Perceptron wielowarstwowy

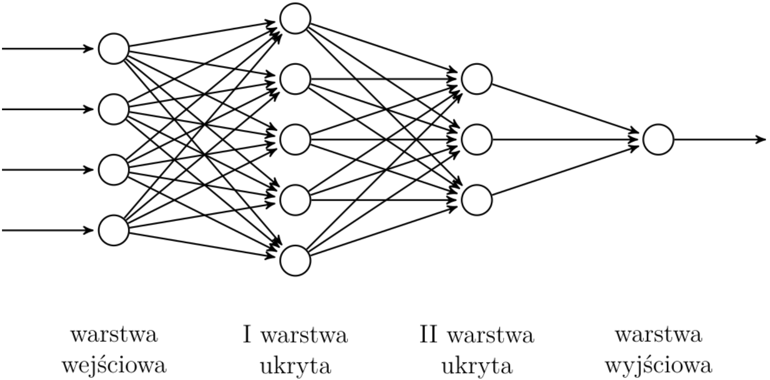
Metody sztucznej inteligencji - sieci neuronowe

Wydział Mechatroniki Politechniki Warszawskiej

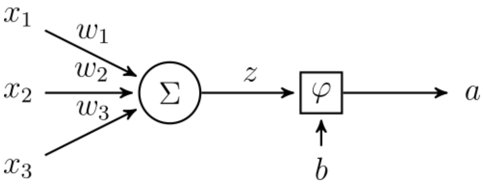
Anna Sztyber

Plan wykładu

Quo vadis?


warstwa
wejściowa
I warstwa
ukryta
II warstwa
ukryta
warstwa
wyjściowa

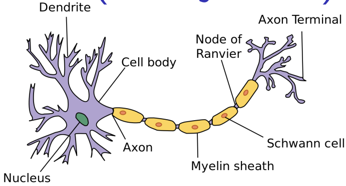
Perceptron



$$z = w_1 x_1 + \cdots + w_n x_n + b$$

$$a = \varphi(z)$$

Analogie biologiczne (raczej luźne)



- Biologiczny neuron jest układem o wielu wejściach (dendryty) i jednym wyjściu (akson)
- Sygnały przekazywane przez dendryty są wzmacniane lub osłabiane przez synapsy
- Gdy suma ważonych sygnałów przekracza wartość progową następuje tzw. zapłon neuronu, który objawia się wygenerowaniem serii impulsów, przekazywanych przez akson do innych neuronów
- Najcenniejszą cechą biologicznej sieci neuronowej jest zdolność do uczenia się, tzn. modyfikowania działanie na podstawie kolejnych doświadczeń.

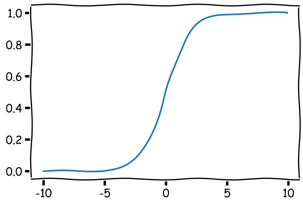
Plan wykładu

Funkcja sigmoidalna

$$\varphi(z) = \frac{1}{1 + \exp(-z)}$$

$$\varphi'(z) = \frac{1}{1 + \exp(-z)} \left(1 - \frac{1}{1 + \exp(-z)}\right)$$

$$\varphi'(z) = \varphi(z)(1 - \varphi(z))$$



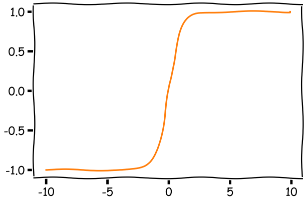
- jako wyjście dla klasyfikacji binarnej
- w praktyce rzadko stosowane w warstwach ukrytych
- pochodna równa 0 dla dużego zakresu wartości z

Tangens hiperboliczny

$$\varphi(z) = \mathrm{tgh}(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}$$

$$\varphi'(z) = (1 - \mathrm{tgh}(z)^2)$$

$$\varphi'(z) = (1 - \varphi(z)^2)$$

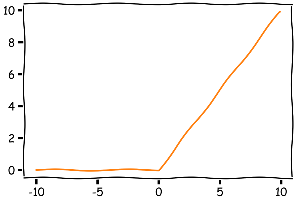


- wartości symetryczne względem 0
- w praktyce prawie zawsze działa lepiej niż funkcja sigmoidalna
- pochodna równa 0 dla dużego zakresu wartości z

Rectified linear unit (ReLU)

$$\varphi(z) = \max(0, z)$$

$$\varphi'(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z > 0 \\ ? & \text{if } z = 0 \end{cases}$$

