SNE 7

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 $p = 1, \dots, P$.

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

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

dla $p = 1, \dots, P$.

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

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

dla $p = 1, \dots, P$.

Mianowicie autoenkoder nauczy się siebie.

$$\begin{tikzcd} U \arrow[r] \arrow[drr] & X \arrow[r] & D \\ & & Z = U \end{tikzcd}$$

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 $p = 1, \dots, P$

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

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

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

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.

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

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

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

$$\begin{aligned} E = E(W,S) &= \frac{1}{2}\sum_{p=1}^{P} \|\vec{d}(p) - \vec{u}(p)\|^2 \quad \text{(norma euklidesowa)} \\ &= \frac{1}{2}\sum_{p=1}^{P} \|\vec{f_m}(S\vec{f_n}(\vec{x}(p))) - \vec{u}(p)\|^2 \\ &= \frac{1}{2}\sum_{p=1}^{P} \|\vec{f_m}(S\vec{f_n}(W\vec{u}(p))) - \vec{u}(p)\|^2 \end{aligned}$$

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.

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

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

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

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): $x^{(\alpha)} = (x_i^{(\alpha)})_{i=1}^{25} = (x_1^{(\alpha)}, \ldots, x_{25}^{(\alpha)}) \in \mathbb{R}^{25} \quad (1 \leq \alpha \leq 3)$
$\square = 0.0$, $\blacksquare = 1.0$

$x^{(1)} = (x_i^{(1)})_{i=1}^{25} = (x_1^{(1)}, \ldots, x_{25}^{(1)}) =$

	■	■		
		■		
		■		
		■		

$\in \mathbb{R}^{25}$

$x^{(2)} = (x_i^{(2)})_{i=1}^{25} = (x_1^{(2)}, \ldots, x_{25}^{(2)}) =$

		■	■	
			■	
			■	

$\in \mathbb{R}^{25}$

$x^{(3)} = (x_i^{(3)})_{i=1}^{25} = (x_1^{(3)}, \ldots, x_{25}^{(3)}) =$

■	■			
	■			
	■			
	■			

$\in \mathbb{R}^{25}$

<u>Enkoder</u> $x^{(\alpha)} \mapsto y^{(\alpha)} \quad (1 \leq \alpha \leq 3)$
Warstwa środkowa: $y^{(\alpha)} = (y_i^{(\alpha)})_{i=1}^{16} = (y_1^{(\alpha)}, \ldots, y_{16}^{(\alpha)}) \in \mathbb{R}^{16} \quad (1 \leq \alpha \leq 3)$

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

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

<u>Dekoder</u> $y^{(\alpha)} \mapsto x'^{(\alpha)} \quad (1 \leq \alpha \leq 3)$
Wyjścia (output): $x'^{(\alpha)} = (x_i'^{(\alpha)})_{i=1}^{25} = (x_1'^{(\alpha)}, \ldots, x_{25}'^{(\alpha)}) \in \mathbb{R}^{25} \quad (1 \leq \alpha \leq 3)$

$$x_k'^{(\alpha)} = f(\sum_{i=1}^{16} w'_{ki} y_i^{(\alpha)} + b'_k) \quad (1 \leq k \leq 25) \quad (1 \leq \alpha \leq 3)$$

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

$$x_k''^{(\alpha)} = f_1(\sum_{i=1}^{16} w'_{ki} y_i^{(\alpha)} + b'_k) \quad (1 \leq k \leq 25) \quad (1 \leq \alpha \leq 3)$$

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

Cel. Za pomocą metody gradientu

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

<u>Gradienty</u>

$$\frac{\partial E}{\partial w'_{pq}} = \sum_{\alpha=1}^{3} (x_p'^{(\alpha)} - x_p^{(\alpha)}) f'(\sum_{i=1}^{16} w'_{pi} y_i^{(\alpha)} + b'_p) y_q^{(\alpha)} \quad (1 \leq p \leq 25, 1 \leq q \leq 16)$$

$$\frac{\partial E}{\partial b'_p} = \sum_{\alpha=1}^{3} (x_p'^{(\alpha)} - x_p^{(\alpha)}) f'(\sum_{i=1}^{16} w'_{pi} y_i^{(\alpha)} + b'_p) \quad (1 \leq p \leq 25)$$

$$\frac{\partial E}{\partial w_{pq}} = \sum_{\alpha=1}^{3} \sum_{k=1}^{25} (x_k'^{(\alpha)} - x_k^{(\alpha)}) f'(\sum_{i=1}^{16} w'_{ki} y_i^{(\alpha)} + b'_k) w'_{kp} f'(\sum_{j=1}^{25} w_{pj} x_j^{(\alpha)} + b_p) x_q^{(\alpha)} \quad (1 \leq p \leq 16, 1 \leq q \leq 25)$$

$$\frac{\partial E}{\partial b_p} = \sum_{\alpha=1}^{3} \sum_{k=1}^{25} (x_k'^{(\alpha)} - x_k^{(\alpha)}) f'(\sum_{i=1}^{16} w'_{ki} y_i^{(\alpha)} + b'_k) w'_{kp} f'(\sum_{j=1}^{25} w_{pj} x_j^{(\alpha)} + b_p) \quad (1 \leq p \leq 16)$$

