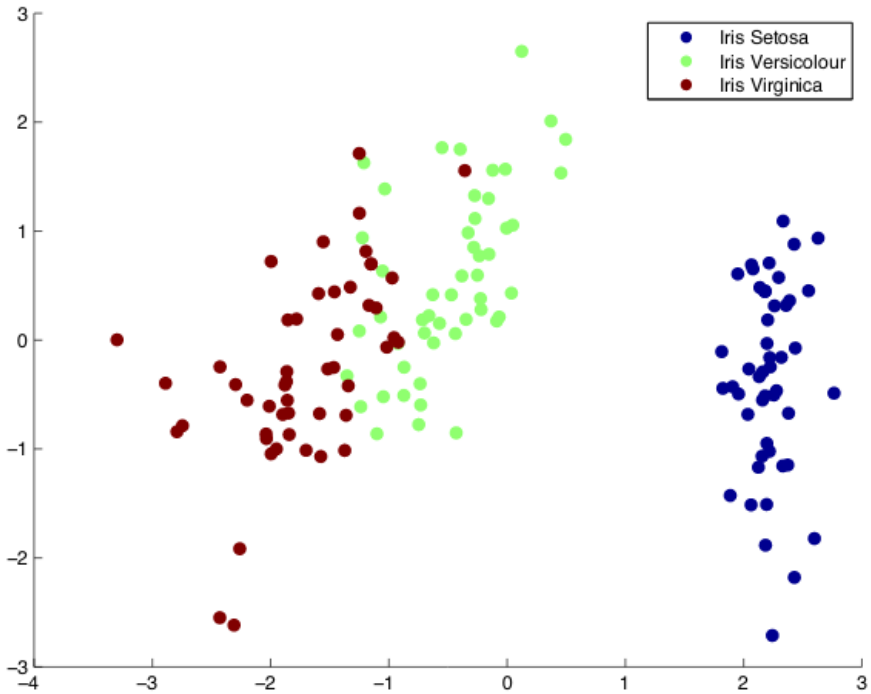
$$m_i = \frac{\lambda_i}{\sum \lambda_j}$$

- Standardowe metody wyznaczania wektorów własnych macierzy $\mathbf{R}_x$ dla wysokowymiarowych danych są złożone

Metody adaptacyjne wyznaczania składowych głównych (np. reguła Oja)

- nie wymagają obliczania macierzy kowariancji $\mathbf{R}_x$
- mogą być stosowane w metodach on-line, gdy nie mamy dostępu do wszystkich danych jednocześnie

