Sztuczne sieci neuronowe

Krzysztof A. Cyran
POLITECHNIKA ŚLĄSKA
Instytut Informatyki, p. 311

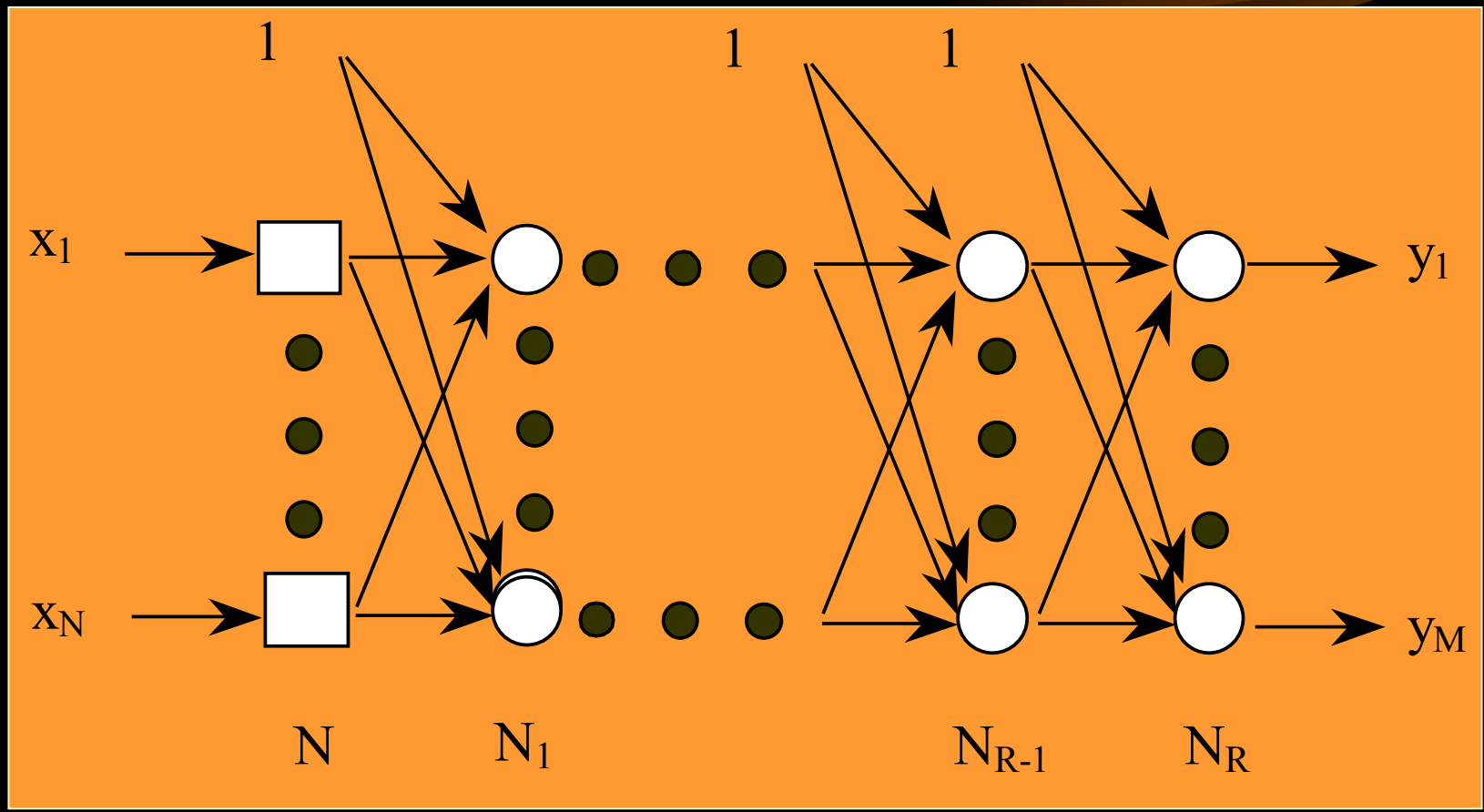
Wykład 3

PLAN:

Algorytm propagacji wstecznej błędu:

- **Oznaczenia**
- **Uczenie jako minimalizacja błędu średniokwadratowego**
- **Pojęcie uogólnionej delty**
- **Cechy algorytmu backpropagation**
- **Podsumowanie algorytmu klasycznego**
- **Inercyjne modyfikacje**

