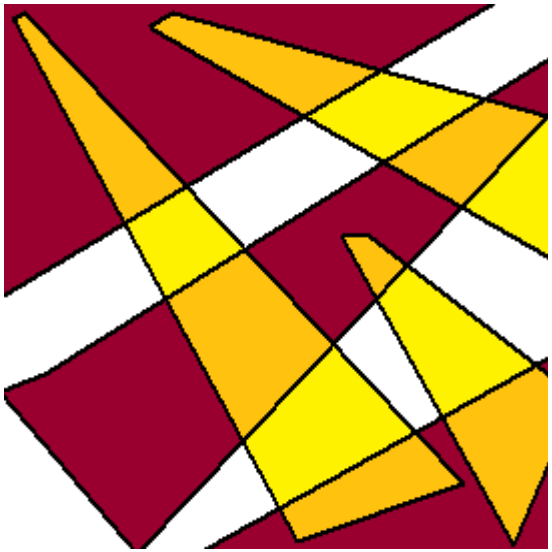
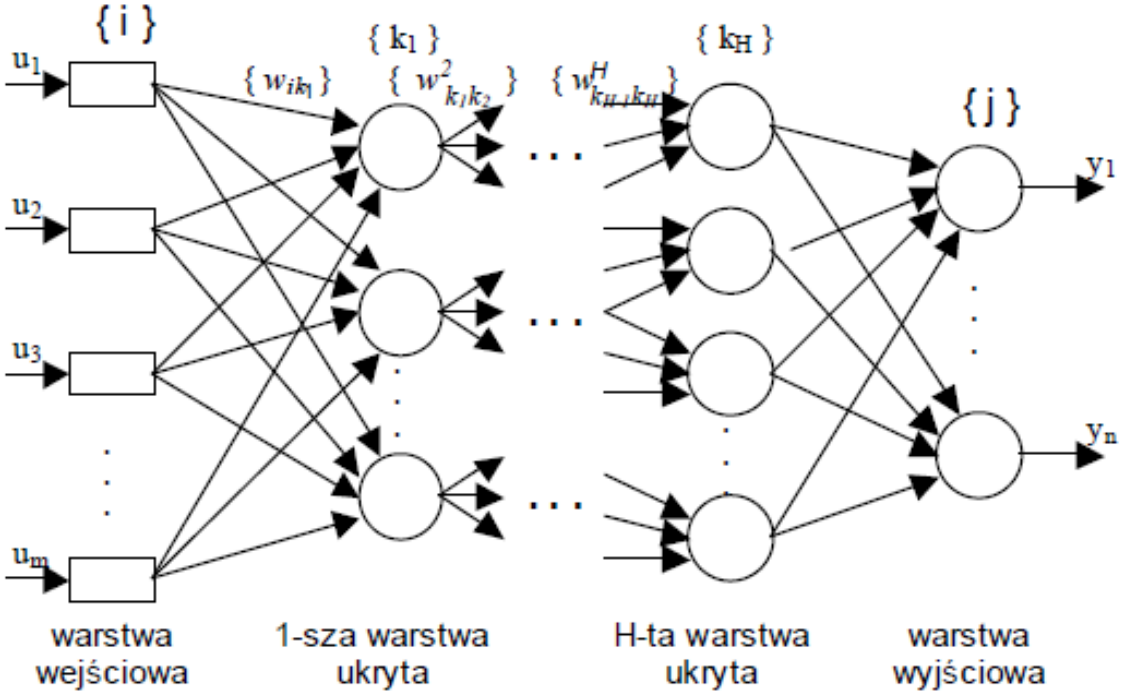
Uczenie sieci typu MLP

Przypomnienie – budowa sieci typu MLP





i – numer elementu warstwy wejściowej (i=1,2,..,m),

j – numer elementu warstwy wyjściowej (j=1,2,..,n),

h – numery kolejnych warstw ukrytych (h=1,2,..,H),

$w^h_{k_{h-1},k_h}$ – waga połączenia pomiędzy elementami k_{h-1}-tym, a k_h-tym odpowiednio w warstwach (*h-1*)-szej i *h*-tej.