Przykład: PCA na danych Iris

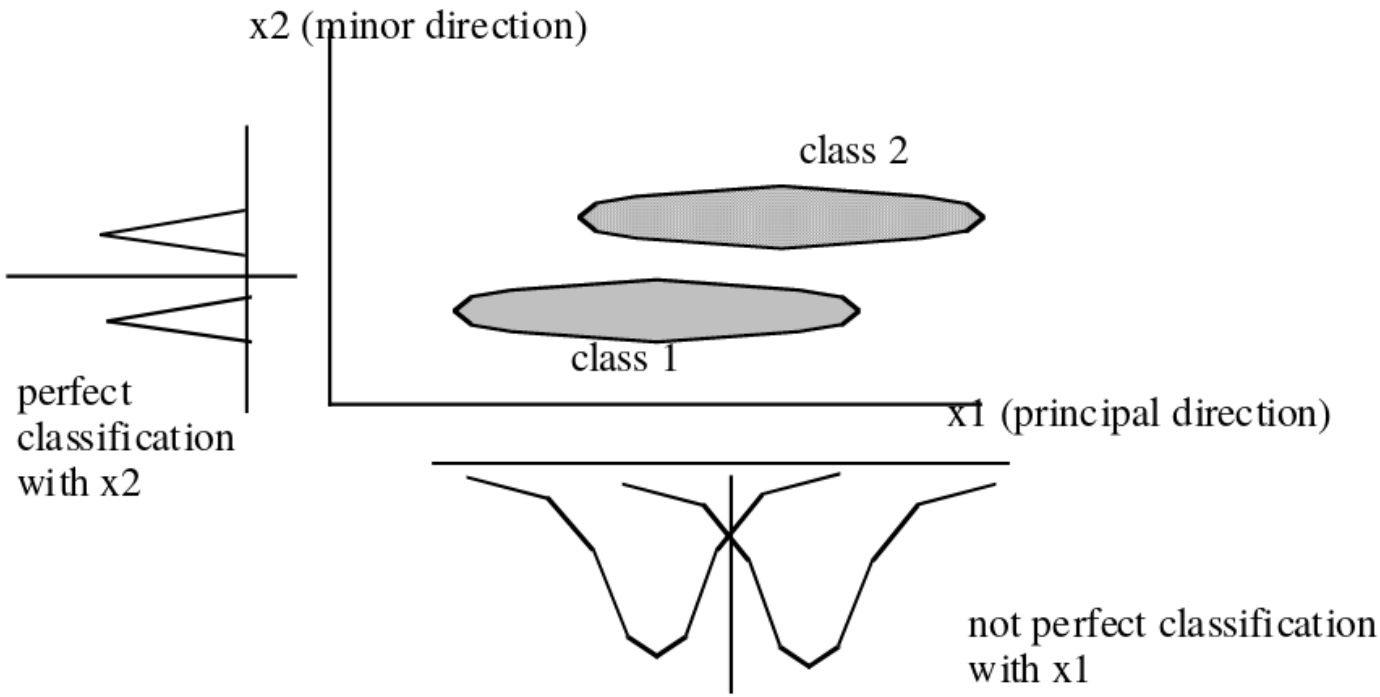


$$\lambda_1 = 2.8914 \quad \lambda_2 = 0.9151 \quad \lambda_3 = 0.1464 \quad \lambda_4 = 0.0205$$

PCA często używane do przetworzenia danych przed treningiem algorytmów uczenia maszynowego

Rys: Boedecker, Machine Learning Summer 2015

PCA i klasyfikacja



Projekcja PCA nie gwarantuje uzyskania najlepszej dyskryminacji

Rys: Principe, Neural and Adaptive Systems

Trening pierwszego kierunku

Estymator pierwszego kierunku (neuron liniowy)

$$y_1 = \mathbf{w}_1^T \mathbf{x} = \sum_{j=0}^{N} w_{1j} x_j$$

uczony regułą Oja

$$\mathbf{w}_1(k+1) = \mathbf{w}_1(k) + \eta(k) y_1(k) \left(\mathbf{x}(k) - \mathbf{w}_1(k) y_1(k)\right)$$

Dobór $\eta(k)$ jest istotny - powinien maleć z czasem, np.:

$$\eta(k) = \frac{\eta(0)}{k^\gamma} \qquad \text{gdzie} \qquad 0.5 \leq \gamma \leq 1$$

Sieci neuronowe uczone regułą Hebba

W warstwie składającej się z neuronów liniowych uczonych regułą Hebba wszystkie neurony będą zbiegały do tego samego kierunku maksymalizującego wariancję.

Mechanizmy konkurencji wymuszające różnorodność rozwiązań:

- WTA, zwycięzca bierze wszystko, tylko neuron z najsilniejszą aktywacją podlega uczeniu
- WTM, grupa k najbardziej wzbudzonych neuronów podlega adaptacji
- wymuszenie ortogonalności między wektorami wag, uogólniona reguła Hebba (GHA)
- dodanie połączeń ze sprzężeniem zwrotnym wpływającymi na pobudzenia sąsiednich neuronów, np. APEX

Uogólniony algorytm hebbowski (GHA)

Jednowarstwowa sieć liniowa (Sanger, Oja, 1989)

$$y_i = \mathbf{w}_i^T \mathbf{x}$$

Reguła uczenia Sangera z czynnikiem ortogonalizującym Grama-Schmidta

$$\Delta \mathbf{w}_i = \eta y_i \left(\mathbf{x} - \sum_{k=1}^{i} y_k \mathbf{w}_k \right)$$

oznaczając

$$\hat{\mathbf{x}}_i(k) = \mathbf{x} - \sum_{k=1}^{i-1} y_k \mathbf{w}_k$$

otrzymujemy lokalną regułę Oji dla przeskalowanych wejść $\hat{\mathbf{x}}_i$

$$\Delta \mathbf{w}_i = \eta y_i \left(\hat{\mathbf{x}} - \mathbf{w}_i y_i \right)$$

