

Sieci MLP (Multi Layer Perceptron)

Przemysław Głomb

Instytut Informatyki Teoretycznej i Stosowanej Polskiej Akademii Nauk

redaktor pomocniczy: Bartłomiej Gardas



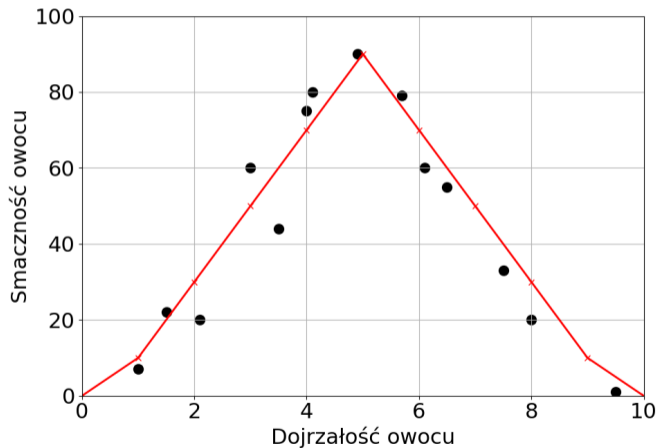
Uczenie / trening sieci



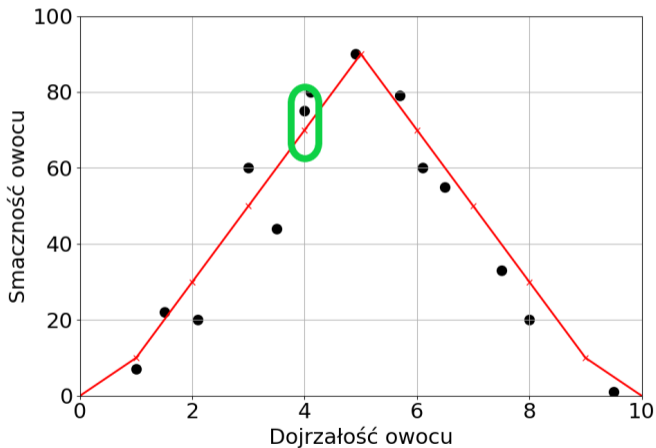
Backpropagation

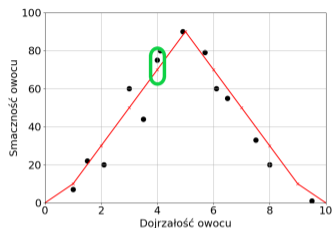


$$y_{11} = \max(20x - 10, 0) \quad y_{12} = \max(40x - 200, 0)$$
$$y_{21} = \max(y_{11} - y_{12}, 0)$$



$$y_{11} = \max(20x - 10, 0) \quad y_{12} = \max(40x - 200, 0)$$
$$y_{21} = \max(y_{11} - y_{12}, 0)$$





$$(x = 4, y^{\text{true}} = 75)$$

$$y_{11} = \max(20 \times 4 - 10, 0) = \max(70, 0) = 70$$

$$w_{111} = 20, b_{11} = -10$$

$$y_{12} = \max(40 \times 4 - 200, 0) = \max(-40, 0) = 0$$

$$w_{121} = 40, b_{12} = -200$$

$$y_{21} = \max(70 - 0, 0) = 70$$

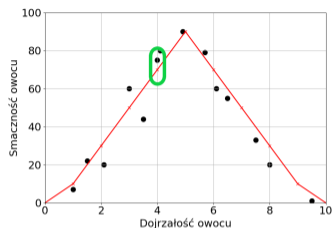
$$w_{211} = 1, w_{212} = -1, b_{21} = 0$$