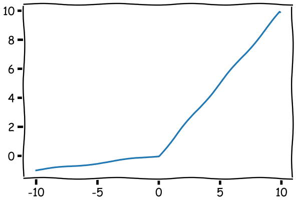


- pochodna nieciągła dla $z = 0$
- w praktyce można dla $z = 0$ wstawić 0 lub 1
- częściowo rozwiązuje problem z zerowaniem się pochodnej
- często stosowane

Leaky ReLU

$$\varphi(z) = \max(0.01z, z)$$

$$\varphi'(z) = \begin{cases} 0.01 & \text{if } z < 0 \\ 1 & \text{if } z > 0 \\ ? & \text{if } z = 0 \end{cases}$$



- pochodna nieciągła dla $z = 0$
- rozwiązuje problem z zerowaniem się pochodnej
- zamiast 0.01 można też zastosować inną małą liczbę

Aktywacja liniowa

$$\varphi(z) = z$$

$$\varphi'(z) = 1$$

- brak funkcji aktywacji
- stosowana jako warstwa wyjściowa w problemach regresji
- nie ma sensu stosowanie w warstwach ukrytych -

Aktywacja liniowa

$$\varphi(z) = z$$

$$\varphi'(z) = 1$$

- brak funkcji aktywacji
- stosowana jako warstwa wyjściowa w problemach regresji
- nie ma sensu stosowanie w warstwach ukrytych - złożenie funkcji liniowych jest funkcją liniową - nawet przy wielu warstwach nie otrzymamy nic ciekawego

Softmax

często stosowane jako warstwa wyjściowa dla problemów klasyfikacji

Dla M klas otrzymujemy z warstwy poprzedniej:

$$z_1, z_2, \ldots, z_M$$

Przekształcenie:

$$\sigma(z_i) = \frac{\exp(z_i)}{\sum_{i=1}^{M} \exp(z_i)}$$

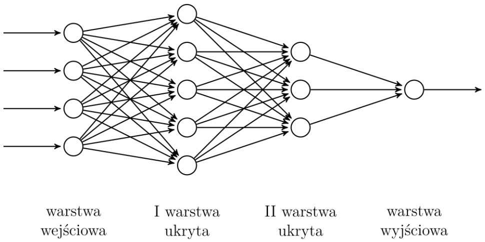
Normalizacja

- otrzymujemy liczby z zakresu (0,1) o sumie równiej 1
- możliwa interpretacja jako rozkład prawdopodobieństwa

Plan wykładu

Perceptron wielowarstwowy

MLP - Multi Layer Perceptron, Dense Network

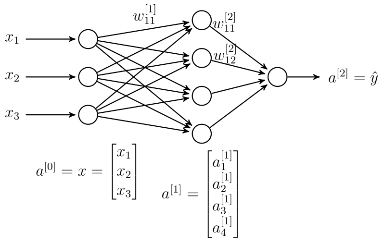


- warstwa wejściowa
- warstwy ukryte
- warstwa wyjściowa
- każdy neuron warstwy poprzedniej połączony ze wszystkimi neuronami warstwy następnej

Wektoryzacja

Na podstawie: Deeplearning.ai: Neural Networks and Deep Learning

Warstwa [1]:



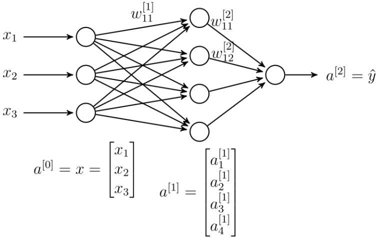
$$\begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} & w_{13}^{[1]} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ w_{41}^{[1]} & w_{42}^{[1]} & w_{43}^{[1]} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix}$$

$$z^{[1]} = W^{[1]}x + b^{[1]}$$

$$a^{[1]} = \varphi^{[1]}(z^{[1]})$$

Wektoryzacja

Na podstawie: Deeplearning.ai: Neural Networks and Deep Learning



Warstwa [2]:

$$z^{[2]} = W^{[2]} a^{[1]} + b^{[2]}$$

$$a^{[2]} = \varphi^{[2]}(z^{[2]})$$

Wektoryzacja dla wielu przykładów

- Chcemy wyznaczyć wyjście sieci dla m przykładów uczących
- notacja $[i](j)$: i - numer warstwy, (j) - numer przykładu