- Wektory wag $\mathbf{w}_i$ kolejnych neuronów zbiegają do wartości własnych macierzy kowariancji $\mathbf{R}$ (kolejne składowe główne).
- GHA jest algorytmem on-line, nie wymaga obliczania macierzy kowariancji $\mathbf{R}$

Uczenie anty-hebbowskie

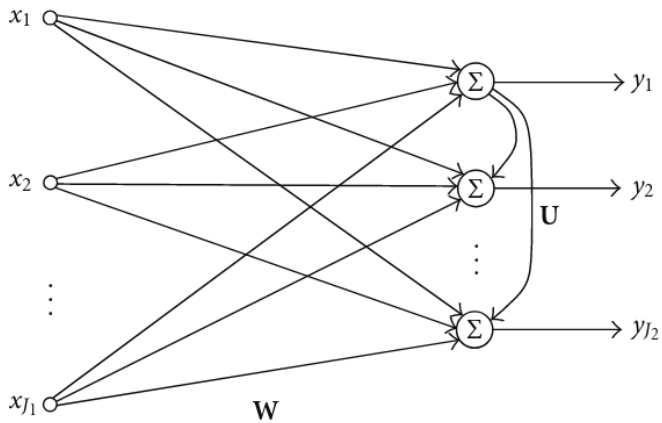
Reguła anty-Hebbowska

$$\Delta w_{ij} = -\eta x_i y_j$$

- znak minus sprawia, że trening prowadzi do znalezienia kierunku **minimalizującego** wariancję wyjścia
- uczenie jednostki liniowej prowadzi do zerowania wyjścia - trening dąży do znalezienia projekcji danych do punktu
- jeżeli nie istnieje komponent (kierunek) o zerowej wariancji to wagi zbiegają do 0

APEX

Adaptive Principal Component Extraction (Diamantaras 1990)
Dla określonych $i-1$ wektorów $\mathbf{w_1}, \ldots, \mathbf{w}_{i-1}$ realizujących kierunki PCA sieć iteracyjnie wyznacza i-tą składową główną



$$\mathbf{y} = \mathbf{W}\mathbf{x}$$

$$y_i = \mathbf{w}_i^T \mathbf{x} + \mathbf{u}_i^T \mathbf{y}$$

$\mathbf{y} = (y_1, \ldots, y_{i-1})^T$ wyjścia pierwszych $i-1$ neuronów

$\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_{i-1}]^T$ macierz wag pierwszych $i-1$ neuronów

$\mathbf{u}_i = [u_{1i}, u_{2i}, \ldots, u_{(i-1)i}]^T$ wagi połączeń wewnątrz warstwy między i-tym neuronem a pozostałymi neuronami

Rys: Qiu, et.al., Neural Network Implementations for PCA and Its Extensions

Aktualizacja wag APEX

Dla ustalonych $i-1$ składowych uczeniu podlegają wagi i-tego neuronu $\mathbf{w}_i$ i $\mathbf{u}_i$

- aktualizacja wag $\mathbf{w}_i$ za pomocą reguły Oji

$$\Delta \mathbf{w}_i = \eta(n) y_i(n) \left[\mathbf{x}(n) - y_i(n)\mathbf{w}_i\right]$$

- połączenia między-warstwowe u_{ij} uczone regułą anty-hebbowską (odwrotna reguła Oji)

$$\Delta \mathbf{u}_i = -\eta(n) y_i(n) \left[\mathbf{y}(n) + y_i(n)\mathbf{u}_i\right]$$

- wagi $\mathbf{u}$ odpowiadają za wydzielenie pierwszych $i-1$ składowych z wag i-tego neuronu
- uczenie anty-hebbowskie wymusza ortogonalizację $\mathbf{w}_i$ względem poprzednich składowych