$$\text{błąd (loss)} (75 - 70)^2 = 25$$

$$L(y^{\text{true}}, y^{\text{out}}) = (y^{\text{true}} - y_{21})^2$$

$$\text{błąd} \sim w_{ijk}?$$





$$(x = 4, y^{\text{true}} = 75)$$

$$y_{11} = \max(20 \times 4 - 10, 0) = \max(70, 0) = 70$$

$$y_{12} = \max(40 \times 4 - 200, 0) = \max(-40, 0) = 0$$

$$y_{21} = \max(70 - 0, 0) = 70$$

$$\text{błąd (loss)} (75 - 70)^2 = 25$$

$$\text{błąd} \sim w_{ijk}?$$

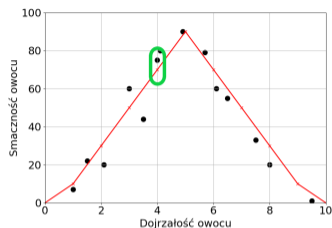
$$w_{111} = 20, b_{11} = -10$$

$$w_{121} = 40, b_{12} = -200$$

$$w_{211} = 1, w_{212} = -1, b_{21} = 0$$

$$L(y^{\text{true}}, y^{\text{out}}) = (y^{\text{true}} - y_{21})^2$$





$$(x = 4, y^{\text{true}} = 75)$$

$$y_{11} = \max(20 \times 4 - 10, 0) = \max(70, 0) = 70$$

$$w_{111} = 20, b_{11} = -10$$

$$y_{12} = \max(40 \times 4 - 200, 0) = \max(-40, 0) = 0$$

$$w_{121} = 40, b_{12} = -200$$

$$y_{21} = \max(70 - 0, 0) = 70$$

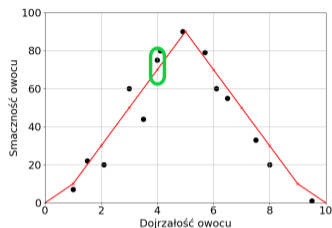
$$w_{211} = 1, w_{212} = -1, b_{21} = 0$$