Dane wejściowe:

$$X = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(m)} \\ x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(m)} \\ x_3^{(1)} & x_3^{(2)} & \dots & x_3^{(m)} \end{bmatrix}$$

Warstwa [1]:

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$

$$A^{[1]} = \varphi(Z^{[1]})$$

$$Z^{[1]} = \begin{bmatrix} z_1^{[1](1)} & z_1^{[1](2)} & \dots & z_1^{[1](m)} \\ \vdots & \vdots & \vdots & \vdots \\ z_4^{[1](1)} & z_4^{[1](2)} & \dots & z_4^{[1](m)} \end{bmatrix}$$

Wektoryzacja dla wielu przykładów

- Chcemy wyznaczyć wyjście sieci dla m przykładów uczących
- notacja $[i](j)$: i - numer warstwy, (j) - numer przykładu

Dane wejściowe:

$$X = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(m)} \\ x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(m)} \\ x_3^{(1)} & x_3^{(2)} & \dots & x_3^{(m)} \end{bmatrix}$$

Warstwa [1]:

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$

$$A^{[1]} = \varphi(Z^{[1]})$$

$$A^{[1]} = \begin{bmatrix} a_1^{[1](1)} & a_1^{[1](2)} & \dots & a_1^{[1](m)} \\ \vdots & \vdots & \vdots & \vdots \\ a_4^{[1](1)} & a_4^{[1](2)} & \dots & a_4^{[1](m)} \end{bmatrix}$$

Wektoryzacja dla wielu przykładów

- Chcemy wyznaczyć wyjście sieci dla m przykładów uczących
- notacja $[i](j)$: i - numer warstwy, (j) - numer przykładu

Dane wejściowe:

$$X = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(m)} \\ x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(m)} \\ x_3^{(1)} & x_3^{(2)} & \dots & x_3^{(m)} \end{bmatrix}$$

Warstwa [2]:

$$Z^{[2]} = W^{[2]} A^{[1]} + b^{[2]}$$

$$A^{[2]} = \varphi(Z^{[2]})$$

Wyznaczanie wyjścia - podsumowanie

Dla sieci o dwóch warstwach

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$

$$A^{[1]} = \varphi(Z^{[1]})$$

$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$

$$A^{[2]} = \varphi(Z^{[2]})$$

$$\hat{y} = A^{[2]}$$

Plan wykładu

Funkcje kosztu

Błąd średniokwadratowy (MSE) - dla zadań regresji:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Entropia krzyżowa:

- dla dwóch klas (binary cross-entrophy) - dla klasyfikacji binarnej

$$-\frac{1}{m}\sum_{i=1}^{m}\left(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right)$$

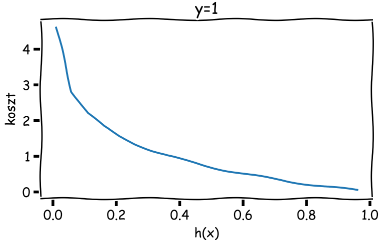
- dla wielu klas (categorical cross-entrophy)

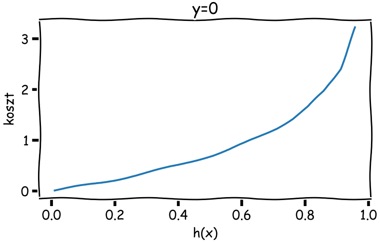
$$-\frac{1}{m}\sum_{i=1}^{m}\sum_{c=1}^{M} y_{ic}\log(\hat{y}_{ic})$$

Entropia krzyżowa - intuicja

Dla pojedynczego przykładu:

$$\text{koszt}(\hat{y}^{(i)}, y^{(i)}) = \begin{cases} -\log(\hat{y}^{(i)}) \text{ dla } y^{(i)} = 1 \\ -\log(1-\hat{y}^{(i)}) \text{ dla } y^{(i)} = 0 \end{cases}$$



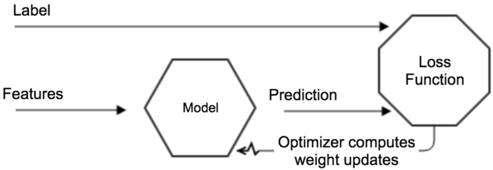


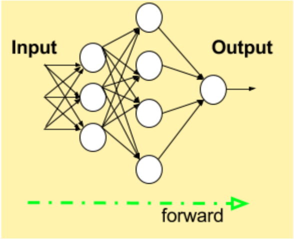
Sumaryczna funkcja kosztu:

