5. Uczenie głębokie (Deep learning)

Propagacja wsteczna (Back Propagation, skrótowo BP) to w sensie już jest uczeniem głębokiem, tylko na dzisiejszy dzień uczenie głębokie oznacza różne sposoby dodatkowe aby przyspieszać (wzmacniać) BP. Jeśli BP jest "głębokie", to zawiera dużo parametrów i trudno uzyskać optymalne rozwiązanie za pomocą metody gradientu. Stosując autoenkoder i CNN (Convolutional Neural Network), możemy uniknąć ten problem.

5.1. Autoenkoder

Zbiór sygnałów wejściowych (input) jest

$$U = \{u(1), \dots, u(P)\},$$

 $u(p) = (u_1(p), \dots, u_m(p)) \in \{0, 1\}^m \subset \mathbb{R}^m$

dla p = 1, ..., P.

Sygnał wyjściowy (output) $d_i(p) \in D = \{d(1), \dots, d(P)\}$ ma postać

$$d(p) = (d_1(p), \dots, d_m(p)) \qquad (\ell = m)$$

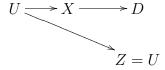
dla p = 1, ..., P.

W autoenkoderze, sygnały nauczyciela $Z = \{z(1), \ldots, z(P)\}$ jest U:

$$z(p) = (z_1(p), \dots, z_m(p)) = u(p) = (u_1(p), \dots, u_m(p))$$
 $(\ell = m)$

dla p = 1, ..., P.

Mianowicie autoenkoder nauczy się siebie.



Tak jak w uczeniu XORu, X jest zbiorem sygnałów z warstwy między U a D(=Y):

$$X = \{x(1), \dots, x(P)\}.$$

Każdy element $x(p) \in X$ ma postać $x(p) = (x_1(p), \dots, x_n(p)) \in \mathbb{R}^n$ dla $p = 1, \dots, P$. W autoenkoderze często weźmiemy $n \ll m$. Czyli informacja U jest kodowana (szyfrowana) w X. $U \to X$ to kodowanie, a $X \to D$ to dekodowanie.

Pierwsa warstwa neuronów: Dla każdego $p = 1, \ldots, P$

$$x_i(p) = f(\sum_{j=1}^m w_{ij}u_j(p)) \quad (i = 1, ..., n).$$

Druga (i ostatnia) warstwa neuronów: Dla każdego $p = 1, \ldots, P$

$$d_j(p) = f(\sum_{i=1}^n s_{ji}x_i(p)).$$

Celem propagacji wstecznej jest minimalizcja błędu

$$E = E(w_{ij}, s_{ji}) = \frac{1}{2} \sum_{p=1}^{P} \sum_{j=1}^{\ell} (d_j(p) - z_j(p))^2 = \frac{1}{2} \sum_{p=1}^{P} \sum_{j=1}^{\ell} (d_j(p) - u_j(p))^2$$

przy użyciu metody gradientu.

Uwaga Korzystając z oznaczenia wektorów i macierz, można opisać powyższą sytuację w

następujący sposób.

$$\vec{u}(p) = \begin{bmatrix} u_1(p) \\ \vdots \\ u_m(p) \end{bmatrix}, W = \begin{bmatrix} w_{11} & \cdots & w_{1m} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nm} \end{bmatrix}, \vec{x}(p) = \begin{bmatrix} x_1(p) \\ \vdots \\ x_n(p) \end{bmatrix} = W\vec{u}(p),$$

$$\vec{f}_n(\vec{x}(p)) = \begin{bmatrix} f(x_1(p)) \\ \vdots \\ f(x_n(p)) \end{bmatrix},$$

$$S = \begin{bmatrix} s_{11} & \cdots & s_{1n} \\ \vdots & \ddots & \vdots \\ s_{m1} & \cdots & s_{mn} \end{bmatrix}, \vec{d}(p) = \begin{bmatrix} d_1(p) \\ \vdots \\ d_m(p) \end{bmatrix} = \vec{f}_m(S\vec{f}_n(\vec{x}(p))) = \vec{f}_m(S\vec{f}_n(W\vec{u}(p)))$$

$$E = E(W, S) = \frac{1}{2} \sum_{p=1}^{P} ||\vec{d}(p) - \vec{u}(p)||^2 \quad \text{(norma euklidesowa)}$$

$$= \frac{1}{2} \sum_{p=1}^{P} ||\vec{f}_m(S\vec{f}_n(\vec{x}(p))) - \vec{u}(p)||^2$$

$$= \frac{1}{2} \sum_{p=1}^{P} ||\vec{f}_m(S\vec{f}_n(W\vec{u}(p))) - \vec{u}(p)||^2$$