Perceptron R-warstwowy



Back propagation: oznaczenia

- N - ilość wejść
- M - ilość wyjść
- R - ilość warstw
- L - ilość wzorców uczących
- N_i - ilość neuronów warstwy i-tej ($N_R = M$)
- $x^{(j)}$ - j-ty wektor wejściowy
- $y^{(j)}$ - j-ty wektor wyjściowy
- $d^{(j)}$ - j-ty oczekiwany wektor wyjściowy
- $O_j^{(l)R}$ - sygnał na wyjściu j-tego neuronu R-tej warstwy ($R = 1..N_R$) po podaniu l-tego wzorca ($l = 1..L$)
- E - całkowity błąd sieci po podaniu wszystkich wzorców uczących
- $E^{(l)}$ - całkowity błąd sieci po podaniu l-tego wzorca uczącego
- $E_m^{(l)}$ - błąd po podaniu l-tego wzorca uczącego dla m-tego wyjścia
- η - współczynnik uczenia

Backpropagation

- Dany jest ciąg uczący postaci:

$\{(\mathbf{x}^{(1)}, \mathbf{d}^{(1)}), ..., (\mathbf{x}^{(L)}, \mathbf{d}^{(L)})\}$

Wówczas:

$$E = \sum_{l=1}^{L} E^{(l)}$$

Gdzie:

$$E^{(l)} = \sum_{m=1}^{M} E_m^{(l)} = \frac{1}{2} \sum_{m=1}^{M} (d_m^{(l)} - y_m^{(l)})^2$$

Backpropagation (cd)

- Uczenie sieci jest minimalizacją błędu E, w której zmiennymi niezależnymi są wagi w_{ij}
- Ponieważ nawet najprostsze sieci mają bardzo dużo wag jest to proces minimalizacji pola skalarnego nad przestrzenią wektorową o setkach (tysiącach) wymiarów

Backpropagation (cd)

- Do minimalizacji błędu E wykorzystujemy gradientową metodę największego spadku
- Zgodnie z nią: $$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = -\eta \sum_{l=1}^{L} \frac{\partial E^{(l)}}{\partial w_{ij}}$$
- Z powyższego wzoru wynika, że zmiany wag dokonujemy po podaniu całego zestawu wzorców. Często jednak, dla uproszczenia algorytmu, rezygnujemy z tego warunku zmieniając wagi po podaniu każdego wzorca, odpowiednio zmniejszając η. Zatem:

$$\Delta w_{ij}^{(l)} = -\eta \frac{\partial E^{(l)}}{\partial w_{ij}}$$

Backpropagation (cd)

- Ponieważ błąd generowany przez sieć nie zależy bezpośrednio od wag lecz od wyjść poszczególnych neuronów, te zaś dopiero zależą od wag, zatem:

$$\Delta w_{ij}^{(l)} = -\eta \frac{\partial E^{(l)}}{\partial w_{ij}} = -\eta \frac{\partial E^{(l)}}{\partial O_i^{(l)}} \frac{\partial O_i^{(l)}}{\partial w_{ij}} = -\eta \frac{\partial E^{(l)}}{\partial O_i^{(l)}} \frac{\partial O_i^{(l)}}{\partial n_i^{(l)}} \frac{\partial n_i^{(l)}}{\partial w_{ij}}$$

Backpropagation (cd)

- Ponieważ:

$$n_i^r = \sum w_{ij}^r o_j^{r-1}, \quad O_i^r = f\left(n_i^r\right)$$

Zatem:

$$\frac{\partial n_i^{(l)}}{\partial w_{ij}} = O_j^{(l)}$$

oraz:

$$\Delta w_{ij}^{(l)} = -\eta \frac{\partial E^{(l)}}{\partial O_i^{(l)}} \frac{\partial O_i^{(l)}}{\partial n_i^{(l)}} O_j^{(l)}$$

Backpropagation (cd)

Zdefiniujmy tzw. uogólnioną deltę jako:

$$\delta_i^{(l)} = -\frac{\partial E^{(l)}}{\partial O_i^{(l)}} \frac{\partial O_i^{(l)}}{\partial n_i^{(l)}}$$

Wówczas:

$$\Delta w_{ij}^{(l)} = \eta \delta_i^{(l)} O_j^{(l)}$$

Backpropagation (cd)

- Znaczenie uogólnionej delty danego neuronu uzależnione jest od położenia rozważanego neuronu
- Dla warstwy wyjściowej: $O_i^{(l)R} = y_i^{(l)}$

Zatem:

$$\frac{\partial E^{(l)}}{\partial O_i^{(l)R}} = \frac{\partial\left\{\sum_{m=1}^{M}\frac{1}{2}(d_m^{(l)} - y_m^{(l)})^2\right\}}{\partial y_i^{(l)}} = \frac{1}{2}\sum_{m=1}^{M}\frac{\partial\left\{(d_m^{(l)} - y_m^{(l)})^2\right\}}{\partial y_i^{(l)}} =$$

$$= \frac{1}{2}\frac{\partial\left\{(d_i^{(l)} - y_i^{(l)})^2\right\}}{\partial y_i^{(l)}} = -(d_i^{(l)} - y_i^{(l)})$$