$$\text{błąd (loss)} (75 - 70)^2 = 25$$

$$L(y^{\text{true}}, y^{\text{out}}) = (y^{\text{true}} - y_{21})^2$$

$$\text{błąd} \sim w_{ijk}?$$





$$(x = 4, y^{\text{true}} = 75)$$

$$y_{11} = \max(20 \times 4 - 10, 0) = \max(70, 0) = 70$$

$$y_{12} = \max(40 \times 4 - 200, 0) = \max(-40, 0) = 0$$

$$y_{21} = \max(70 - 0, 0) = 70$$

$$\text{błąd (loss)} (75 - 70)^2 = 25$$

$$\text{błąd} \sim w_{ijk}?$$

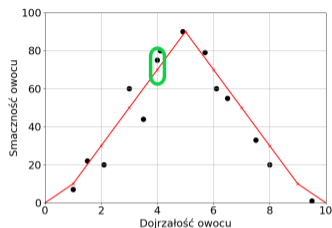
$$w_{111} = 20, b_{11} = -10$$

$$w_{121} = 40, b_{12} = -200$$

$$w_{211} = 1, w_{212} = -1, b_{21} = 0$$

$$L(y^{\text{true}}, y^{\text{out}}) = (y^{\text{true}} - y_{21})^2$$





$$(x = 4, y^{\text{true}} = 75)$$

$$y_{11} = \max(20 \times 4 - 10, 0) = \max(70, 0) = 70$$

$$y_{12} = \max(40 \times 4 - 200, 0) = \max(-40, 0) = 0$$

$$y_{21} = \max(70 - 0, 0) = 70$$

$$\text{błąd (loss)} (75 - 70)^2 = 25$$

$$\text{błąd} \sim w_{ijk}?$$

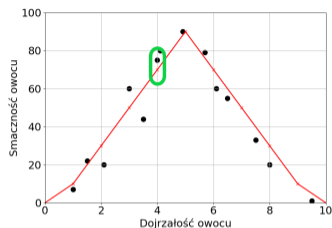
$$w_{111} = 20, b_{11} = -10$$

$$w_{121} = 40, b_{12} = -200$$

$$w_{211} = 1, w_{212} = -1, b_{21} = 0$$

$$L(y^{\text{true}}, y^{\text{out}}) = (y^{\text{true}} - y_{21})^2$$





$$(x = 4, y^{\text{true}} = 75)$$

$$y_{11} = \max(20 \times 4 - 10, 0) = \max(70, 0) = 70$$

$$y_{12} = \max(40 \times 4 - 200, 0) = \max(-40, 0) = 0$$

$$y_{21} = \max(70 - 0, 0) = 70$$

$$\text{błąd (loss)} (75 - 70)^2 = 25$$

$$\text{błąd} \sim w_{ijk}?$$

$$w_{111} = 20, b_{11} = -10$$

$$w_{121} = 40, b_{12} = -200$$

$$w_{211} = 1, w_{212} = -1, b_{21} = 0$$

$$L(y^{\text{true}}, y^{\text{out}}) = (y^{\text{true}} - y_{21})^2$$



$$y_{11} = \max(20x - 10, 0) \quad y_{12} = \max(40x - 200, 0) \quad y_{21} = \max(y_{11} - y_{12}, 0)$$

$$y_{11} = \max(z_{11}, 0) \quad z_{11} = w_{111}x + b_{11}$$

$$y_{12} = \max(z_{12}, 0) \quad z_{12} = w_{121}x + b_{12}$$

$$y_{21} = \max(z_{21}, 0) \quad z_{21} = w_{211}y_{11} + w_{212}y_{12} + b_{21}$$

$$\frac{\partial L}{\partial y_{21}} = \frac{\partial}{\partial y_{21}} (y^{\text{true}^2} - 2y^{\text{true}}y_{21} + y_{21}^2) = 2y_{21} - 2y^{\text{true}}$$

$$\frac{\partial L}{\partial z_{21}} = \frac{\partial L}{\partial y_{21}} \frac{\partial y_{21}}{\partial z_{21}} \quad \frac{\partial y_{21}}{\partial z_{21}} = \{0 \text{ if } z_{21} < 0; 1 \text{ if } z_{21} > 0\}$$

(Reguła łańcuchowa $\frac{\partial L}{\partial w} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial z} \frac{\partial z}{\partial w}$)

$$\frac{\partial L}{\partial w_{211}} = \frac{\partial L}{\partial y_{21}} \frac{\partial y_{21}}{\partial z_{21}} \frac{\partial z_{21}}{\partial w_{211}} \quad \frac{\partial z_{21}}{\partial w_{211}} = y_{11}$$

...

$$\frac{\partial L}{\partial w_{ijk}} \text{ (zależność błędu od wagi)}$$

$$\Delta w_{ijk} = -\gamma \frac{\partial L}{\partial w_{ijk}} \text{ (aktualizacja wagi)}$$



$$y_{11} = \max(20x - 10, 0) \quad y_{12} = \max(40x - 200, 0) \quad y_{21} = \max(y_{11} - y_{12}, 0)$$

$$y_{11} = \max(z_{11}, 0) \quad z_{11} = w_{111}x + b_{11}$$

$$y_{12} = \max(z_{12}, 0) \quad z_{12} = w_{121}x + b_{12}$$

$$y_{21} = \max(z_{21}, 0) \quad z_{21} = w_{211}y_{11} + w_{212}y_{12} + b_{21}$$

$$\frac{\partial L}{\partial y_{21}} = \frac{\partial}{\partial y_{21}} (y^{\text{true}^2} - 2y^{\text{true}}y_{21} + y_{21}^2) = 2y_{21} - 2y^{\text{true}}$$

$$\frac{\partial L}{\partial z_{21}} = \frac{\partial L}{\partial y_{21}} \frac{\partial y_{21}}{\partial z_{21}} \quad \frac{\partial y_{21}}{\partial z_{21}} = \{0 \text{ if } z_{21} < 0; 1 \text{ if } z_{21} > 0\}$$