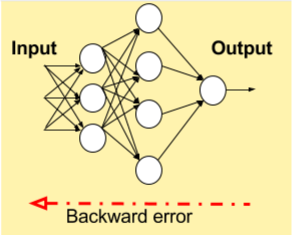
$$J(w) = -\frac{1}{m}\Sigma_{j=1}^{m}(y^{(i)}\log(\hat{y}^{(i)})) + (1-y^{(i)})\log(1-\hat{y}^{(i)}))$$

Plan wykładu

Algorytm wstecznej propagacji błędów







Algorytm wstecznej propagacji błędów

- forward: wyznaczamy wyjścia sieci i wartość funkcji kosztu J
- backward: chcemy wyznaczyć $\frac{\partial J}{\partial w_{kl}^{[i]}}$
  - pochodne w warstwie $[i]$ można wyznaczyć na podstawie pochodnych w warstwie $[i+1]$
  - obliczenia od warstwy ostatniej do pierwszej
  - chain rule - wzór na pochodną funkcji złożonej

Obecnie dostępne jest wiele narzędzi, które krok backward wykonują w sposób automatyczny

Narzędzia

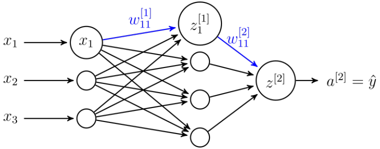
- Caffe2
- Microsoft Cognitive Toolkit (CNTK)
- Matlab Neural Network Toolbox
- MXNET
- PyTorch
- Tensorflow
- Keras
- ...

Algorytmy gradientowe

1. Przypisz wagom losowe wartości początkowe
2. Wyznacz wyjście sieci $\hat{y}$
3. Wyznacz wartość funkcji kosztu J
4. $$dW = \frac{\partial J}{\partial W} \qquad db = \frac{\partial J}{\partial b}$$
$$W = W - \alpha dW$$
$$b = b - \alpha db$$
5. dopóki nie warunek stopu idź do 2

Algorytm wstecznej propagacji błędów - przykład

Chcemy wyznaczyć $\frac{\partial J}{\partial w_{11}^{[1]}}$ (dla pojedynczego przykładu, pomijamy j)

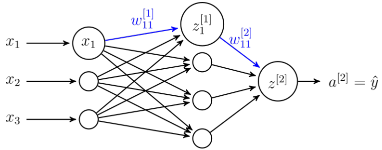


Dla klasyfikacji binarnej:

$$J = -(y \log(\hat{y}) + (1-y) \log(1-\hat{y}))$$

Algorytm wstecznej propagacji błędów - przykład

Chcemy wyznaczyć $\frac{\partial J}{\partial w_{11}^{[1]}}$ (dla pojedynczego przykładu, pomijamy j)



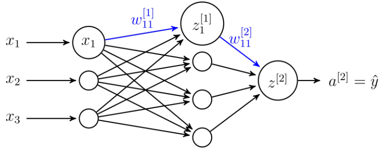
Dla klasyfikacji binarnej:

$$J = -(y \log(\hat{y}) + (1-y)\log(1-\hat{y}))$$

$$\frac{dJ}{\partial \hat{y}} = ?$$

Algorytm wstecznej propagacji błędów - przykład

Chcemy wyznaczyć $\frac{\partial J}{\partial w_{11}^{[1]}}$ (dla pojedynczego przykładu, pomijamy j)



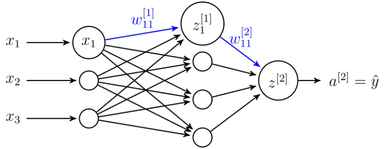
Dla klasyfikacji binarnej:

$$J = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y}))$$

$$\frac{dJ}{\partial \hat{y}} = -\frac{y}{\hat{y}} + \frac{1 - y}{1 - \hat{y}}$$

Algorytm wstecznej propagacji błędów - przykład

Chcemy wyznaczyć $\frac{\partial J}{\partial w_{11}^{[1]}}$ (dla pojedynczego przykładu, pomijamy j)



Dla klasyfikacji binarnej:

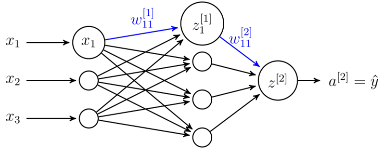
$$J = -(y\log(\hat{y}) + (1-y)\log(1-\hat{y}))$$

$$\frac{dJ}{\partial \hat{y}} = -\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}}$$

$$\frac{\partial J}{\partial z^{[2]}} = \frac{dJ}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^{[2]}}$$