Przypomnienie budowy neuronu

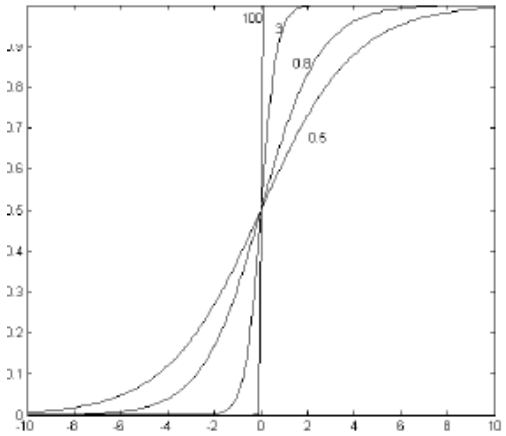
- Neuron ze skokową funkcją aktywacji jest zły!!!
- Powszechnie stosuje -> modele z sigmoidalną funkcją aktywacji
- β - współczynnik nastromienia. Im większy tym bardziej skokowa funkcja aktywacji

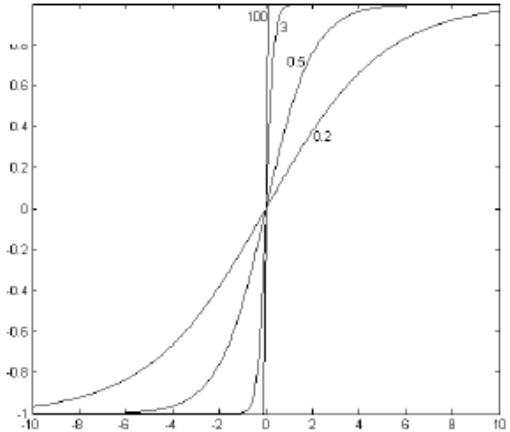
$$z = \sum_{i}^{M} w_i x_i + w_0$$

$$f_1(z) = \frac{1}{1+\exp(-\beta z)}$$

$$f_2(z) = tgh(\beta z)$$

$$f_3(z) = 2 f_1(z) - 1$$





Różniczkowalność funkcji sigmoidalnej

- Pochodne funkcji aktywacji

$$\frac{df_1(x)}{dx} = \beta f_1(x)\left(1 - f_1(x)\right)$$

$$\frac{df_2(x)}{dx} = \beta\left(1 - f_2^2(x)\right)$$

$$\frac{df_3(x)}{dx} = 2\beta f_1(x)\left(1 - f_1(x)\right)$$

Trochę o uczeniu

Uczenie sieci MLP to optymalizacja wartości wag w celu minimalizacji błędu popełnianego przez sieć.

Funkcja celu - kryterium, według którego można oceniać dokonany wybór rozwiązania najlepszego spośród dopuszczalnych rozwiązań (wariantów), czyli jak dany system w procesie swego działania zbliża się do osiągnięcia wyznaczonego celu. Działając zgodnie z zasadami ekonomii (zasadą oszczędności i zasadą wydajności) dąży się każdorazowo do maksymalizacji lub minimalizacji funkcji celu w zależności od postawionego celu działania. Funkcja celu określa więc w sposób formalny zależność między celem systemu (firmy) a środkami służącymi do jego realizacji.

wg. portalwiedzy.onet.pl

Jak zdefiniować funkcję celu?

Stosując metody gradientowe funkcja celu musi spełniać warunek różniczkowalności!!!