Backpropagation (cd)

- Po równoczesnym wstawieniu wzoru na pochodną funkcji sigmoidalnej uzyskujemy dla warstwy wyjściowej *R*:

$$\delta_i^{(l)R} = \beta(d_i^{(l)} - y_i^{(l)})O_i^{(l)R}(1-O_i^{(l)R}) =$$
$$= \beta(d_i^{(l)} - y_i^{(l)})y_i^{(l)}(1-y_i^{(l)})$$

Backpropagation (cd)

- Dla warstw ukrytych obliczenie uogólnionych delt jest trudniejsze. Oblicza się je zaczynając od warstwy przedostatniej przechodząc do warstwy pierwszej:
- Na początek załóżmy że neuron jest w ostatniej warstwie ukrytej tj. warstwie R-1.
- Wówczas:

$$\frac{\partial E^{(l)}}{\partial O_i^{(l)R-1}} = \frac{\partial \sum_{k=1}^{M} E_k^{(l)}}{\partial O_i^{(l)R-1}} = \sum_{k=1}^{M} \frac{\partial E_k^{(l)}}{\partial O_i^{(l)R-1}} = \sum_{k=1}^{M} \frac{\partial E_k^{(l)}}{\partial n_k^{(l)R}} \frac{\partial n_k^{(l)R}}{\partial O_i^{(l)R-1}}$$

Backpropagation (cd)

- Ponieważ:

$$n_k^{(l)R} = \sum_{j=1}^{N_{R-1}} w_{kj} O_j^{(l)R-1}$$

więc:

$$\frac{\partial n_k^{(l)R}}{\partial O_i^{(l)R-1}} = w_{ki}$$

Backpropagation (cd)

- Dlatego:

$$\frac{\partial E^{(l)}}{\partial O_i^{(l)R-1}} = \sum_{k=1}^{M} \frac{\partial E_k^{(l)}}{\partial n_k^{(l)R}} w_{ki} = \sum_{k=1}^{M} \frac{\partial E_k^{(l)}}{\partial O_k^{(l)R}} \frac{\partial O_k^{(l)R}}{\partial n_k^{(l)R}} w_{ki} =$$

$$= \sum_{k=1}^{M} \frac{\partial E^{(l)}}{\partial O_k^{(l)R}} \frac{\partial O_k^{(l)R}}{\partial n_k^{(l)R}} w_{ki} = -\sum_{k=1}^{M} \delta_k^{(l)R} w_{ki}$$

i stąd nazwa propagacji wstecznej błędów.

Backpropagation (cd)

- Ostatecznie dla warstwy *R*-1:

$$\delta_i^{(l)R-1} = \beta O_i^{(l)R-1}(1 - O_i^{(l)R-1}) \sum_{k=1}^{M} \delta_k^{(l)R} w_{ki}$$

- A dla dowolnej warstwy *r*, gdzie *r* = 1,..., *R-1*

$$\delta_i^{(l)r} = \beta O_i^{(l)r}(1 - O_i^{(l)r}) \sum_{k=1}^{N_{r+1}} \delta_k^{(l)r+1} w_{ki}$$

Backpropagation (podsumowanie)

- Obliczamy uogólnione delty dla neuronów wyjściowych

$$\delta_i^{(l)R} = \beta(d_i^{(l)} - y_i^{(l)}) y_i^{(l)} (1 - y_i^{(l)})$$

- Obliczamy uogólnione delty dla kolejnych (licząc od ostatniej do pierwszej) warstw ukrytych:

$$\delta_i^{(l)r} = \beta O_i^{(l)r} (1 - O_i^{(l)r}) \sum_{k=1}^{N_{r+1}} \delta_k^{(l)r+1} w_{ki}$$

- Dokonujemy modyfikacji wag według:

$$\Delta w_{ij}^{(l)} = \eta \delta_i^{(l)} O_j^{(l)}$$

Zalety algorytmu backpropagation

- metoda lokalna
- małe wymagania pamięciowe
- metoda uniwersalna

Wady algorytmu backpropagation

- Metoda wolnozbieżna (zwłaszcza przy płaskich funkcjach błędów – małe kroki, ale także przy zbyt stromych przy zbyt dużym współczynniku uczenia – oscylacje wokół minimum)
- Metoda znajduje minima lokalne funkcji błędu a nie globalne, których się poszukuje