Algorytm wstecznej propagacji błędów - przykład

Chcemy wyznaczyć $\frac{\partial J}{\partial w_{11}^{[1]}}$ (dla pojedynczego przykładu, pomijamy j)



Dla klasyfikacji binarnej:

$$J = -(y \log(\hat{y}) + (1-y) \log(1-\hat{y}))$$

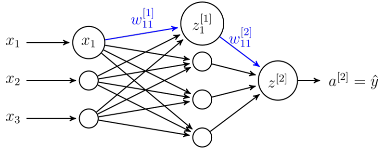
$$\frac{dJ}{\partial \hat{y}} = -\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}}$$

$$\frac{\partial J}{\partial z^{[2]}} = \frac{dJ}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^{[2]}}$$

$$\frac{\partial \hat{y}}{\partial z^{[2]}} = ?$$

Algorytm wstecznej propagacji błędów - przykład

Chcemy wyznaczyć $\frac{\partial J}{\partial w_{11}^{[1]}}$ (dla pojedynczego przykładu, pomijamy j)



$$\frac{\partial J}{\partial z^{[2]}} = \frac{dJ}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^{[2]}}$$

$$\frac{\partial \hat{y}}{\partial z^{[2]}} = ?$$

Dla klasyfikacji binarnej:

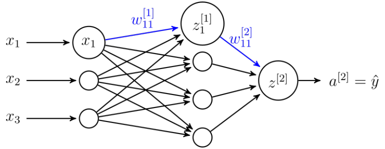
$$J = -(y \log(\hat{y}) + (1-y) \log(1-\hat{y}))$$

$$\hat{y} = a^{[2]} = \frac{1}{1 + \exp(-z^{[2]})}$$

$$\frac{dJ}{\partial \hat{y}} = -\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}}$$

Algorytm wstecznej propagacji błędów - przykład

Chcemy wyznaczyć $\frac{\partial J}{\partial w_{11}^{[1]}}$ (dla pojedynczego przykładu, pomijamy j)



$$\frac{\partial J}{\partial z^{[2]}} = \frac{dJ}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^{[2]}}$$

$$\frac{\partial \hat{y}}{\partial z^{[2]}} = ?$$

Dla klasyfikacji binarnej:

$$J = -(y \log(\hat{y}) + (1-y) \log(1-\hat{y}))$$

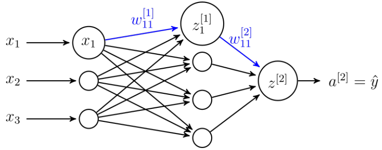
$$\hat{y} = a^{[2]} = \frac{1}{1 + \exp(-z^{[2]})}$$

$$\frac{dJ}{\partial \hat{y}} = -\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}}$$

$$\frac{\partial \hat{y}}{\partial z^{[2]}} = \hat{y}(1-\hat{y})$$

Algorytm wstecznej propagacji błędów - przykład

Chcemy wyznaczyć $\frac{\partial J}{\partial w_{11}^{[1]}}$ (dla pojedynczego przykładu, pomijamy j)



Dla klasyfikacji binarnej:

$$J = -(y \log(\hat{y}) + (1-y)\log(1-\hat{y}))$$

$$\frac{dJ}{\partial \hat{y}} = -\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}}$$

$$\frac{\partial J}{\partial z^{[2]}} = \frac{dJ}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^{[2]}}$$

$$\frac{\partial \hat{y}}{\partial z^{[2]}} = ?$$

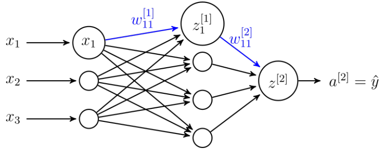
$$\hat{y} = a^{[2]} = \frac{1}{1+\exp(-z^{[2]})}$$

$$\frac{\partial \hat{y}}{\partial z^{[2]}} = \hat{y}(1-\hat{y})$$

$$\frac{\partial J}{\partial z^{[2]}} = \hat{y} - y$$

Algorytm wstecznej propagacji błędów - przykład

Chcemy wyznaczyć $\frac{\partial J}{\partial w_{11}^{[1]}}$ (dla pojedynczego przykładu, pomijamy j)