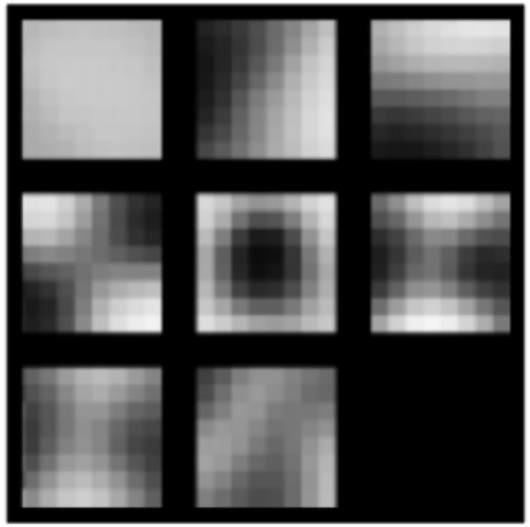
Zastosowanie PCA

- Redukcja wymiarowości
- Wizualizacja danych nD $\rightarrow$ 2D, 3D
- Usuwanie szumu
- Usuwanie korelacji z danych
- Kompresja sygnału, np. kompresja obrazów
- Dane po transformacji PCA często używane do treningu algorytmów uczenia maszynowego

Kompresja danych za pomocą PCA

Wektory treningowe to ramki obrazu (8x8 = 64 współrzędne)
Przykład: obraz 512x512 z 8 bitami na piksel





ordering of the neurons:

neuron 0	neuron 1	neuron 2
neuron 3	neuron 4	neuron 5
neuron 6	neuron 7	(unused)

Neurony w kolejności zawierającej najwięcej informacji potrzebnej do rekonstrukcji

- neuron 0: średni poziom szarości
- neuron 1 i 2: gradient obrazu
- neurony 3-5: pochodna drugiego rzędu obrazu

Rys: Krose, Smagta, An Introduction to Neural Networks, 1996

Przykład: Kompresja obrazu za pomocą APEX

a) oryginał
b) 4 PC
c) 8 PC
d) 12 PC



Rys: Hu Y. H., Hwang J., Handbook of Neural Network Signal Processing

Sieci Heraulta-Juttena

- liniowe sieci samoorganizujące oparte o regułę uczenia Hebba uogólnioną do funkcji nieliniowych
- pierwotnie stosowane do ślepej separacji sygnałów, używane również do analizy składników niezależnych (ICA), składników głównych (PCA), dekonwolucji
- struktura sieci rekurencyjna lub jednokierunkowa
- przetwarzanie sygnałów w czasie rzeczywistym (realizacje sprzętowe)
- funkcje nieliniowe odgrywają istotną rolę w algorytmie uczenia ale nie wpływają na strukturę sieci

Separacja sygnałów

Dla n niezależnych sygnałów źródłowych $s_i(t)$ dostępne są tylko sygnały $x_i(t)$ będące liniową superpozycją sygnałów oryginalnych $s_i(t)$