(Reguła łańcuchowa $\frac{\partial L}{\partial w} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial z} \frac{\partial z}{\partial w}$)

$$\frac{\partial L}{\partial w_{211}} = \frac{\partial L}{\partial y_{21}} \frac{\partial y_{21}}{\partial z_{21}} \frac{\partial z_{21}}{\partial w_{211}} \quad \frac{\partial z_{21}}{\partial w_{211}} = y_{11}$$

...

$$\frac{\partial L}{\partial w_{ijk}} \text{ (zależność błędu od wagi)}$$

$$\Delta w_{ijk} = -\gamma \frac{\partial L}{\partial w_{ijk}} \text{ (aktualizacja wagi)}$$



$$y_{11} = \max(20x - 10, 0) \quad y_{12} = \max(40x - 200, 0) \quad y_{21} = \max(y_{11} - y_{12}, 0)$$

$$y_{11} = \max(z_{11}, 0) \quad z_{11} = w_{111}x + b_{11}$$

$$y_{12} = \max(z_{12}, 0) \quad z_{12} = w_{121}x + b_{12}$$

$$y_{21} = \max(z_{21}, 0) \quad z_{21} = w_{211}y_{11} + w_{212}y_{12} + b_{21}$$

$$\frac{\partial L}{\partial y_{21}} = \frac{\partial}{\partial y_{21}} (y^{\text{true}^2} - 2y^{\text{true}}y_{21} + y_{21}^2) = 2y_{21} - 2y^{\text{true}}$$

$$\frac{\partial L}{\partial z_{21}} = \frac{\partial L}{\partial y_{21}} \frac{\partial y_{21}}{\partial z_{21}} \quad \frac{\partial y_{21}}{\partial z_{21}} = \{0 \text{ if } z_{21} < 0; 1 \text{ if } z_{21} > 0\}$$

(Reguła łańcuchowa $\frac{\partial L}{\partial w} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial z} \frac{\partial z}{\partial w}$)

$$\frac{\partial L}{\partial w_{211}} = \frac{\partial L}{\partial y_{21}} \frac{\partial y_{21}}{\partial z_{21}} \frac{\partial z_{21}}{\partial w_{211}} \quad \frac{\partial z_{21}}{\partial w_{211}} = y_{11}$$

...

$$\frac{\partial L}{\partial w_{ijk}} \text{ (zależność błędu od wagi)}$$

$$\Delta w_{ijk} = -\gamma \frac{\partial L}{\partial w_{ijk}} \text{ (aktualizacja wagi)}$$



$$y_{11} = \max(20x - 10, 0) \quad y_{12} = \max(40x - 200, 0) \quad y_{21} = \max(y_{11} - y_{12}, 0)$$

$$y_{11} = \max(z_{11}, 0) \quad z_{11} = w_{111}x + b_{11}$$

$$y_{12} = \max(z_{12}, 0) \quad z_{12} = w_{121}x + b_{12}$$

$$y_{21} = \max(z_{21}, 0) \quad z_{21} = w_{211}y_{11} + w_{212}y_{12} + b_{21}$$

$$\frac{\partial L}{\partial y_{21}} = \frac{\partial}{\partial y_{21}} (y^{\text{true}^2} - 2y^{\text{true}}y_{21} + y_{21}^2) = 2y_{21} - 2y^{\text{true}}$$

$$\frac{\partial L}{\partial z_{21}} = \frac{\partial L}{\partial y_{21}} \frac{\partial y_{21}}{\partial z_{21}} \quad \frac{\partial y_{21}}{\partial z_{21}} = \{0 \text{ if } z_{21} < 0; 1 \text{ if } z_{21} > 0\}$$

