plik wk5.tex, 10 listopada 2010,

5 Algorytm backpropagation, zasady uczenia sieci

5.1 Wstęp

Generalnie, dla wyznaczenia wag w sieciach typu MLP (lub sieciach pracujących na zasadzie *forward*, czyli podawanie sygnału do przodu) stosuje się sekwencyjne algorytmy gradientowe, por. np. Rozdział 3 notatek pt. Błąd sieci i algorytmy jego minimalizacji. Najprostszy algorytm gradientowy działa w następujący sposób: Mamy przybliżenie wektora wag $\mathbf{w}^{(s)}$ uzyskane w s-tej iteracji. Wtedy przybliżenie w następnej, $s+1$-ej iteracji obliczamy jako:

$$\mathbf{w}^{(s+1)} = \mathbf{w}^{(s)} + \Delta\mathbf{w},$$

gdzie $\mathbf{w}^{(s)}$ jest *starym* a $\mathbf{w}^{(s+1)}$ *nowym* wektorem wag; gdzie

$$\Delta\mathbf{w} = -\eta\{\partial E/\partial\mathbf{w}\}|_{w=w^{(s)}},$$

natomiast $\eta > 0$ jest przyjętym współczynnikiem uczenia, na ogół $0 < eta < 1$. Wektor wag jest tutaj rozszerzonym wektorem wag $\mathbf{w} = [w_0, w_1, \ldots, w_d]^T$ zawierającym również składnik w_0 odpowiadający *biasowi*.

Algorytm *backpropagation*, nazywany w skrócie *backprop* został zaproponowany w 1986 roku przez Rumelharta, Hintona i Williamsa [2] dla sieci typu *feedforward*. Są to sieci podające sygnał wejściowy do przodu, poprzez warstwy ukryte, aż do warstwy wyjściowej. Przykładem takiej sieci jest perceptron wielowarstwowy (MLP).

Przedstawiony dalej opis jest wzorowany na książce Haykina [3]. Dobry opis znajduje sie również w książce Bishopa [1].

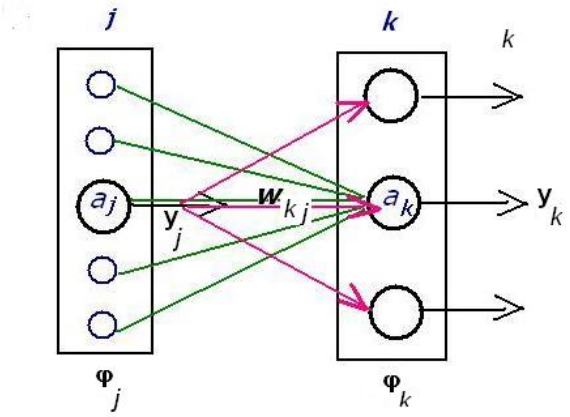
5.2 Oznaczenia

W dalszym ciągu rozpatrzymy perceptron posiadający trzy warstwy: wejściową, ukrytą i wyjściową. Warstwy te będą ogólnie oznaczane symbolami i, j, k pełniącymi podwójną rolę, wskazując na pozycję warstwy oraz jej reprezentatywnego neuronu. Pozycję warstwy jest wskazywana następująco: warstwa 'i' poprzedza warstwę 'j', która z kolei poprzedza warstwę 'k'. Jednocześnie symbole i,j,k oznaczają ogólnie dowolny neuron znajdujący sie w warstwie 'i', 'j' lub 'k'.

Neurony w warstwach 'j', 'k' są neuronami *roboczymi*, tzn. każdy neuron *nr j* otrzymuje sygnały z poprzedniej, i-tej warstwy, przetwarza je (sumowanie z wagami, nałożenie funkcji aktywacji), a wynik wyjściowy tych operacji umieszcza na swoim wyjściu jako nowy sygnał który jest przekazywany do następnej, k-tej, warstwy. Sytuacja taka jest pokazana na rysunku 5.1.