$$x_i(t) = \sum_{j=1}^{n} a_{ij} s_j(t)$$

Macierz mieszająca

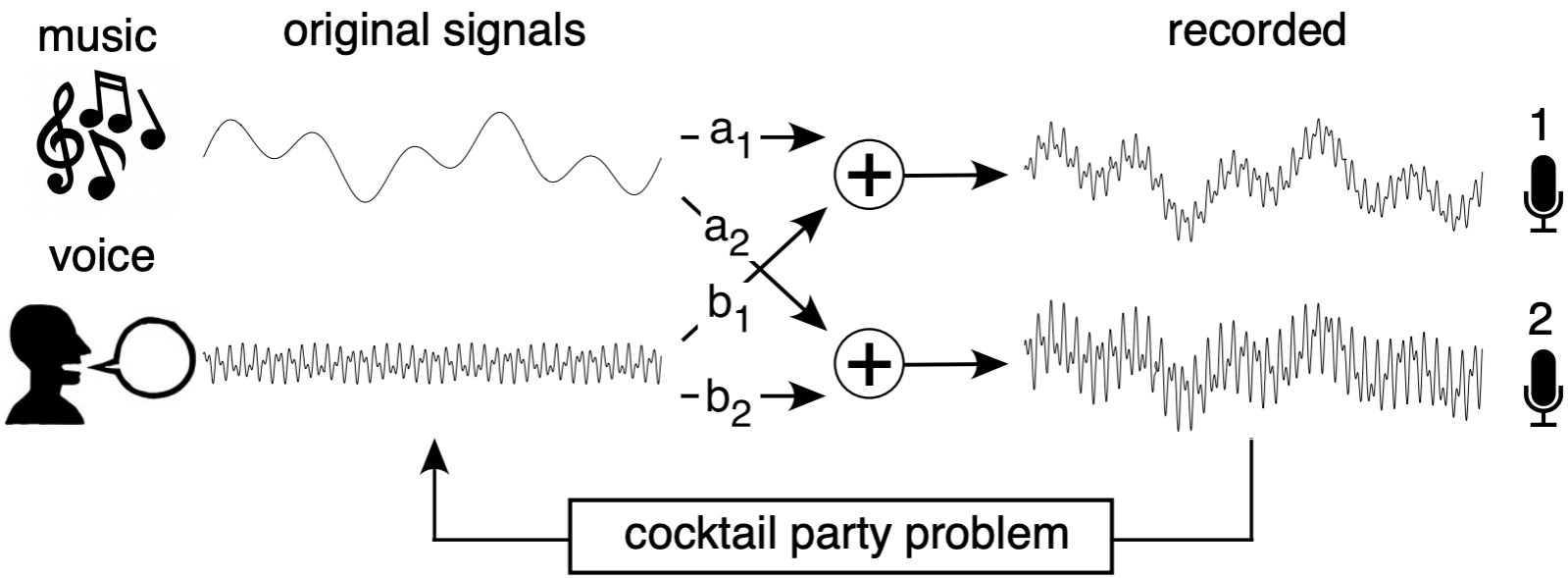
$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

W notacji macierzowej

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t)$$

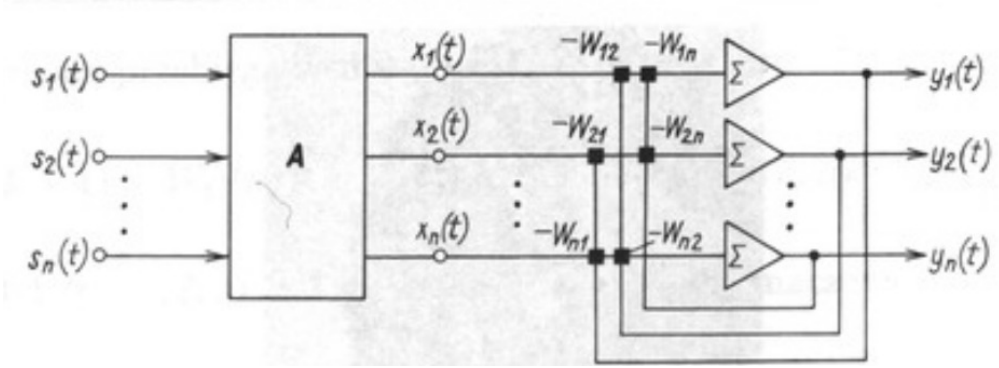
Ślepa separacja źródeł (BSS): zadanie odzyskania oryginalnych komponentów sygnału $\mathbf{s}(t)$ przy nieznanych $\mathbf{A}$ jak i $\mathbf{s}(t)$.
Przykład: *cocktail party problem*

Separacja sygnałów



☞ *J. Shlens, A Tutorial on Independent Component Analysis*

Sieć rekurencyjna Heraulta-Juttena



Działanie realizowane przez sieć rekurencyjną

$$\boldsymbol{x}(t) = \boldsymbol{A}\boldsymbol{s}(t)$$
$$\boldsymbol{y}(t) = \boldsymbol{x}(t) - \mathbf{W}\boldsymbol{y}(t)$$

gdzie macierz wag (w oryginalnej wersji sieci)

$$\boldsymbol{W} = \begin{bmatrix} 0 & w_{12} & \cdots & w_{1n} \\ w_{21} & 0 & \cdots & w_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ w_{n1} & w_{n2} & \cdots & 0 \end{bmatrix}$$

Szukamy rozwiązania

$$\mathbf{y}(t) = (\mathbf{1} + \mathbf{W})^{-1}\mathbf{x}(t)$$

które odtwarza sygnał pierwotny $\mathbf{s}(t)$.
Warunki:

- odtwarzanie z dokładnością do skali

  $$\boldsymbol{y}(t) = \boldsymbol{D}\boldsymbol{s}(t)$$

  gdzie $\mathbf{D}$ jest macierzą diagonalną
- odtwarzanie z dokładnością do kolejności sygnałów

  $$\mathbf{y}(t) = \mathbf{P}\mathbf{s}(t)$$

  gdzie $\mathbf{P}$ jest macierzą permutacji

Dla małej liczby kanałów ($n = 2$) istnieje rozwiązanie analityczne.
Dla większej liczby kanałów stosuje się metodę adaptacyjną.