Funkcja celu

- Błąd średniokwadratowy dla sieci o M wyjściach

$$E = \frac{1}{2}\sum_{i=1}^{M}\left(y_i - d_i\right)^2$$

y – rzeczywista wartość i-tego wyjścia sieci

d – wyliczona wartość i-tego wyjścia sieci

Całkowita wartość funkcji celu po prezentacji *n* przypadków uczących ma postać

$$E = \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{M}\left(y_i\left(\mathbf{x}_j\right) - d_i\left(\mathbf{x}_j\right)\right)^2$$

Inne odmiany funkcji celu

- Funkcja z normą L_1

$$E = \frac{1}{2}\sum_{i=1}^{M}|y_i - d_i|$$

Minimalizacja wszystkich błędów równomiernie

- Funkcja z normą wyższych rzędów

$$E = \frac{1}{2}\sum_{i=1}^{M}(y_i - d_i)^{2K}$$

Minimalizacja największych błędów (małe błędy stają się nie istotne)

Inne odmiany funkcji celu. CD.

- Kombinacja dwóch powyższych (Karayiannis):

$$E = \frac{1}{2}\lambda\sum_{i=1}^{M}\left(y_i - d_i\right)^2 + \left(1-\lambda\right)\sum_{i=1}^{M}\phi\left(y_i - d_i\right)$$

- Dla λ=1 -> minimalizacja błędu średniokwadratowego
- Dla λ=0 -> minimalizacja błędu zdefiniowanego przez funkcję ϕ
- W praktyce uczymy zaczynając od λ=1 i stopniowo w trakcie uczenia zmniejszamy λ do 0

$$\phi(a) = \frac{1}{\beta}\ln\left(\cosh\left(\beta a\right)\right)$$

Dla dużych β zachodzi $\phi(a)=|a|$

Problem uczenia sieci MLP

- Jak dobrać odpowiednie wartości wag?
- Jak wyznaczyć błąd popełniany przez warstwy ukryte?
- Jak więc uczyć warstwy ukryte by minimalizować ów błąd?
- Jak określić kierunek zmian wartości wag, czy + czy -, o jaką wartość zmieniać wagi?

Metody optymalizacji

- Stochastyczne
  - Monte carlo
  - Algorytmy genetyczne
  - Algorytmy ewolucyjne
- Gradientowe
  - Największego spadku (reguła delta)

$$W(k+1) = W(k) + \Delta W$$

$$\Delta W = \eta p(W)$$

η- współczynnik ucenia
p(W) – kierunek i wartość zmian wektora W

Algorytm wstecznej propagacji błędu

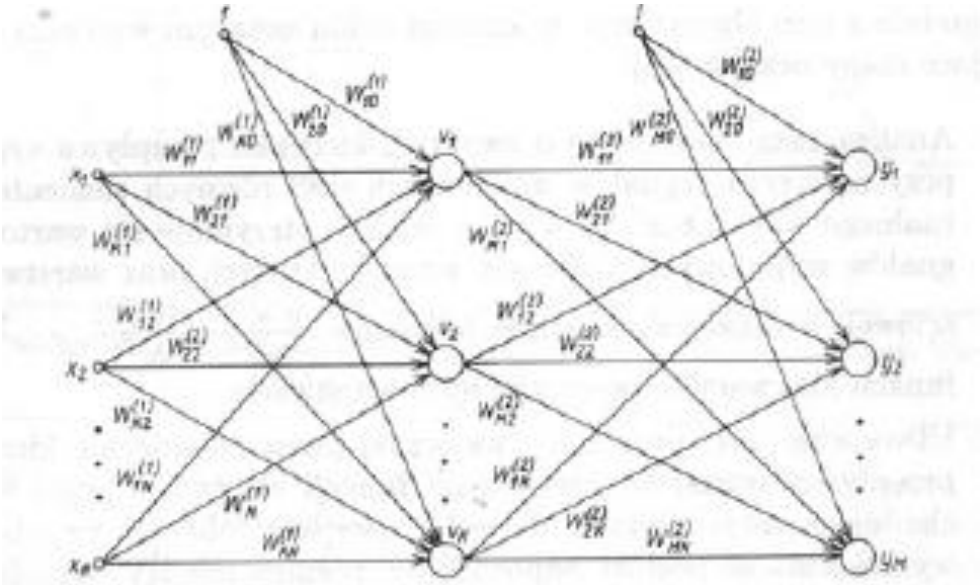
1. Analiza sieci neuronowej o zwykłym kierunku przepływu sygnałów. Podanie na wejście danego wektora x_i i wyznaczenie odpowiedzi każdego z nauronów dla każdej z warstw (odpowiednio d_i dla wyjściowej i s_i dla ukrytej).
2. Stworzenie sieci propagacji wstecznej zamieniając wejścia sieci na jej wyjścia oraz zamieniając funkcje aktywacji neuronu na pochodne oryginalnych funkcji aktywacji. Na wejście sieci należy podać różnicę sygnałów wyjściowego i oczekiwanego (y_i-d_i)
3. Uaktualnienie wag odbywa się na podstawie wyników uzyskanych w punkcie 1 i 2 wg. zależności
4. Opisany proces powatarzaj aż błąd nie spadnie poniżej wartości progowej $\varepsilon < threshold$