Przyjmujemy następujące oznaczenia:

♠ Symbole i, j, k oznaczają ogólnie neurony znajdujące się w warstwach i, j, k. Liczba neuronów roboczych (z wyłączeniem neuronu odpowiadającego biasowi i przyjmującego tożsamościowo wartość równą $+1.0$) wynosi odpowiednio I, J, K.



Rysunek 5.1: Warstwa środkowa (j) z zaznaczonym neuronem 'j' o aktywacji a_j oraz warstwa następna (k) z zaznaczonym neuronem 'k' o aktywacji a_k. Neuron k ma określone wagi $w_{kj},\ j = 0, 1, \ldots, J$, gdzie J oznacza liczbę neuronów ukrytych w warstwie j; wagi te służą do ważonego sumowanie sygnałów dochodzących z poprzedniej warstwy (j); tym samym aktywacja tego neuronu jest wyznaczona wzorem $a_k = \sum_j y_j w_{kj},\ j = 0, 1, \ldots, J$. Na otrzymaną wartość a_k zostaje nałożona funkcji aktywacji φ_k, w rezultacie tego na wyjściu neuronu k pojawia się sygnał wyjściowy $y_k = \varphi_k(a_k)$.

- ♠ Neurony i, j, k mają wartości wyjściowe y_i, y_j, y_k. Jeśli warstwa 'i' jest warstwą wejściową sieci, to $y_i \equiv x_i$: czyli jest tożsamościowo równa i-tej składowej wektora danych wejściowych $\mathbf{x}$; dla $i = 0$, $y_i \equiv +1$.
- ♠ Neurony j, k mają przypisane wektory wag $\mathbf{w}_j$ oraz $\mathbf{w}_k$. Wektory wag są używane do sumowania sygnałow dochodzacych do neuronów z warstwy j z warstwy poprzedzającej. Obowiązują następujace oznaczenia składowych tych wektorów:

$$\mathbf{w}_j = (w_{ji}),\ i = 0, \ldots, I\,, \qquad \mathbf{w}_k = (w_{kj}),\ j = 0, 1, \ldots, J\,,$$

  tzn. pierwszy wskaźnik oznacza numer neuronu 'pracującego', tj. 'aktywizującego się', a drugi wskaźnik - numer neuronu warstwy poprzedniej który wysłał sygnał do neuronu aktualnie pracującego. Czyli: neuron j otrzymuje sygnał od neuronu 'i', którym może być dowolny neuron warstwy 'i'. Podobnie dla wagi w_{kj}.
- ♠ Symbole φ_j, φ_k oznaczają funkcje aktywacji stosowaną do wszystkich neuronów w warstwach j i k odpowiednio.
- ♠ Symbole a_j, a_k oznaczają aktywacje (induced local field of energy) neuronów 'j' oraz 'k' po otrzymaniu przez nie sygnałów z poprzednich warstw. Wartości te są wyznaczane następująco:

$$a_j = \sum_{i=0}^{I} w_{ji} y_i, \quad a_k = \sum_{j=0}^{J} w_{kj} y_j.$$

♠ Sygnały błędu (error signals) e_k są obliczane na warstwie wyjściowej całej sieci jako $e_k = t_k - y_k$, gdzie t_k jest wartością docelową (targetem) oczekiwanym na k-tym wyjściu sieci (k=1, ..., K).

♠ Funkcja błędu sieci jest obliczana jako

$$E = \sum_{k=1}^{K} \frac{1}{2} e_k^2.$$

Funkcja ta stanowi kryterium optymalizacyjne, które ma być zminimalizowane przez odpowiedni dobór wag. Inaczej mówiąc: chcemy dobrać wagi w ten sposób, aby otrzymać możliwie małą wartość funkcji błędu.