Reguła Heraulta-Juttena

Przy założeniu statystycznej niezależności sygnałów można wyprowadzić regułę uczenia tożsamą z regułą Hebba

$$\frac{dW_{ij}}{dt} = \eta y_i(t) y_j(t)$$

Zbieżność procesu uczącego wymaga aby w stanie ustalonym $E[\frac{dW_{ij}}{dt}] = 0$ stąd sygnały $y_i(t)$ i $y_j(t)$ nie mogą być skorelowane $E[(y_i - E[y_i])(y_j - E[y_j])] = 0$

Reguła Heraulta-Juttena

$$\frac{dW_{ij}}{dt} = \eta(t) f\left(y_i(t)\right) g\left(y_j(t)\right)$$

$f()$ i $g()$ to funkcje nieparzyste, które przy założeniu statystycznej niezależności sygnałów zapewniają $E[f\left(y_i(t)\right) g\left(y_j(t)\right)] = 0$, co jest warunkiem zbieżności algorytmu uczenia.

Przykłady: $f(x) = x^3, x^5$, $g(x) = \tanh(x), \arctan(x), x, \mathrm{sgn}(x)$

Zmodyfikowany algorytm Cichockiego

Reguła Heraulta-Juttena skuteczna tylko dla sygnałów $s_i(t)$ o niewielkim zróżnicowaniu amplitudy (dla względnej różnicy amplitud $< 10^2$). Sygnały o małej amplitudzie nie są separowane.

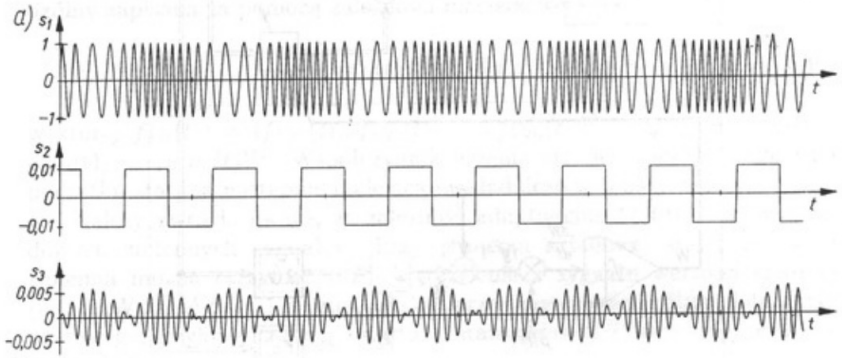
Modyfikacja Cichockiego

$$\frac{d\mathbf{W}}{dt} = \eta(t)\left[\boldsymbol{f}(\boldsymbol{y}(t))[\boldsymbol{g}(\boldsymbol{y}(t))]^T - \mathbf{1}\right]$$

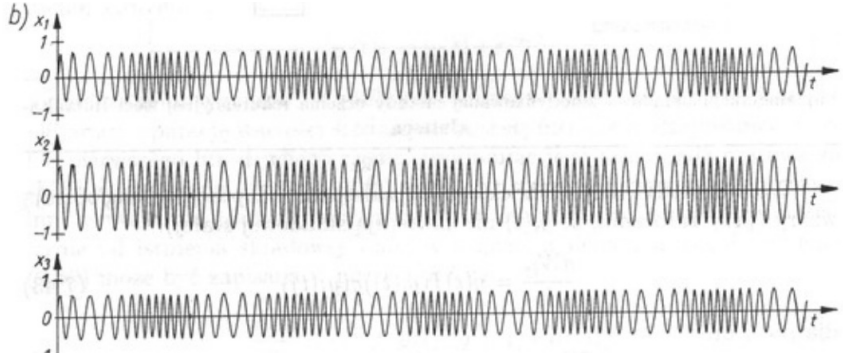
gdzie $\boldsymbol{y}(t) = (\mathbf{1} + \boldsymbol{W})^{-1}\boldsymbol{x}(t)$