Trochę wzorów

Funkcja celu uwzględniając dwie warstwy ukryte:

$$E=\frac{1}{2}\sum_{k=1}^{M}\left[f\left(\sum_{i=0}^{K}W_{ki}^{(2)}v_i\right)-d_k\right]^2=\frac{1}{2}\sum_{k=1}^{M}\left[f\left(\sum_{i=0}^{K}W_{ki}^{(2)}f\left(\sum_{j=0}^{N}W_{ij}^{(1)}x_j\right)\right)-d_k\right]^2$$

v_i – wyjścia warstwy ukrytej, co dalej możemy zapisać jako
Uwaga sumowanie po K od 0 bo zakładamy że nasz wektor ma postać $\mathbf{x}=[1\ x_1\ x_2\ \ldots\ x_N]^T$ i odpowiednio $\mathbf{v}=[1\ v_1\ v_2\ \ldots\ v_K]^T$
Uwaga N-wejść, K- neuronów ukrytych i M wyjść z sieci



Źródło rysunku i wzorów: Ossowski, Sieci neuronowe w ujęciu algorytmicznym", WNT

Wzory cd.

- Zmaina wag warstwy wy.

$$\frac{\partial E}{\partial W_{ij}^{(2)}} = (y_i - d_i)\frac{df(u_i^{(2)})}{du_i^{(2)}} v_j$$

  - Gdzie $u_i^{(2)} = \sum_{j=0}^{K} W_{ij}^{(2)} v_j.$ przyjmując: $\delta_i^{(2)} = (y_i - d_i)\frac{df\left(u_i^{(2)}\right)}{du_i^{(2)}}$

- Ostatecznie zmianę wag dla wa-wy 2 możemy zapisać jako:

$$\frac{\partial E}{\partial W_{ij}^{(2)}} = \delta_i^{(2)} v_j$$

- Dla warstwy ukrytej (nr 1) zależność ta przyjmuje postać:

$$\frac{\partial E}{\partial W_{ij}^{(1)}} = \sum_{k=1}^{M} (y_k - d_k)\frac{dy_k}{dv_i}\frac{dv_i}{dW_{ij}^{(1)}}$$

Gdzie zmiana wag wynikająca z wa-wy wyj (2), zmiana wag z wa-wy ukrytej(1)

Źródło rysunku i wzorów: Ossowski, Sieci neuronowe w ujęciu algorytmicznym", WNT

Wzory cd..