Inercyjne modyfikacje Backpropagation

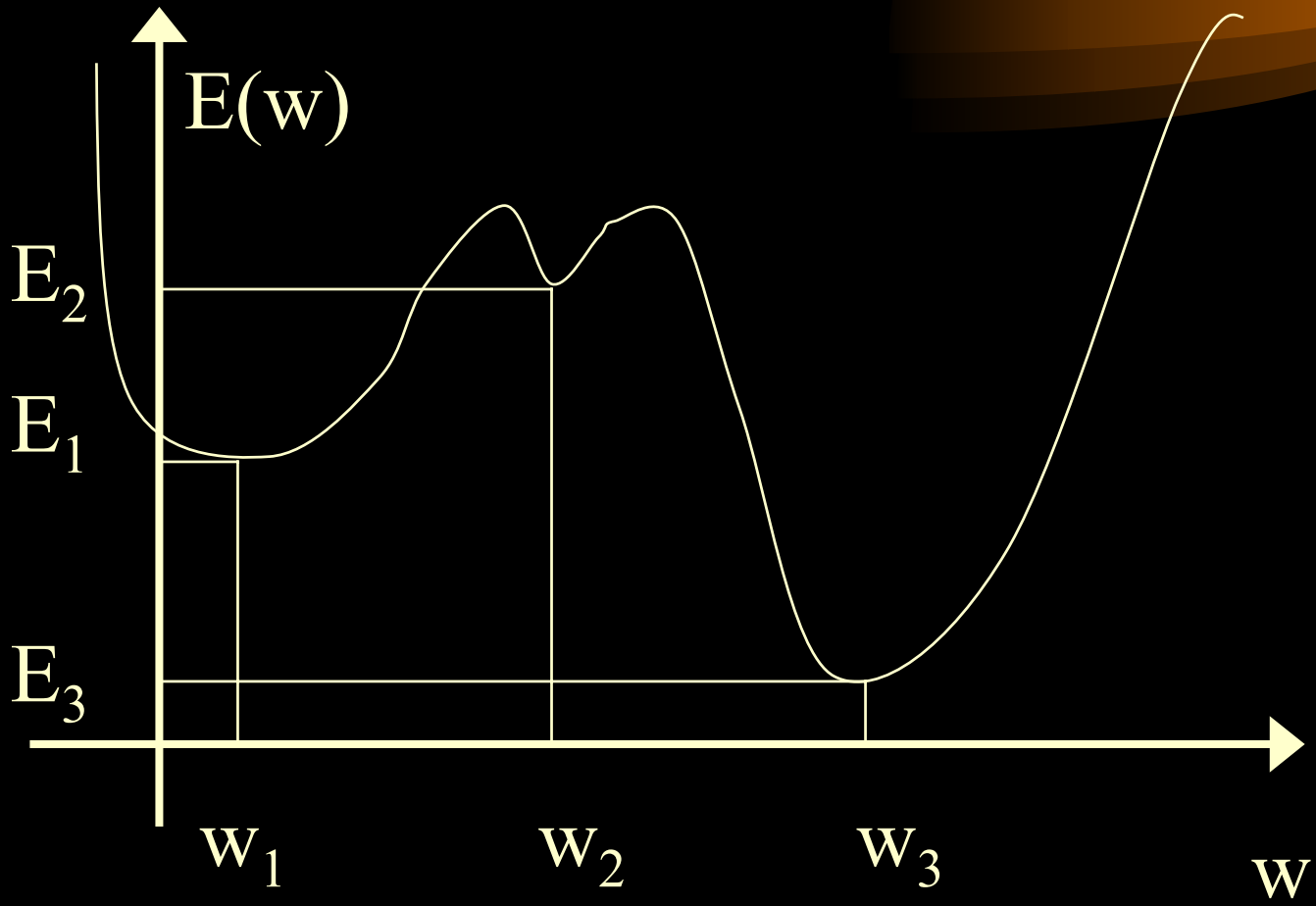
- Uczenie z momentum:

$$\Delta w_{ij}^{(l)} = \eta \delta_i^{(l)} O_j^{(l)} + \mu \Delta w_{ij}^{(l-1)}$$

- Uczenie z wykładniczym wygładzaniem:

$$\Delta w_{ij}^{(l)} = \eta((1-\mu)\delta_i^{(l)} O_j^{(l)} + \mu \Delta w_{ij}^{(l-1)})$$

Minima lokalne vs. minimum globalne (przypadek 1D)



Eliminacja wady typu: znajdowanie minimum lokalnego

- Rozpoczynanie uczenia od wielu różnych warunków początkowych (multistart) – wada: brak dobrego kryterium ustalającego ilość prób
- Stosowanie algorytmów miękkiej selekcji (symulowane wyżarzanie, algorytmy ewolucyjne) – wada: algorytmy te są nieefektywne czasowo