- wprowadza sprzężenia zwrotne własne neuronu $w_{ii} \neq 0$ powodujące samonormalizację sygnałów wyjściowych $y_i(t)$
- w stanie ustalonym $E[\frac{d\mathbf{W}}{dt}] = 0$ stąd $E[f(y_i)g(y_j)] = \delta_{ij}$, co jest warunkiem zbieżności algorytmu uczenia
- separacja możliwa nawet dla sygnałów o różnicy amplitud rzędu 10^{10}

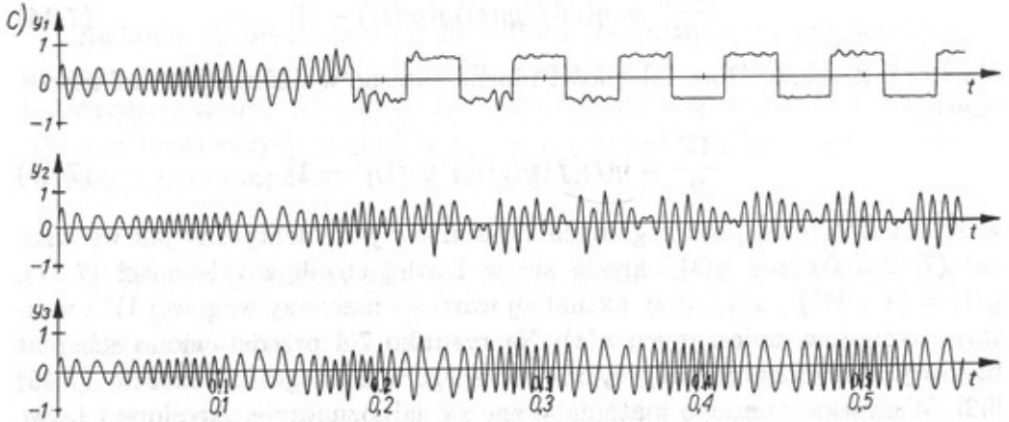
Sygnały oryginalne $s_i(t)$



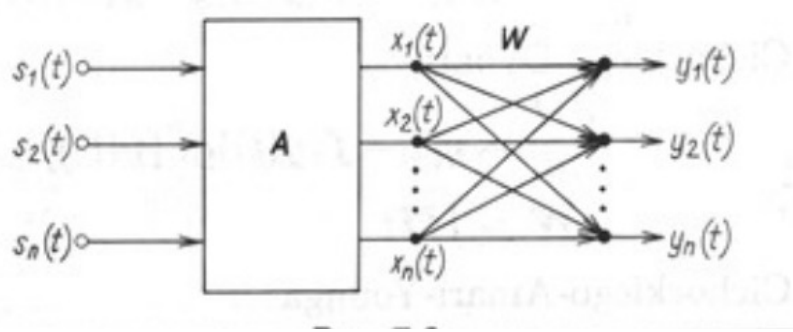
Sygnały zmieszane $x_i(t)$



Proces wydzielania sygnałów $y_i(t)$ odtwarzających sygnały źródłowe $s_i(t)$



Sieć jednokierunkowa



$$\mathbf{y} = \mathbf{W}\mathbf{x}$$

macierz $\mathbf{W} \in \mathbb{R}^{n \times n}$ jest macierzą pełną.

Sieć jednokierunkowa jest równoważna sieci ze sprzężeniami zwrotnymi gdy $\mathbf{W} = (\hat{\mathbf{W}} + \mathbf{1})^{-1}$ w której $\hat{\mathbf{W}}$ to macierz wag sieci rekurencyjnej

Zmodyfikowana reguła adaptacyjna Cichockiego

$$\frac{d\mathbf{W}}{dt} = \eta(t)\mathbf{W}\left[\mathbf{1} - \boldsymbol{f}(\mathbf{y}(t))[\boldsymbol{g}(\mathbf{y}(t))]^T\right]\boldsymbol{W}$$