- Uwzględniając poszczególne składniki otrzymujemy

$$\frac{\partial E}{\partial W_{ij}^{(1)}} = \sum_{k=1}^{M} (y_k - d_k) \frac{df(u_k^{(2)})}{du_k^{(2)}} W_{ki}^{(2)} \frac{df(u_i^{(1)})}{du_i^{(1)}} x_j$$

- Co dla poniższych

$$\delta_k^{(2)} = (y_k - d_k) \frac{df(u_k^{(2)})}{du_k^{(2)}}$$

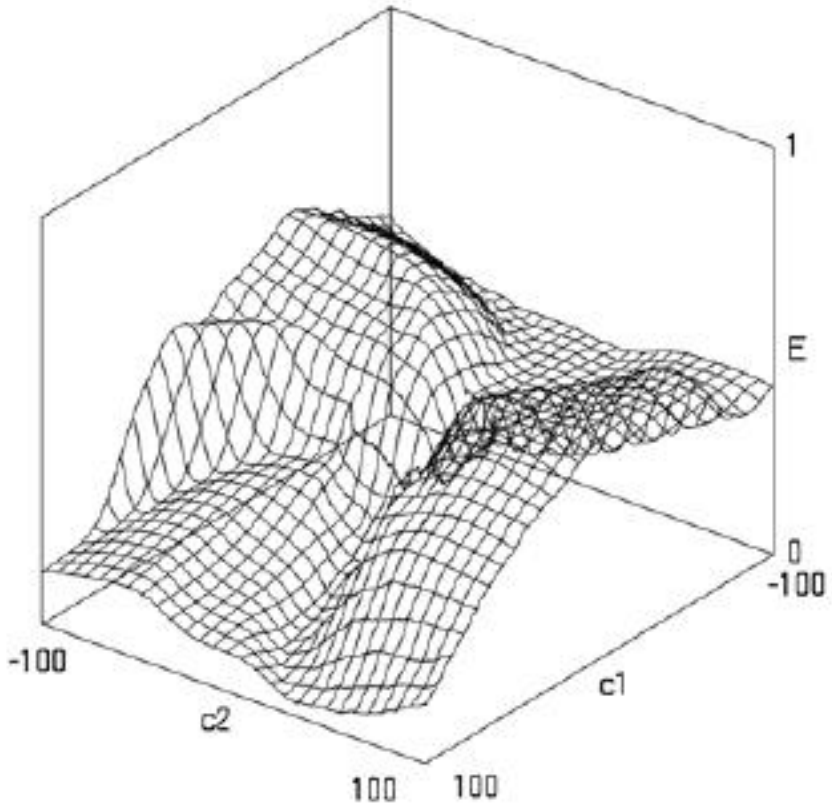
$$\delta_i^{(1)} = \sum_{k=1}^{M} \delta_k^{(2)} W_{ki}^{(2)} \frac{df(u_i^{(1)})}{du_i^{(1)}}$$

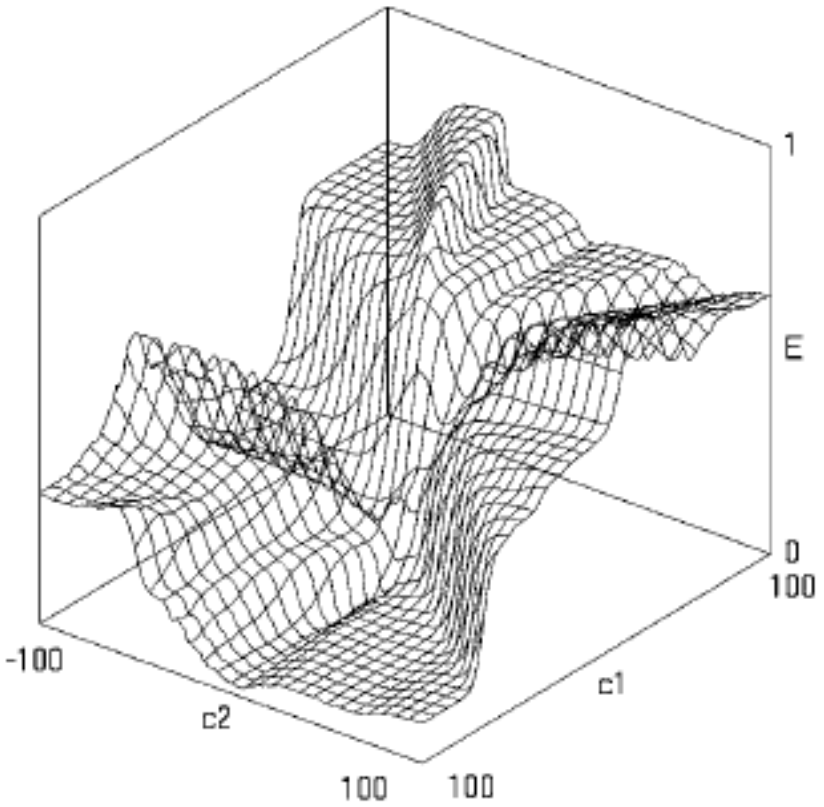
- Pozwala zapisać pochodną funkcji kosztu w warstwie ukrytej ja $\frac{\partial E}{\partial W_{ij}^{(1)}} = \delta_i^{(1)} x_j$