Zauważmy, że jeśli $\vec{f}_n(\vec{x}) = \vec{x}$ itd. (liniowa funkcja progowa), to $\vec{d}(p) = SW\vec{u}(p)$. Wówczas E staje się nastepująco.

$$E = E(W, S) = \frac{1}{2} \sum_{p=1}^{P} ||SW\vec{u}(p) - \vec{u}(p)||^{2}$$

Komentarz (Google Cat Paper)

Znany artykuł "Google Cat Paper" zawiera tylko jedeną formułę (str. 4/11), która właśnie dotyczy autoenkodera.

$$\underset{W_1, W_2}{\text{minimize}} \sum_{i=1}^m \left(\|W_2 W_1^T x^{(i)} - x^{(i)}\|_2^2 + \lambda \sum_{i=1}^k \sqrt{\epsilon + H_j(W_1^T x^{(i)})^2} \right)$$

Nasza funkcja aktywacji f(x) = x w tej pracy Google'a, a nie $f(x) = \frac{1}{1 + e^{-\beta x}}$. Wyraz

$$\sum_{i=1}^{m} \|W_2 W_1^T x^{(i)} - x^{(i)}\|_2^2$$

jest dokładnie nasze kryterium $E(w_{ij}, s_{ji}) = E(W, S)$ z liniową funkcją progową f(x) = x. Natomiasto drugi wyraz $H_j(W_1^T x^{(i)})^2$ został wprowadzony, żeby współczynnik W_1 (macierz składająca się z naszych w_{ij}) był numerycznie stabilny (opanowany).

Cytat z Google Cat Paper

"3.3. Learning and Optimization Learning: During learning, the parameters of the second

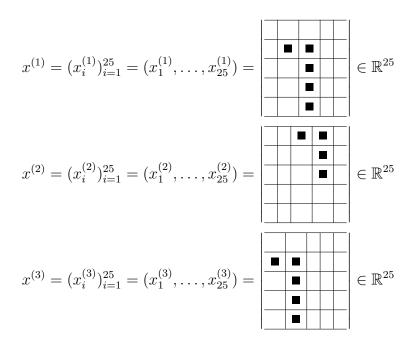
sublayers (H) are fixed to uniform weights, whereas the encoding weights W_1 and decoding weights W_2 of the first sublayers are adjusted using the following optimization problem

$$\underset{W_1, W_2}{\text{minimize}} \sum_{i=1}^{m} \left(\|W_2 W_1^T x^{(i)} - x^{(i)}\|_2^2 + \lambda \sum_{i=1}^{k} \sqrt{\epsilon + H_j(W_1^T x^{(i)})^2} \right)$$
(1)

Here, λ is a tradeoff parameter between sparsity and reconstruction; m, k are the number of examples and pooling units in a layer respectively; H_j is the vector of weights of the j-th pooling unit. In our experiments, we set $\lambda = 0.1$." Koniec cytatu

Przykład.

Wejścia (input):
$$x^{(\alpha)} = (x_i^{(\alpha)})_{i=1}^{25} = (x_1^{(\alpha)}, \dots, x_{25}^{(\alpha)}) \in \mathbb{R}^{25}$$
 $(1 \le \alpha \le 3)$ $\square = 0.0, \blacksquare = 1.0$



$$\begin{array}{l} \underline{\operatorname{Enkoder}} \ x^{(\alpha)} \mapsto y^{(\alpha)} \quad (1 \leq \alpha \leq 3) \\ \operatorname{Warstwa} \ \operatorname{\acute{s}rodkowa} \colon \ y^{(\alpha)} = (y_i^{(\alpha)})_{i=1}^{16} = (y_1^{(\alpha)}, \dots, y_{16}^{(\alpha)}) \in \mathbb{R}^{16} \quad \ (1 \leq \alpha \leq 3) \\ \end{array}$$

$$y_i^{(\alpha)} = f(\sum_{j=1}^{25} w_{ij} x_j^{(\alpha)} + b_i) \quad (1 \le i \le 16) \quad (1 \le \alpha \le 3)$$

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

$$\frac{\text{Dekoder}}{\text{Wyjścia (output):}} \ x'^{(\alpha)} \quad (1 \leq \alpha \leq 3)$$

$$\text{Wyjścia (output):} \ x'^{(\alpha)} = (x_i'^{(\alpha)})_{i=1}^{25} = (x_1'^{(\alpha)}, \dots, x_{25}'^{(\alpha)}) \in \mathbb{R}^{25} \quad (1 \leq \alpha \leq 3)$$

$$x_k^{\prime(\alpha)} = f(\sum_{i=1}^{16} w_{ki}' y_i^{(\alpha)} + b_k') \quad (1 \le k \le 25) \quad (1 \le \alpha \le 3)$$

