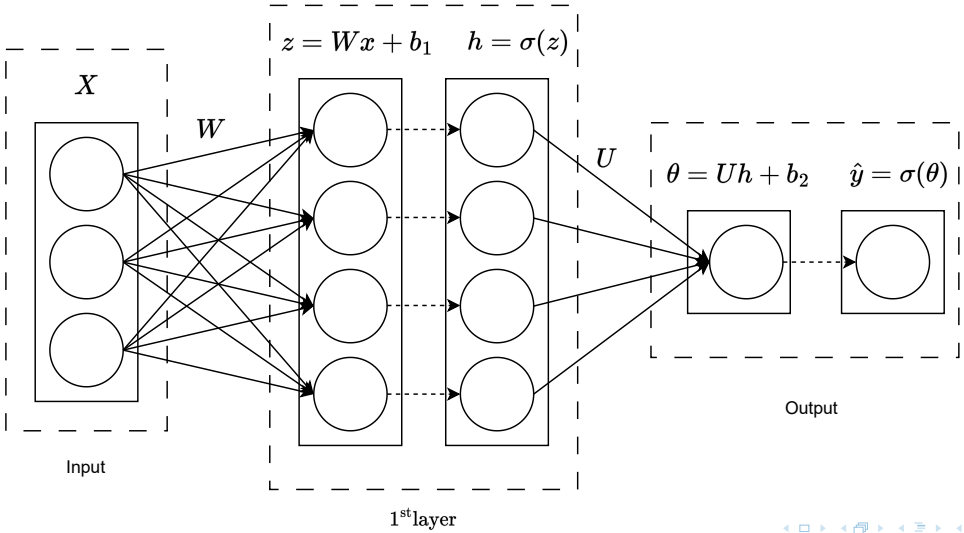
Liczenie gradientów w prostej sieci neuronowej z jedną warstwą ukrytą. Wersja 2 (zaktualizowana po zajęciach)

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 (mamy tylko dwie warstwy: ukrytą i output),
- 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 klasycznego algorytmu optymalizacyjnego 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 6, w szczególności 6.5 *Back-Propagation Computation in Fully Connected MLP*.

Potrzebujemy do SGD następujących gradientów:

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

Użyjemy reguły łańcuchowej (ang. *chain rule*), żeby je obliczyć.

Gradient δ_1

Liczymy gradienty wag z warstwy `output`. Zauważmy, że zawierają one wspólną część, którą nazwiemy δ_1.

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

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

$$\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \theta} = \frac{\partial L}{\partial \theta} = \delta_1 = \hat{y} - y. \tag{3}$$

Równanie (3) przyjmuje prostą postać ze względu na użytą funkcję aktywacji σ (wybraliśmy sigmoid) oraz funkcję straty $L = CE$.

Gradient δ_2

Licząc gradienty wag względem warstwy ukrytej *1st layer*: $\frac{\partial L}{\partial W}, \frac{\partial L}{\partial b_2}$ wystąpi część wspólna $\frac{\partial L}{\partial z}$.

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial \theta}\frac{\partial \theta}{\partial h}\frac{\partial h}{\partial z}\frac{\partial z}{\partial W}$$
$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial \theta}\frac{\partial \theta}{\partial h}\frac{\partial h}{\partial z}\frac{\partial z}{\partial b_2}$$

Oznaczmy część wspólna jako δ_2. Jej intuicyjne znaczenie: "(...) convert the gradient (...) into a gradient on the pre-nonlinearity activation." (źródło 2).

$$\frac{\partial L}{\partial z} = \delta_2 = \frac{\partial L}{\partial \theta}\frac{\partial \theta}{\partial h}\frac{\partial h}{\partial z}$$

Gradient δ_2 c.d.

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

gdzie [1] oraz [4] odnoszą się do operacji opisanych w źródle 1, a $\circ$ to element-wise multiplication. W naszej konkretnej sieci neuronowej, którą rozważamy:

$$\begin{aligned} \delta_2 &= \delta_1 \cdot [u_{11}, u_{21}, u_{31}, u_{41}] \circ \begin{bmatrix} \sigma'(z_1) \\ \sigma'(z_2) \\ \sigma'(z_3) \\ \sigma'(z_4) \end{bmatrix} \\ &= \delta_1 \cdot [u_{11} \cdot \sigma'(z_1), u_{21} \cdot \sigma'(z_2), u_{31} \cdot \sigma'(z_3), u_{41} \cdot \sigma'(z_4)], \end{aligned} \tag{5}$$

Gradient funkcji straty względem macierzy W

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

Przypomnienie: $z = Wx$. Jak policzyć gradient z względem macierzy W, żeby zachować efektywność zwektoryzowanych operacji np. w `numpy`?
Niech $\alpha = 4$ będzie rozmiarem warstwy ukrytej, a $\beta = 3$ rozmiarem inputu. Moglibyśmy macierz $W_{\alpha \times \beta}$ zareprezentować jako wektor o długości $\alpha \times \beta$, i policzyć gradient względem takiego wektora, lecz tracimy efektywność obliczeniową. Przypomnijmy, że w SGD

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

Chcemy więc by kształt (*shape*) $\frac{\partial L}{\partial W}$ był taki sam, jak kształt $W_{\alpha \times \beta}$.

Okazuje się [źródło 1, operacja (5)], że możemy w naszej sieci neuronowej przedstawić ten gradient w formie macierzowej. Skrótowo: wychodzimy od żądanej postaci $\frac{\partial L}{\partial W}$ (tj. α wierszy i β kolumn), i rozważamy jak powinien wyglądać element $\left(\frac{\partial L}{\partial W}\right)_{ij}$, gdzie $i = 1, \ldots, \alpha$, zaś $j = 1, \ldots, \beta$.
Okazuje się, że możemy przedstawić całe wyrażenie $\frac{\partial L}{\partial W} = \delta_2 \frac{\partial z}{\partial W}$ jako *outer product* (oznaczony poprzez $\otimes$)

$$\frac{\partial L}{\partial W} = \delta_2 \frac{\partial z}{\partial W} = \delta_2^T \otimes x^T = \begin{bmatrix} \delta_1 \cdot u_{11} \cdot \sigma'(z_1) \cdot x_1 & \cdots & \delta_1 \cdot u_{11} \cdot \sigma'(z_1) \cdot x_3 \\ \vdots & \ddots & \vdots \\ \delta_1 \cdot u_{41} \cdot \sigma'(z_1) \cdot x_1 & \cdots & \delta_1 \cdot u_{41} \cdot \sigma'(z_1) \cdot x_3 \end{bmatrix} \tag{6}$$

Rozumowanie jest analogiczne jak względem gradientu $\frac{\partial L}{\partial W}$. Otrzymamy

$$\frac{\partial L}{\partial U} = \delta_1 \frac{\partial \theta}{\partial U} = \delta_1^T \otimes h^T = \delta_1 \cdot [h_1, h_2, h_3, h_4]. \tag{7}$$

Gradienty względem b_1 oraz b_2

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial \theta}\frac{\partial \theta}{\partial b_2} = \delta_1 \frac{\partial \theta}{\partial b_2} = \delta_1 \in (-1, 1), \tag{8}$$

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

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial b_1} = \delta_2^T. \tag{9}$$

Transpozycja potrzebna, bo chcemy by kształt (*shape*) $\frac{\partial L}{\partial b_1}$ był taki, jak kształt b_1. Kształt b_1 to (4, 1), zaś kształt δ_2 to (1, 4) - stąd potrzeba transpozycji do obliczeń zwektoryzowanych np. w `numpy`.