- Ostatecznie zmian $\Delta W = -\eta \nabla E(W)$ owana jest jako: współczynnik uczenia

Źródło rysunku i wzorów: Ossowski, Sieci neuronowe w ujęciu algorytmicznym", WNT

Problem minimów lokalnych





Rys. M. Kordos „Search-based Algorithms for Multilayer Perceptrons” PhD

Różne wersje algorytmów – algorytmy gradientowe

- W sąsiedztwie najbliższego rozwiązania rozwijają funkcję celu E(W) w szereg Taylora (naiczęściej do pierwszych 3 składników)

$$E(W+p) = E(W) + [g(W)]^T p + \frac{1}{2} p^T H(W) p + \cdots$$

Gdzie:

$$g(W) = \nabla E = \left[\frac{\partial E}{\partial W_1}, \frac{\partial E}{\partial W_2}, \cdots, \frac{\partial E}{\partial W_n}\right]^T$$

Oraz pocł

$$H(W) = \begin{bmatrix} \frac{\partial E}{\partial W_1 \partial W_1} & \cdots & \frac{\partial E}{\partial W_n \partial W_1} \\ \vdots & & \vdots \\ \frac{\partial E}{\partial W_1 \partial W_n} & \cdots & \frac{\partial E}{\partial W_n \partial W_n} \end{bmatrix}$$

macierz drugich

p – wektor kierunkowy liczenia pochodnych zależny od W

Optymalne rozwiązanie gdy $g(W_k)=0$ i $H(W_k)$ jest dodatnio określona (v :y H są > 0)

lub

$$E(W_{k-1}) - E(W_k) \leqslant \tau (1 + |E(W_k)|)$$
$$\|W_{k-1} - W_k\| \leqslant \sqrt{\tau} (1 + \|W_k\|)$$
$$\|g(W_k)\| \leqslant \sqrt[3]{\tau} (1 + |E(W_k)|)$$

Źródło rysunku i wzorów: Ossowski, Sieci neuronowe w ujęciu algorytmicznym", WNT

Inne metody optymalizacji

- Algorytm największego spadku (rozwinięcie tylko do pierwszej pochodnej)
- Algorytm zmiennej metryki (wykorzystanie kwadratowego przybliżenia funkcji E(W) w sąsiedztwie W_k)
- Algorytm Levenberga-Marquardta (najlepsza, zastąpienie H(W) przez aproksymację G(W) z reguloaryzacją)

