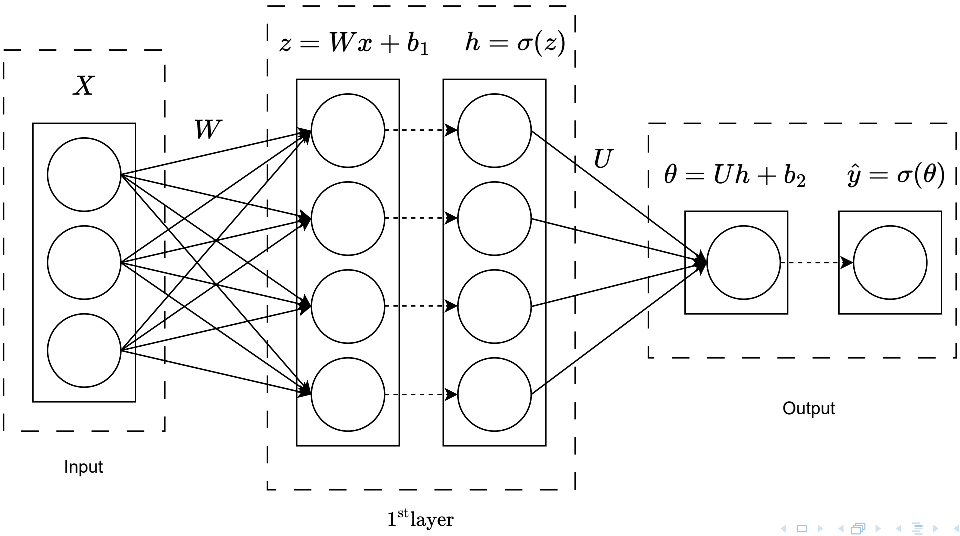
Liczenie gradientów w prostej sieci neuronowej z jedną warstwą ukrytą

Kamil Kmita

Zaawansowane Metody Uczenia Maszynowego
MiNI PW

Architektura sieci



Uwagi

- co do zasady, wagi indeksuje się jako $W^{(1)}$, $W^{(2)}$ itd., tu dla uproszczenia W oraz U,
- nie reprezentujemy X jako $[1|X]$, bias nie jest elementem macierzy W (konwencja tak jak w Keras),
- σ to sigmoid,
- $y \in \{0, 1\}$, $\hat{y} \in (0, 1)$,
- funkcja straty $L(\hat{y}, y) = CE(\hat{y}, y)$ to *binary cross-entropy*, zob. `https://www.tensorflow.org/api_docs/python/tf/keras/losses/BinaryCrossentropy`,
- użyjemy algorytmu SGD.

Literatura i zadanie

Źródła:

1. `https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/readings/gradient-notes.pdf` (uwaga na możliwe literówki),
2. `https://www.deeplearningbook.org/contents/mlp.html`, rozdział 5, w szczególności 6.5 *Back-Propagation Computation in Fully Connected MLP*.

Potrzebujemy do SGD następujących gradientów:

$$\frac{\partial L}{\partial U}, \frac{\partial L}{\partial b_2}, \frac{\partial L}{\partial W}, \frac{\partial L}{\partial b_1}$$

Gradient δ_1

Dwa gradienty do policzenia zawierają wspólną część:

$$\frac{\partial L}{\partial U} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \theta} \frac{\partial \theta}{\partial U}, \tag{1}$$

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \theta} \frac{\partial \theta}{\partial b_2}. \tag{2}$$

$$\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \theta} = \delta_1 = \hat{y} - y = \epsilon \in (-1, 1), \tag{3}$$

bo σ to sigmoid, a funkcja straty to CE.

Gradient δ_2

Podobnie licząc gradienty $\frac{\partial L}{\partial W}, \frac{\partial L}{\partial b_2}$ wystąpi część wspólna $\frac{\partial L}{\partial z}$.