Wyjścia (output) dla wyświetlenia: $x''^{(\alpha)} = (x_i''^{(\alpha)})_{i=1}^{25} = (x_1''^{(\alpha)}, \dots, x_{25}''^{(\alpha)}) \in \mathbb{R}^{25}$ $(1 \le \alpha \le 3)$

$$x_k''^{(\alpha)} = f_1(\sum_{i=1}^{16} w_{ki}' y_i^{(\alpha)} + b_k') \quad (1 \le k \le 25) \quad (1 \le \alpha \le 3)$$

$$f_1(x) = \begin{cases} 0 & \text{gdy } x < 0 \\ 1 & \text{gdy } x \ge 0 \end{cases}$$

Cel. Za pomocą metody gradientu

$$E = \frac{1}{2} \sum_{\alpha=1}^{3} \sum_{k=1}^{25} (x_k'^{(\alpha)} - x_k^{(\alpha)})^2 \to \text{minimum (lokalne)}.$$

Gradienty

$$\frac{\partial E}{\partial w'_{pq}} = \sum_{\alpha=1}^{3} (x'_{p}^{(\alpha)} - x_{p}^{(\alpha)}) f'(\sum_{i=1}^{16} w'_{pi} y_{i}^{(\alpha)} + b'_{p}) y_{q}^{(\alpha)} \quad (1 \le p \le 25, 1 \le q \le 16)$$

$$\frac{\partial E}{\partial b'_{p}} = \sum_{\alpha=1}^{3} (x'_{p}^{(\alpha)} - x_{p}^{(\alpha)}) f'(\sum_{i=1}^{16} w'_{pi} y_{i}^{(\alpha)} + b'_{p}) \quad (1 \le p \le 25)$$

$$\frac{\partial E}{\partial w_{pq}} = \sum_{\alpha=1}^{3} \sum_{k=1}^{25} (x'_{k}^{(\alpha)} - x_{k}^{(\alpha)}) f'(\sum_{i=1}^{16} w'_{ki} y_{i}^{(\alpha)} + b'_{k}) w'_{kp} f'(\sum_{j=1}^{25} w_{pj} x_{j}^{(\alpha)} + b_{p}) x_{q}^{(\alpha)} \quad (1 \le p \le 16, 1 \le q \le 25)$$

$$\frac{\partial E}{\partial b_{p}} = \sum_{\alpha=1}^{3} \sum_{k=1}^{25} (x'_{k}^{(\alpha)} - x_{k}^{(\alpha)}) f'(\sum_{i=1}^{16} w'_{ki} y_{i}^{(\alpha)} + b'_{k}) w'_{kp} f'(\sum_{j=1}^{25} w_{pj} x_{j}^{(\alpha)} + b_{p}) \quad (1 \le p \le 16)$$

Koniec przykładu

Uwaga

Enkoder i dekoder moga mieć kilka warstw. X ma warstwy jak

$$X = \coprod_{q=1}^{Q} X^{(q)}, \quad X^{(q)} \subset \mathbb{R}^{n_q}.$$

Enkoder jest

$$X^{(1)} \rightarrow \cdots \rightarrow X^{(Q)}$$

a dekoder ma postać

$$D = \coprod_{q=1}^{Q} D^{(q)}, \quad D^{(q)} \subset \mathbb{R}^{n_q}.$$

Dekoder jest

$$D^{(Q)} \to D^{(Q-1)} \to \cdots D^{(1)} \to D^{(0)}$$

Propagację wsteczną stosuje się do

$$U \xrightarrow{X^{(1)}} \cdots \xrightarrow{X^{(Q)}} D^{(Q)} \xrightarrow{} \cdots \xrightarrow{} D^{(1)} \xrightarrow{} D^{(0)}$$

Zastosowanie do uczenia głębokiego

W ogólnej sytuacji metoda gradientu dla propagacji wstecznej (czyli uczenia głębokiego) nie

działa skutecznie bo mamy za dużo parametrów $w_{i_{r+1},i_r}^{(r)}$ do regurowania. Zamiat otrzymania wszystkich wag na raz, spórbujemy regulować wagi po kawałkach warstw neuronów przy użyciu autoenkoderów, a potem regurujemy wszystkie wagi globalnie startując z tych otrzymanych wag jako warunki początkowe.