(Reguła łańcuchowa $\frac{\partial L}{\partial w} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial z} \frac{\partial z}{\partial w}$)

$$\frac{\partial L}{\partial w_{211}} = \frac{\partial L}{\partial y_{21}} \frac{\partial y_{21}}{\partial z_{21}} \frac{\partial z_{21}}{\partial w_{211}} \quad \frac{\partial z_{21}}{\partial w_{211}} = y_{11}$$

...

$$\frac{\partial L}{\partial w_{ijk}} \text{ (zależność błędu od wagi)}$$

$$\Delta w_{ijk} = -\gamma \frac{\partial L}{\partial w_{ijk}} \text{ (aktualizacja wagi)}$$



$$y_{11} = \max(20x - 10, 0) \quad y_{12} = \max(40x - 200, 0) \quad y_{21} = \max(y_{11} - y_{12}, 0)$$

$$y_{11} = \max(z_{11}, 0) \quad z_{11} = w_{111}x + b_{11}$$

$$y_{12} = \max(z_{12}, 0) \quad z_{12} = w_{121}x + b_{12}$$

$$y_{21} = \max(z_{21}, 0) \quad z_{21} = w_{211}y_{11} + w_{212}y_{12} + b_{21}$$

$$\frac{\partial L}{\partial y_{21}} = \frac{\partial}{\partial y_{21}} (y^{\text{true}^2} - 2y^{\text{true}}y_{21} + y_{21}^2) = 2y_{21} - 2y^{\text{true}}$$

$$\frac{\partial L}{\partial z_{21}} = \frac{\partial L}{\partial y_{21}} \frac{\partial y_{21}}{\partial z_{21}} \quad \frac{\partial y_{21}}{\partial z_{21}} = \{0 \text{ if } z_{21} < 0; 1 \text{ if } z_{21} > 0\}$$

(Reguła łańcuchowa $\frac{\partial L}{\partial w} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial z} \frac{\partial z}{\partial w}$)

$$\frac{\partial L}{\partial w_{211}} = \frac{\partial L}{\partial y_{21}} \frac{\partial y_{21}}{\partial z_{21}} \frac{\partial z_{21}}{\partial w_{211}} \quad \frac{\partial z_{21}}{\partial w_{211}} = y_{11}$$

...

$$\frac{\partial L}{\partial w_{ijk}} \text{ (zależność błędu od wagi)}$$

$$\Delta w_{ijk} = -\gamma \frac{\partial L}{\partial w_{ijk}} \text{ (aktualizacja wagi)}$$



$$y_{11} = \max(20x - 10, 0) \quad y_{12} = \max(40x - 200, 0) \quad y_{21} = \max(y_{11} - y_{12}, 0)$$

$$y_{11} = \max(z_{11}, 0) \quad z_{11} = w_{111}x + b_{11}$$

$$y_{12} = \max(z_{12}, 0) \quad z_{12} = w_{121}x + b_{12}$$

$$y_{21} = \max(z_{21}, 0) \quad z_{21} = w_{211}y_{11} + w_{212}y_{12} + b_{21}$$

$$\frac{\partial L}{\partial y_{21}} = \frac{\partial}{\partial y_{21}} (y^{\text{true}^2} - 2y^{\text{true}}y_{21} + y_{21}^2) = 2y_{21} - 2y^{\text{true}}$$

$$\frac{\partial L}{\partial z_{21}} = \frac{\partial L}{\partial y_{21}} \frac{\partial y_{21}}{\partial z_{21}} \quad \frac{\partial y_{21}}{\partial z_{21}} = \{0 \text{ if } z_{21} < 0; 1 \text{ if } z_{21} > 0\}$$

(Reguła łańcuchowa $\frac{\partial L}{\partial w} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial z} \frac{\partial z}{\partial w}$)

$$\frac{\partial L}{\partial w_{211}} = \frac{\partial L}{\partial y_{21}} \frac{\partial y_{21}}{\partial z_{21}} \frac{\partial z_{21}}{\partial w_{211}} \quad \frac{\partial z_{21}}{\partial w_{211}} = y_{11}$$