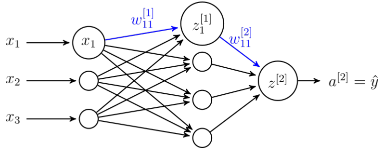


$$\frac{\partial J}{\partial w_1^{[2]}} = \frac{dJ}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial w_1^{[2]}}$$

$$\frac{\partial z^{[2]}}{\partial w_1^{[2]}} = ?$$

Algorytm wstecznej propagacji błędów - przykład

Chcemy wyznaczyć $\frac{\partial J}{\partial w_{11}^{[1]}}$ (dla pojedynczego przykładu, pomijamy j)



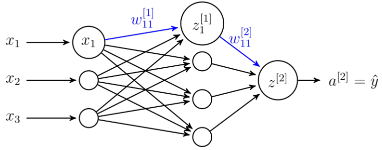
$$z^{[2]} = a_1^{[1]} w_1^{[2]} + \cdots + a_4^{[1]} w_4^{[2]} + b^{[2]}$$

$$\frac{\partial J}{\partial w_1^{[2]}} = \frac{dJ}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial w_1^{[2]}}$$

$$\frac{\partial z^{[2]}}{\partial w_1^{[2]}} = ?$$

Algorytm wstecznej propagacji błędów - przykład

Chcemy wyznaczyć $\frac{\partial J}{\partial w_{11}^{[1]}}$ (dla pojedynczego przykładu, pomijamy j)



$$z^{[2]} = a_1^{[1]} w_1^{[2]} + \cdots + a_4^{[1]} w_4^{[2]} + b^{[2]}$$

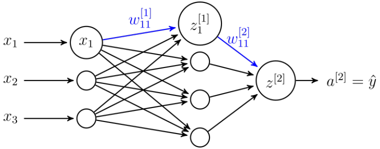
$$\frac{\partial J}{\partial w_1^{[2]}} = \frac{dJ}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial w_1^{[2]}}$$

$$\frac{\partial z^{[2]}}{\partial w_1^{[2]}} = a_1^{[1]}$$

$$\frac{\partial z^{[2]}}{\partial w_1^{[2]}} =?$$

Algorytm wstecznej propagacji błędów - przykład

Chcemy wyznaczyć $\frac{\partial J}{\partial w_{11}^{[1]}}$ (dla pojedynczego przykładu, pomijamy j)



$$z^{[2]} = a_1^{[1]} w_1^{[2]} + \cdots + a_4^{[1]} w_4^{[2]} + b^{[2]}$$

$$\frac{\partial J}{\partial w_1^{[2]}} = \frac{dJ}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial w_1^{[2]}}$$

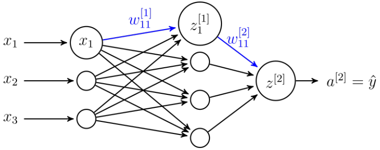
$$\frac{\partial z^{[2]}}{\partial w_1^{[2]}} = ?$$

$$\frac{\partial z^{[2]}}{\partial w_1^{[2]}} = a_1^{[1]}$$

$$\frac{\partial J}{\partial w_1^{[2]}} = \frac{dJ}{\partial z^{[2]}} a_1^{[1]}$$

Algorytm wstecznej propagacji błędów - przykład

Chcemy wyznaczyć $\frac{\partial J}{\partial w_{11}^{[1]}}$ (dla pojedynczego przykładu, pomijamy j)



$$z^{[2]} = a_1^{[1]} w_1^{[2]} + \cdots + a_4^{[1]} w_4^{[2]} + b^{[2]}$$

$$\frac{\partial z^{[2]}}{\partial w_1^{[2]}} = a_1^{[1]}$$

$$\frac{\partial J}{\partial w_1^{[2]}} = \frac{dJ}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial w_1^{[2]}}$$

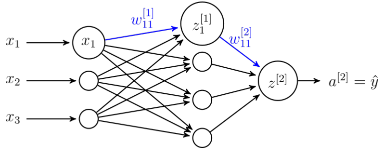
$$\frac{\partial J}{\partial w_1^{[2]}} = \frac{dJ}{\partial z^{[2]}} a_1^{[1]}$$

$$\frac{\partial z^{[2]}}{\partial w_1^{[2]}} = ?$$

$$\frac{\partial J}{\partial b} = \frac{dJ}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial b} = ?$$