Dobór współczynnika uczenia η

- Stały współczynnik uczenia

  W praktyce jeśli jest stosowany to jest on wyznaczany niezależnie dla każdej warstwy (n_i-liczba wejść i-tego neuronu)

$$\eta \leqslant \min\left(\frac{1}{n_i}\right)$$

- Adaptacyjny dobór wsp. Uczenia

  Przyjmując jako błąd uczenia $\varepsilon = \sqrt{\sum_{j=1}^{M}(y_j - d_j)^2}$ oraz $\eta_{(i+1)}$, η_i – współczynniki uczenia w iterazji *i* oraz *i+1* oraz odpowiednio błąd uczenia $\varepsilon_{(i+1)}$, ε_i , k_w – dopuszczalny wzrost wartości wsp η

  if $\varepsilon_i > k_w \varepsilon_{i-1}$ then $\eta_{i+1} = \eta_i \rho_d$ else $\eta_{i+1} = \eta_i \rho_i$

Gdzie ρ_d<1 (np. 0.7) oraz ρ_i>1 (np. 1.05)

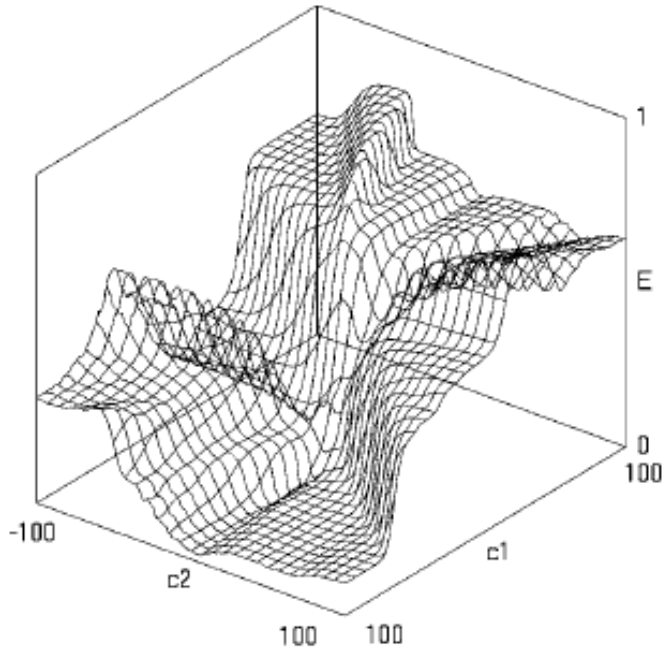
Źródło rysunku i wzorów: Ossowski, Sieci neuronowe w ujęciu algorytmicznym", WNT

Dobór współczynnika uczenia η (inne metody)

- Dobór wsp. uczania przez minimalizację kierunkową
- Reguła delta-bar-delta doboru wsp. uczenia

Inicjalizacja wag

Inicjalizacja wag wpływa na rozwiązanie – zależy w którym miejscu funkcji powierzchni funkcji celu zaczniemy optymalizację



- Losowa
- PCA
- W praktyce – zastosowanie metody wielostartu

Rys. M. Kordos „Search-based Algorithms for Multilayer Perceptrons” PhD

Metody optymalizacji globalnej

- Dotychczasowe metody mają charakter lokalny (optymalizujemy w obrębie najbliższych rozwiązań)
- Metody globalne – patrzą na problem całościowy i całościowo optymalizują sieć.
- Optymalizacja globalna to metody optymalizacji stochastycznej – symulowane wyżarzania, algorytmy genetyczne i ewolucyjne

Przykład – symulowane wyżarzanie

1. Start procesu z rozwiązania początkowego W, temperatura $T=T_{max}$
2. Dopóki T>0 wykonaj L razy
   - Wybierz nowe rozwiązanie W' w pobliżu W
   - Oblicz funkcję celu $\Delta=E(W')-E(W)$
   - Jeżeli $\Delta <= 0$ to W=W'
     W przeciwnym przypadku ($\Delta>0$)
     jeżeli $e^{-\Delta/T}>R$ to W=W' (gdzie R to liczba losowa z przedziału [0,1])
3. Zredukuj temperaturę T=rT (*r* –współczynnik redukcji z przedziału [0,1])
4. Po redukcji temperatury T do 0 ucz metodą gradientową