...

$$\frac{\partial L}{\partial w_{ijk}} \text{ (zależność błędu od wagi)}$$

$$\Delta w_{ijk} = -\gamma \frac{\partial L}{\partial w_{ijk}} \text{ (aktualizacja wagi)}$$



$$y_{11} = \max(20x - 10, 0) \quad y_{12} = \max(40x - 200, 0) \quad y_{21} = \max(y_{11} - y_{12}, 0)$$

$$y_{11} = \max(z_{11}, 0) \quad z_{11} = w_{111}x + b_{11}$$

$$y_{12} = \max(z_{12}, 0) \quad z_{12} = w_{121}x + b_{12}$$

$$y_{21} = \max(z_{21}, 0) \quad z_{21} = w_{211}y_{11} + w_{212}y_{12} + b_{21}$$

$$\frac{\partial L}{\partial y_{21}} = \frac{\partial}{\partial y_{21}} (y^{\text{true}^2} - 2y^{\text{true}}y_{21} + y_{21}^2) = 2y_{21} - 2y^{\text{true}}$$

$$\frac{\partial L}{\partial z_{21}} = \frac{\partial L}{\partial y_{21}} \frac{\partial y_{21}}{\partial z_{21}} \quad \frac{\partial y_{21}}{\partial z_{21}} = \{0 \text{ if } z_{21} < 0; 1 \text{ if } z_{21} > 0\}$$

(Reguła łańcuchowa $\frac{\partial L}{\partial w} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial z} \frac{\partial z}{\partial w}$)

$$\frac{\partial L}{\partial w_{211}} = \frac{\partial L}{\partial y_{21}} \frac{\partial y_{21}}{\partial z_{21}} \frac{\partial z_{21}}{\partial w_{211}} \quad \frac{\partial z_{21}}{\partial w_{211}} = y_{11}$$

...

$$\frac{\partial L}{\partial w_{ijk}} \text{ (zależność błędu od wagi)}$$

$$\Delta w_{ijk} = -\gamma \frac{\partial L}{\partial w_{ijk}} \text{ (aktualizacja wagi)}$$



$$y_{11} = \max(20x - 10, 0) \quad y_{12} = \max(40x - 200, 0) \quad y_{21} = \max(y_{11} - y_{12}, 0)$$

$$y_{11} = \max(z_{11}, 0) \quad z_{11} = w_{111}x + b_{11}$$

$$y_{12} = \max(z_{12}, 0) \quad z_{12} = w_{121}x + b_{12}$$

$$y_{21} = \max(z_{21}, 0) \quad z_{21} = w_{211}y_{11} + w_{212}y_{12} + b_{21}$$

$$\frac{\partial L}{\partial y_{21}} = \frac{\partial}{\partial y_{21}} (y^{\text{true}^2} - 2y^{\text{true}}y_{21} + y_{21}^2) = 2y_{21} - 2y^{\text{true}}$$

$$\frac{\partial L}{\partial z_{21}} = \frac{\partial L}{\partial y_{21}} \frac{\partial y_{21}}{\partial z_{21}} \quad \frac{\partial y_{21}}{\partial z_{21}} = \{0 \text{ if } z_{21} < 0; 1 \text{ if } z_{21} > 0\}$$

(Reguła łańcuchowa $\frac{\partial L}{\partial w} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial z} \frac{\partial z}{\partial w}$)

$$\frac{\partial L}{\partial w_{211}} = \frac{\partial L}{\partial y_{21}} \frac{\partial y_{21}}{\partial z_{21}} \frac{\partial z_{21}}{\partial w_{211}} \quad \frac{\partial z_{21}}{\partial w_{211}} = y_{11}$$

...

$$\frac{\partial L}{\partial w_{ijk}} \text{ (zależność błędu od wagi)}$$

$$\Delta w_{ijk} = -\gamma \frac{\partial L}{\partial w_{ijk}} \text{ (aktualizacja wagi)}$$