Algorytm wstecznej propagacji błędów - przykład

Chcemy wyznaczyć $\frac{\partial J}{\partial w_{11}^{[1]}}$ (dla pojedynczego przykładu, pomijamy j)



$$z^{[2]} = a_1^{[1]} w_1^{[2]} + \cdots + a_4^{[1]} w_4^{[2]} + b^{[2]}$$

$$\frac{\partial z^{[2]}}{\partial w_1^{[2]}} = a_1^{[1]}$$

$$\frac{\partial J}{\partial w_1^{[2]}} = \frac{dJ}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial w_1^{[2]}}$$

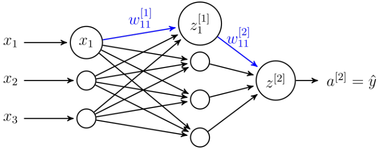
$$\frac{\partial J}{\partial w_1^{[2]}} = \frac{dJ}{\partial z^{[2]}} a_1^{[1]}$$

$$\frac{\partial z^{[2]}}{\partial w_1^{[2]}} = ?$$

$$\frac{\partial J}{\partial b} = \frac{dJ}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial b} = \frac{dJ}{\partial z^{[2]}}$$

Algorytm wstecznej propagacji błędów - przykład

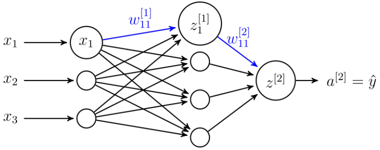
Chcemy wyznaczyć $\frac{\partial J}{\partial w_{11}^{[1]}}$ (dla pojedynczego przykładu, pomijamy j)



$$\frac{dJ}{\partial z_1^{[1]}} = \frac{dJ}{\partial a^{[1]}} \varphi'(z_1)$$

Algorytm wstecznej propagacji błędów - przykład

Chcemy wyznaczyć $\frac{\partial J}{\partial w_{11}^{[1]}}$ (dla pojedynczego przykładu, pomijamy j)

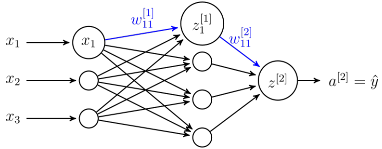


$$\frac{dJ}{\partial z_1^{[1]}} = \frac{dJ}{\partial a^{[1]}} \varphi'(z_1)$$

$$\frac{dJ}{\partial a^{[1]}} = \frac{dJ}{\partial z^{[2]}} w_1^{[2]}$$

Algorytm wstecznej propagacji błędów - przykład

Chcemy wyznaczyć $\frac{\partial J}{\partial w_{11}^{[1]}}$ (dla pojedynczego przykładu, pomijamy j)



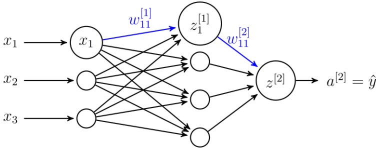
$$\frac{dJ}{\partial z_1^{[1]}} = \frac{dJ}{\partial a^{[1]}} \varphi'(z_1)$$

$$z_1^{[1]} = w_{11}^{[1]} x_1 + \cdots + w_{13}^{[1]} x_3 + b_1^{[1]}$$

$$\frac{dJ}{\partial a^{[1]}} = \frac{dJ}{\partial z^{[2]}} w_1^{[2]}$$

Algorytm wstecznej propagacji błędów - przykład

Chcemy wyznaczyć $\frac{\partial J}{\partial w_{11}^{[1]}}$ (dla pojedynczego przykładu, pomijamy j)



$$z_1^{[1]} = w_{11}^{[1]} x_1 + \cdots + w_{13}^{[1]} x_3 + b_1^{[1]}$$

$$\frac{\partial J}{\partial w_{11}^{[1]}} = \frac{dJ}{\partial z_1^{[1]}} x_1$$

$$\frac{\partial J}{\partial b_1^{[1]}} = \frac{dJ}{\partial z_1^{[1]}}$$

$$\frac{dJ}{\partial z_1^{[1]}} = \frac{dJ}{\partial a^{[1]}} \varphi'(z_1)$$

$$\frac{dJ}{\partial a^{[1]}} = \frac{dJ}{\partial z^{[2]}} w_1^{[2]}$$

Przykład

$$\frac{\partial J}{\partial z^{[2]}} = \hat{y} - y$$

$$\frac{\partial J}{\partial w_1^{[2]}} = \frac{dJ}{\partial z^{[2]}} a_1^{[1]}$$