Najpierw rozważmy kawałek

$$U = X^{(0)} \to X^{(1)}$$

i regurujemy tylko $w^{(0)}_{i_1,i_0}$. Potrakutujemy $U=X^{(0)}\to X^{(1)}$ jako enkoder i do tego połączymy dekoder $X^{(1)}\to D^{(1)}$, który jest zdefiniowany wzorem

$$d_{i_0}^{(0)}(p) = f(\sum_{i_1=1}^{n_1} s_{i_0,i_1}^{(0)} x_{i_1}^{(1)}(p)) \quad (i_0 = 1, \dots, n_0 = m).$$

W sumie otrzymujemy autoenkoder

$$U = X^{(0)} \to X^{(1)} \to D^{(0)}$$

Sygnały $x^{(1)}(p)$ pierwszej warstwy są obliczone wzorami

$$x_{i_1}^{(1)}(p) = f(\sum_{i_0=1}^{n_0} w_{i_1,i_0}^{(1)} u_{i_0}(p)) \quad (n_0 = m, i_1 = 1, \dots, n_1).$$

Kryterium, które chcemy minimalizować, jest

$$E = E(w_{i_1,i_0}^{(1)}, s_{i_0,i_1}^{(0)}) = \frac{1}{2} \sum_{p=1}^{P} \sum_{i_0=1}^{n_0} (d_{i_0}^{(0)}(p) - u_{i_0}(p))^2.$$

Teraz z tymi $x_{i_1}^{(1)}(p)$ rozważmy enkoder $X^{(1)} \to X^{(2)}$:

$$x_{i_2}^{(2)}(p) = f(\sum_{i_1=1}^{n_1} w_{i_2,i_1}^{(2)} x_{i_1}^{(1)}(p)) \quad (i_2 = 1, \dots, n_2).$$

Do tego połączymy dekoder $X^{(2)} \to D^{(2)}$, który jest zdefiniowany wzorem

$$d_{i_1}^{(1)}(p) = f(\sum_{i_2=1}^{n_2} s_{i_1,i_2}^{(1)} x_{i_2}^{(2)}(p)) \quad (i_1 = 1, \dots, n_1).$$

Kryterium, które minimalizujemy, jest

$$E = E(w_{i_2,i_1}^{(2)}, s_{i_1,i_2}^{(1)}) = \frac{1}{2} \sum_{p=1}^{P} \sum_{i_1=1}^{n_1} (d_{i_1}^{(1)}(p) - x_{i_1}^{(1)}(p))^2.$$

W sposób iteracyjny jeśli mamy sygnały $x_{i_r}^{(r)}(p)$, to enkoder podany jest wzorem

$$x_{i_{r+1}}^{(r+1)}(p) = f(\sum_{i_r=1}^{n_r} w_{i_{r+1},i_r}^{(r)} x_{i_r}^{(r)}(p)) \quad (i_{r+1} = 1, \dots, n_{r+1}).$$

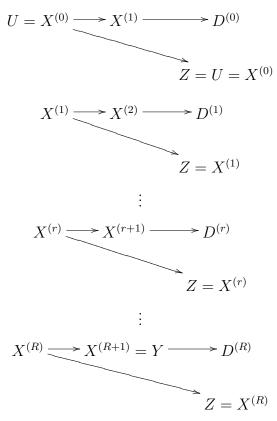
Dekoderem jest

$$d_{i_r}^{(r)}(p) = f(\sum_{i_{r+1}=1}^{n_{r+1}} s_{i_r, i_{r+1}}^{(r)} x_{i_{r+1}}^{(r+1)}(p)) \quad (i_r = 1, \dots, n_r).$$

Kryterium, które będzie minimalizowane, jest

$$E = E(w_{i_{r+1},i_r}^{(r)}, s_{i_r,i_{r+1}}^{(r)}) = \frac{1}{2} \sum_{p=1}^{P} \sum_{i_r=1}^{n_r} (d_{i_r}^{(r)}(p) - x_{i_r}^{(r)}(p))^2.$$

Podobną procedurę kontynujemy aż do r = R + 1.



W ten sposób uzyskaliśmy $w_{i_{r+1},i_r}^{(r)}$ dla wszystkich $r=0,\ldots,R$. Startując z tych parametrów teraz wykonujemy propagacę wsteczną dla całości tak jak w poprzednim wykładzie

$$X^{(0)} = U \to X^{(1)} \to X^{(2)} \to \cdots \to X^{(R)} \to X^{(R+1)} = Y.$$