Intuicja: "(...) convert the gradient (...) into a gradient on the pre-nonlinearity activation." (źródło 2).

$$\frac{\partial L}{\partial z} = \delta_2 = \frac{\partial L}{\partial \theta}\frac{\partial \theta}{\partial h}\frac{\partial h}{\partial z} \overset{(1)}{=} \delta_1 U \frac{\partial h}{\partial z} \overset{(4)}{=} \delta_1 U \circ \sigma'(z), \tag{4}$$

gdzie (1) oraz (4) odnoszą się do operacji opisanych w źródle 1, a *circ* to element-wise multiplication.

Gradient δ_2 c.d.

$$
\begin{aligned}
\delta_2 &= \epsilon \cdot [u_{11}, u_{21}, u_{31}, u_{41}] \circ \begin{bmatrix} \sigma'(z_1) \\ \sigma'(z_2) \\ \sigma'(z_2) \\ \sigma'(z_2) \end{bmatrix} && (5) \\
&= \epsilon \cdot [u_{11} \cdot \sigma'(z_1), u_{21} \cdot \sigma'(z_2), u_{31} \cdot \sigma'(z_3), u_{41} \cdot \sigma'(z_4)], && (6)
\end{aligned}
$$

Gradient względem macierzy W

Niech $\alpha = 4$ będzie rozmiarem warstwy ukrytej, a $\beta = 3$ rozmiarem inputu. Intuicja: możemy macierz $W_{\alpha\times\beta}$ zareprezentować jako wektor o długości $\alpha \times \beta$, lecz tracimy efektywność obliczeniową, bo chcemy w SGD

$$W^{\text{new}} = W^{\text{old}} - \eta \cdot \frac{\partial L}{\partial W^{\text{old}}}.$$

Okazuje się [źródło 1, operacja (5)], że możemy w naszej sieci neuronowej przedstawić ten gradient w formie macierzowej. Skrótowo: wychodzimy od zakładanej postaci $\frac{\partial L}{\partial W}$, i rozważamy jak powinien wyglądać element $\left(\frac{\partial L}{\partial W}\right)_{ij}$.

Jeżeli $z = Wx$, zaś $\delta_2 = \frac{\partial L}{\partial z}$, to

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial W} = \delta_2 \frac{\partial z}{\partial W}$$

możemy przedstawić jako *outer product* (oznaczony poprzez $\otimes$)

$$\frac{\partial L}{\partial W} = \delta_2^T \otimes x^T = \begin{bmatrix} \epsilon \cdot u_{11} \cdot \sigma'(z_1) \cdot x_1 & \cdots & \epsilon \cdot u_{11} \cdot \sigma'(z_1) \cdot x_3 \\ \vdots & \ddots & \vdots \\ \epsilon \cdot u_{41} \cdot \sigma'(z_1) \cdot x_1 & \cdots & \epsilon \cdot u_{41} \cdot \sigma'(z_1) \cdot x_3 \end{bmatrix} \tag{7}$$

$\frac{\partial L}{\partial U}$

podobnie jak względem macierzy W,

$$\frac{\partial L}{\partial U} = \delta_1 \frac{\partial \theta}{\partial U} = \delta_1^T \otimes h^T = \epsilon \cdot [h_1, h_2, h_3, h_4].$$

Gradienty względem b_1 oraz b_2

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial \theta}\frac{\partial \theta}{\partial b_2} = delta_1 \frac{\partial \theta}{\partial b_2} = \epsilon \in (-1, 1),$$

bo $b_2 \in R$.
Natomiast b_1 to wektor, i mamy

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial b_1} = \delta_2^T.$$

Transpozycja potrzebna, bo chcemy by kształt (*shape*) $\frac{\partial L}{\partial \Omega}$ był taki, jak kształt Ω. Kształt b_1 to (4, 1), zaś kształt δ_2 to (1, 4) - stąd potrzebna transpozycji.