1. Sieć neuronowa: wagi (+bias, architektura)
2. Przykład (sample) i wartość wymagana dla niego (ground truth)
3. Wyliczenie wartości częściowych i końcowych dla przykładu (forward pass)
4. Miara błędu (loss)
5. Wsteczna propagacja błędu (backpropagation) – wyznaczenie gradientów
6. Aktualizacja wag sieci



[Stochastic] gradient descent



Mamy:

1. Adaptacja (aktualizacja) wag dla konkretnego przykładu (punktu danych)

Dalsze zagadnienia:

1. Wiele punktów danych – optymalizacja
2. Model danych i skuteczna predykcja z modelu
3. Współczynnik szybkości uczenia (learning rate), zanikające i eksplodujące gradienty, zagadnienia optymalizacji, inicjalizacja, ...



Mamy:

1. Adaptacja (aktualizacja) wag dla konkretnego przykładu (punktu danych)

Dalsze zagadnienia:

1. Wiele punktów danych – optymalizacja
2. Model danych i skuteczna predykcja z modelu
3. Współczynnik szybkości uczenia (learning rate), zanikające i eksplodujące gradienty, zagadnienia optymalizacji, inicjalizacja, ...



„Gradient prosty” (gradient descent):

1. Mamy $\frac{\partial L}{\partial w_{ijk}}$
2. Minimalizujemy L – błąd zmniejsza się najszybciej jeżeli idziemy w kierunku przeciwnym do gradientu
3. Iteracyjnie modyfikujemy wagi $w_{ijk}^{n+1} = w_{ijk}^n - \gamma \frac{\partial L}{\partial w_{ijk}}$
4. Zejdziemy wzdłuż gradientu do zera i problem rozwiązany? Niestety, bardziej skomplikowane. . .
5. Na początek – jak wyliczyć gradient – mamy $m = 12$ przykładów

●●●●●●●●●●●●	(wykorzystujemy wszystkie)	batch gradient descent
○●○○○○○○○○○○	(wykorzystujemy jeden)	stochastic gradient descent
○○○○●●●●○○○○	(wykorzystujemy podzbiór)	minibatch gradient descent



„Gradient prosty” (gradient descent):

1. Mamy $\frac{\partial L}{\partial w_{ijk}}$
2. Minimalizujemy L – błąd zmniejsza się najszybciej jeżeli idziemy w kierunku przeciwnym do gradientu
3. Iteracyjnie modyfikujemy wagi $w_{ijk}^{n+1} = w_{ijk}^n - \gamma \frac{\partial L}{\partial w_{ijk}}$
4. Zejdziemy wzdłuż gradientu do zera i problem rozwiązany? Niestety, bardziej skomplikowane. . .
5. Na początek – jak wyliczyć gradient – mamy $m = 12$ przykładów

●●●●●●●●●●●●	(wykorzystujemy wszystkie)	batch gradient descent
○●○○○○○○○○○○	(wykorzystujemy jeden)	stochastic gradient descent
○○○○●●●●○○○○	(wykorzystujemy podzbiór)	minibatch gradient descent



1. Gradient descent – iteracyjne poprawianie sieci
 - 1.1 **batch gradient descent** – aktualizacja modelu dopiero po przejściu przez wszystkie przykłady treningowe – kosztowne obliczeniowo i pamięciowo
 - 1.2 **stochastic gradient descent** – aktualizacja modelu po analizie pojedynczego przykładu – mniejsze wymagania obliczeniowe i pamięciowe, mogą pojawić się fluktuacje błędu
 - 1.3 **minibatch gradient descent** – praktyczny kompromis
2. Przejście przez wszystkie przykłady zbioru treningowego – epoka (**epoch**)
3. Przygody na szlaku z gradientem
 - 3.1 Lokalne minima (**local minima**)
 - 3.2 Punkt siodłowy (**saddle point**)
 - 3.3 Zanikający gradient (**vanishing gradient**)
 - 3.4 Eksplodujący gradient (**exploding gradient**)