♠ Wartości wyjściowe j-tego i k-tego neuronu wynoszą odpowiednio:

$$y_j = \varphi_j(a_j) = \varphi_j(\sum_{i=0}^{I} w_{ji} y_i), \quad \text{przy czym } a_j = \sum_{i=0}^{I} w_{ji} y_i$$

$$y_k = \varphi_k(a_k) = \varphi_k(\sum_{j=0}^{J} w_{kj} y_j), \quad \text{przy czym } a_k = \sum_{j=0}^{J} w_{kj} y_j.$$

5.3 Działanie algorytmu *backpropagation*

Uwaga: **Algorytm ten stosuje się do sieci typu feedforward o conajmniej 3 warstwach!**

1. Ustalamy próbkę uczącą $\mathbf{X_{N\times d}}$, jej wartości docelowe $\mathbf{T_{N\times K}}$.

2. Wykonujemy N razy następujące czynności (punkty 3 – 7 poniżej)

3. Wybieramy losowo wektor $\mathbf{x}$ próbki uczącej (wiersz tablicy danych), i odpowiadający mu wektor wartości docelowych $\mathbf{t}$ (wiers tablicy $\mathbf{T}$).

4. Podajemy na wejście sieci wektor wektor $\mathbf{x}$ jako jako sygnał wejściowy i interpretujemy jego składowe jako sygnały wyjściowe poszczególnych neuronów $\{y_i\}$ $i = 1, \ldots, I$. Sygnały te są uzupełnione o dodatkową składową $y_0 \equiv +1$. Wartości y_i, $i = 0, \ldots, I$ są propagowane wprzód poprzez wszystkie warstwy sieci, aż dojdziemy do warstwy ostatniej, i otrzymamy ostateczny wynik sieci $\{y_k\}$ $k = 1, \ldots, K$. Pode drogą zapamiętujemy wszystkie sygnały wyjściowe neuronów pracujących $\{y_j\}$ wszystkich warstw przez które przechodziliśmy.

5. Po dojściu do ostatniej warstwy sieci obliczamy błędy e_k oraz wielkości δ_k, $k = 1, \ldots, K$ jako (wyrażenie $\varphi_k'(a_k)$ oznacza pochodną funkcji aktywacji φ_k wziętą względem jej argumentu a_k)

$$\delta_k = -\frac{\partial E}{\partial a_k} = (t_k - y_k) \cdot \varphi_k'(a_k).$$

Szukana poprawka Δw_{kj} do wag neuronów k-tej warstwy wynosi wtedy:

$$\Delta w_{kj} = eta \cdot \delta_k \cdot y_j$$

6. Cofamy się do poprzedniej warstwy i sprawdzamy, czy to jest warstwa wejściowa sieci. Jeśli tak, to przechodzimy do punktu 8; w przeciwnym przypadku nazywamy tę warstwę 'j', a poprzednio obliczaną warstwę 'k' i wykonujemy obliczenia wartości δ_j (szczegółowe obliczenia pokazano niżej w sekcji 5.4)

$$\delta_j = -\frac{\partial E}{\partial a_j} = (\sum_k \delta_k w_{kj})\varphi_j{}'(a_j).$$

Szukana poprawka Δw_{ji} do wag neuronów j-tej warstwy wynosi wtedy:

$$\Delta w_{ji} = eta \cdot \delta_j \cdot y_i.$$

7. Postępowanie z punktu 6 powyżej powtarzamy tak długo, aż uzyskamy poprawki dla wszystkich warstw ukrytych sieci.

Wykonanie czynności punktów 2 – 7 (tj. wykonanie poprawek wyznaczanych na podstawie N (spermutowanych) wierszy danych uczących) nazywamy **epoką**. Generalnie postępowanie to powtarzamy określoną liczbę razy (max_l_epok) lub też określając dopuszczalną wielkość błędu sieci.

5.4 Etapy algorytmu backprop

Algorytm backpropagation proponuje wyznaczać przyrost Δw_{ji} według wzoru ($j = 1, \ldots, J;\ i = 1, \ldots, I$):

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} = -\eta \frac{\partial E}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}} = \eta\, \delta_j\, y_i \quad \text{gdzie} \quad \delta_j = -\frac{\partial E}{\partial a_j}. \tag{5.1}$$

Tak więc, kolejna poprawka Δw_{ji} zależy implicite tylko od wartości y_i podawanej przez i-ty neuron warstwy poprzedzającej oraz od wartości pochodnej $\partial E/\partial a_j$, która to pochodna, – wzięta ze znakiem '–', stanowi definicję symbolu δ_j.