$$\frac{\partial J}{\partial b^{[2]}} = \frac{dJ}{\partial z^{[2]}}$$

$$\frac{dJ}{\partial z_1^{[1]}} = \frac{dJ}{\partial z^{[2]}} w_1^{[2]} \varphi^{[1]\prime}(z_1)$$

$$\frac{\partial J}{\partial w_{11}^{[1]}} = \frac{dJ}{\partial z_1^{[1]}} x_1$$

$$\frac{\partial J}{\partial b_1^{[1]}} = \frac{dJ}{\partial z_1^{[1]}}$$

Wektoryzacja

$$\frac{\partial J}{\partial Z^{[2]}} = A^{[2]} - Y$$

$$\frac{\partial J}{\partial W^{[2]}} = \frac{1}{m} \frac{dJ}{\partial Z^{[2]}} A^{[1]T}$$

$$\frac{\partial J}{\partial b^{[2]}} = \frac{1}{m} np.sum(\frac{dJ}{\partial Z^{[2]}}, axis = 1)$$

$$\frac{dJ}{\partial Z^{[1]}} = W^{[2]T} \frac{dJ}{\partial Z^{[2]}} * \varphi^{[1]\prime}(z_1)$$

$$\frac{\partial J}{\partial w^{[1]}} = \frac{dJ}{\partial Z^{[1]}} X^T$$

$$\frac{\partial J}{\partial b^{[1]}} = \frac{1}{m} np.sum(\frac{dJ}{\partial Z^{[1]}}, axis = 1)$$

Algorytmy optymalizacji

Metoda gradientu prostego

- gradient descent - w każdym kroku wyznaczamy wyjścia dla wszystkich przykładów ze zbioru uczącego
- stochastic gradient descent - dla jednego przykładu ze zbioru uczącego
- batch gradient descent - dla n przykładów ze zbioru uczącego

Problemy:

- oscylacje
- zanikające lub wybuchające gradienty

Modyfikacje:

- momentum [Qian, 1999]
- rmsprop [Hinton et al.,]
- Adam [Kingma and Ba, 2014]
- ...

Inicjalizacja wag

Nie można ustawić wszystkich wartości początkowych na 0

Inicjalizacja wag

Nie można ustawić wszystkich wartości początkowych na 0

1. symetria
2. aktywacje wszystkich neuronów na początku są takie same
3. wszystkie gradienty są takie same
4. wagi już zawsze będą takie same

Metoda postępowania:

1. inicjalizacja w małymi wartościami losowymi
2. inicjalizacja b zerami (w są różne, nie ma już problemu symetrii)

Dobór parametrów i hiper-parametrów sieci

Parametry

- w, b
- dobierane za pomocą algorytmu uczenia

Hiper-parametry

- liczba warstw
- liczba neuronów w warstwach
- liczba iteracji algorytmu gradientowego
- funkcje aktywacji
- metoda regularyzacji (o tym później)
- i wiele innych
- dobieramy z wykorzystaniem wyników na zbiorze walidacyjnym

Uwagi praktyczne

- sieci nie mają właściwości ekstrapolacyjnych
- dane uczące powinny być reprezentatywne dla całego zakresu zmienności wejść
- do budowy sieci o wielu parametrach potrzebujemy dużo danych (wyjątek: powtórne wykorzystanie sieci przeznaczonej do rozwiązywania podobnego problemu, transfer learning)
- modelowane zjawiska (np proces przemysłowy) zmieniają się w czasie - model będzie tracił dokładność

Literatura I

Goodfellow, I., Bengio, Y., and Courville, A. (2016).
Deep Learning.
MIT Press.
http://www.deeplearningbook.org.

Gulli, A.
Deep learning 101 - demystifying deep learning and machine learning with tensorflow, keras, tflearn - antonio gulli.
https://docs.google.com/presentation/d/1xd65LC8BV4jazwQEg672J-v2E5mFWpGnu0bcb6tFKyQ/edit#slide=id.p.

Hinton, G., Srivastava, N., and Swersky, K.
Overview of mini-batch gradient descent.
Neural Networks for Machine Learning,
http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

Kingma, D. P. and Ba, J. (2014).
Adam: A method for stochastic optimization.
CoRR, abs/1412.6980.

Qian, N. (1999).
On the momentum term in gradient descent learning algorithms.
Neural Networks, 12(1):145 – 151.