Koniec przykładu

Uwaga

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

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

Enkoder jest

$$X^{(1)} \to \cdots \to X^{(Q)}.$$

a dekoder ma postać

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

Dekoder jest

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

Propagację wsteczną stosuje się do

$$\begin{tikzcd}
U \arrow[r] \arrow[drrrrrrr] & X^{(1)} \arrow[r] & \cdots \arrow[r] & X^{(Q)} \arrow[r] & D^{(Q)} \arrow[r] & \cdots \arrow[r] & D^{(1)} \arrow[r] & D^{(0)} \\
& & & & & & & Z = U
\end{tikzcd}$$

<u>Zastosowanie do uczenia głębokiego</u>

W ogólnej sytuacji metoda gradientu dla propagacji wstecznej (czyli uczenia głębokiego) nie działa skutecznie bo mamy za dużo parametrów $w^{(r)}_{i_{r+1},i_r}$ 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

$$d^{(0)}_{i_0}(p) = f(\sum_{i_1=1}^{n_1} s^{(0)}_{i_0,i_1} x^{(1)}_{i_1}(p)) \quad (i_0 = 1, \ldots, n_0 = m).$$

W sumie otrzymujemy autoenkoder

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

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

$$x^{(1)}_{i_1}(p) = f(\sum_{i_0=1}^{n_0} w^{(1)}_{i_1,i_0} u_{i_0}(p)) \quad (n_0 = m, i_1 = 1, \ldots, n_1).$$

Kryterium, które chcemy minimalizować, jest

$$E = E(w^{(1)}_{i_1,i_0}, s^{(0)}_{i_0,i_1}) = \frac{1}{2}\sum_{p=1}^{P}\sum_{i_0=1}^{n_0}(d^{(0)}_{i_0}(p) - u_{i_0}(p))^2.$$

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

$$x^{(2)}_{i_2}(p) = f(\sum_{i_1=1}^{n_1} w^{(2)}_{i_2,i_1} x^{(1)}_{i_1}(p)) \quad (i_2 = 1, \ldots, n_2).$$

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

$$d^{(1)}_{i_1}(p) = f(\sum_{i_2=1}^{n_2} s^{(1)}_{i_1,i_2} x^{(2)}_{i_2}(p)) \quad (i_1 = 1, \ldots, n_1).$$

Kryterium, które minimalizujemy, jest

$$E = E(w^{(2)}_{i_2,i_1}, s^{(1)}_{i_1,i_2}) = \frac{1}{2}\sum_{p=1}^{P}\sum_{i_1=1}^{n_1}(d^{(1)}_{i_1}(p) - x^{(1)}_{i_1}(p))^2.$$

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

$$x^{(r+1)}_{i_{r+1}}(p) = f(\sum_{i_r=1}^{n_r} w^{(r)}_{i_{r+1},i_r} x^{(r)}_{i_r}(p)) \quad (i_{r+1} = 1, \ldots, n_{r+1}).$$

Dekoderem jest

$$d^{(r)}_{i_r}(p) = f(\sum_{i_{r+1}=1}^{n_{r+1}} s^{(r)}_{i_r,i_{r+1}} x^{(r+1)}_{i_{r+1}}(p)) \quad (i_r = 1, \ldots, n_r).$$

Kryterium, które będzie minimalizowane, jest

$$E = E(w^{(r)}_{i_{r+1},i_r}, s^{(r)}_{i_r,i_{r+1}}) = \frac{1}{2}\sum_{p=1}^{P}\sum_{i_r=1}^{n_r}(d^{(r)}_{i_r}(p) - x^{(r)}_{i_r}(p))^2.$$

Podobną procedurę kontynujemy aż do $r = R+1$.

$$\begin{tikzcd}
U = X^{(0)} \arrow[r] \arrow[drr] & X^{(1)} \arrow[r] & D^{(0)} \\
 & & Z = U = X^{(0)}
\end{tikzcd}$$

$$\begin{tikzcd}
X^{(1)} \arrow[r] \arrow[drr] & X^{(2)} \arrow[r] & D^{(1)} \\
 & & Z = X^{(1)}
\end{tikzcd}$$

$$\vdots$$

$$\begin{tikzcd}
X^{(r)} \arrow[r] \arrow[drr] & X^{(r+1)} \arrow[r] & D^{(r)} \\
 & & Z = X^{(r)}
\end{tikzcd}$$

$$\vdots$$

$$\begin{tikzcd}
X^{(R)} \arrow[r] \arrow[drr] & X^{(R+1)} = Y \arrow[r] & D^{(R)} \\
 & & Z = X^{(R)}
\end{tikzcd}$$

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

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