5.4.1 Obliczanie δ_k dla ostatniej warstwy (wyjściowej)

Tutaj warstwa robocza jest jednocześnie warstwą wyjściową. Neurony tej warstwy są oznaczaczane numerami $k = 1, \ldots, K$; natomiast wyjścia neuronów tej warstwy oznaczamy $y_k,\ k = 1, \ldots, K$. Funkcja aktywacji tej warstwy jest oznaczona symbolem $\varphi_k(\ .\ )$.

Obliczymy teraz pochodną $\partial E/\partial w_{kj}$ potrzebną do poprawienia aktualnej wartości wagi w_{kj}. Otrzymujemy kolejno:

$$\frac{\partial E}{\partial w_{kj}} = -\eta \frac{\partial E}{\partial y_k}\frac{\partial\, y_k}{\partial\, w_{kj}} = -\eta \underbrace{\underbrace{\frac{\partial E}{\partial y_k}}_{-(t_k - y_k)} \cdot \underbrace{\frac{\partial y_k}{\partial a_k}}_{\varphi_k'(a_k)}}_{-\delta_k} \cdot \underbrace{\frac{\partial a_k}{\partial w_{kj}}}_{y_j} = \eta\, \delta_k\, y_j$$

gdzie symbolem δ_k oznaczyliśmy wyrażenie (obliczone z warstwy wyjściowej sieci)

$$\delta_k = -\frac{\partial E}{\partial a_k} = (t_k - y_k)\, \varphi_a'(a_k).$$

5.4.2 Obliczanie δ_j dla warstwy środkowej

Obliczymy teraz δ_j dla warstwy środkowej, poprzedzającej warstwę 'k'. Sytuacja taka jest pokazana na rysunku 5.1. Mamy ($j = 1, \ldots, J$):

$$\underbrace{\delta_j = -\frac{\partial E}{\partial a_j}}_{z \quad definicji} = -\sum_{k=1}^{K} \frac{\partial E}{\partial a_k} \cdot \frac{\partial a_k}{\partial a_j} = -\sum_k \underbrace{(\frac{\partial E}{\partial a_k})}_{-\delta_k} \cdot \underbrace{\frac{\partial a_k}{\partial y_j}}_{w_{kj}} \cdot \underbrace{\frac{\partial y_j}{\partial a_j}}_{\varphi_j'(a_j)}$$

Ponieważ $a_k = \sum_j w_{kj} y_j$, oraz $y_j = \varphi_j(a_j)$, otrzymujemy ostatecznie

$$\delta_j = -\frac{\partial E}{\partial a_j} = \sum_k \delta_k w_{kj} \varphi_j'(a_j) = \varphi_j'(a_j) \sum_k \delta_k w_{kj}.$$

Ostateczny wzór na δ_j:

$$\delta_j = (\sum_{k=1}^{K} \delta_k w_{kj})\, \varphi_j'(a_j)\,. \tag{5.2}$$

Wzór ten pozwala wyznaczyć delty warstwy środkowej z delt i wag przypisanych warstwie następnej. Rolę sygnału błędu neuronu 'j' pelni tu ważona suma delt δ_k warstwy następnej, w kórej to sumie delty δ_k są ważone wagami w_{kj}.

Dokładna analiza złożoności obliczeniowej tego algorytmu znajduje się w książce Bishopa [1].

Literatura

[1] Ch. M. Bishop, *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford UP, 1st published 1995, 15th reprint 2008.

[2] D.E. Rumelhart, G. Hinton, R.J. Williams, Learning representation by back-propagating, *Nature* vol. 323, 9 October 1986, 533-536.

[3] Simon Haykin. Neural Networks: A Comprehensive Foundation. Macmillan, New York, NY, 1994, 2nd Edition: Pearson, Prentice Hall, Pearson Education, Ninth Reprint 2005.

[4] Raul Rojas, *Neural Networks, A systematic Introduction.* Springer 1996. Rozdział 3: Weighted Networks – The Perceptron, str 55–76; Rozdział 4: Perceptron Learning, str 77–99 .