1. Gradient descent – iteracyjne poprawianie sieci
 - 1.1 **batch gradient descent** – aktualizacja modelu dopiero po przejściu przez wszystkie przykłady treningowe – kosztowne obliczeniowo i pamięciowo
 - 1.2 **stochastic gradient descent** – aktualizacja modelu po analizie pojedynczego przykładu – mniejsze wymagania obliczeniowe i pamięciowe, mogą pojawić się fluktuacje błędu
 - 1.3 **minibatch gradient descent** – praktyczny kompromis
2. Przejście przez wszystkie przykłady zbioru treningowego – epoka (**epoch**)
3. Przygody na szlaku z gradientem
 - 3.1 Lokalne minima (**local minima**)
 - 3.2 Punkt siodłowy (**saddle point**)
 - 3.3 Zanikający gradient (**vanishing gradient**)
 - 3.4 Eksplodujący gradient (**exploding gradient**)



1. Gradient descent – iteracyjne poprawianie sieci
 - 1.1 **batch gradient descent** – aktualizacja modelu dopiero po przejściu przez wszystkie przykłady treningowe – kosztowne obliczeniowo i pamięciowo
 - 1.2 **stochastic gradient descent** – aktualizacja modelu po analizie pojedynczego przykładu – mniejsze wymagania obliczeniowe i pamięciowe, mogą pojawić się fluktuacje błędu
 - 1.3 **minibatch gradient descent** – praktyczny kompromis
2. Przejście przez wszystkie przykłady zbioru treningowego – epoka (**epoch**)
3. Przygody na szlaku z gradientem
 - 3.1 Lokalne minima (**local minima**)
 - 3.2 Punkt siodłowy (**saddle point**)
 - 3.3 Zanikający gradient (**vanishing gradient**)
 - 3.4 Eksplodujący gradient (**exploding gradient**)



1. Gradient descent – iteracyjne poprawianie sieci
 - 1.1 **batch gradient descent** – aktualizacja modelu dopiero po przejściu przez wszystkie przykłady treningowe – kosztowne obliczeniowo i pamięciowo
 - 1.2 **stochastic gradient descent** – aktualizacja modelu po analizie pojedynczego przykładu – mniejsze wymagania obliczeniowe i pamięciowe, mogą pojawić się fluktuacje błędu
 - 1.3 **minibatch gradient descent** – praktyczny kompromis
2. Przejście przez wszystkie przykłady zbioru treningowego – epoka (**epoch**)
3. Przygody na szlaku z gradientem
 - 3.1 Lokalne minima (**local minima**)
 - 3.2 Punkt siodłowy (**saddle point**)
 - 3.3 Zanikający gradient (**vanishing gradient**)
 - 3.4 Eksplodujący gradient (**exploding gradient**)



1. Gradient descent – iteracyjne poprawianie sieci
 - 1.1 **batch gradient descent** – aktualizacja modelu dopiero po przejściu przez wszystkie przykłady treningowe – kosztowne obliczeniowo i pamięciowo
 - 1.2 **stochastic gradient descent** – aktualizacja modelu po analizie pojedynczego przykładu – mniejsze wymagania obliczeniowe i pamięciowe, mogą pojawić się fluktuacje błędu
 - 1.3 **minibatch gradient descent** – praktyczny kompromis
2. Przejście przez wszystkie przykłady zbioru treningowego – epoka (**epoch**)
3. Przygody na szlaku z gradientem
 - 3.1 Lokalne minima (**local minima**)
 - 3.2 Punkt siodłowy (**saddle point**)
 - 3.3 Zanikający gradient (**vanishing gradient**)
 - 3.4 Eksplodujący gradient (**exploding gradient**)



1. Gradient descent – iteracyjne poprawianie sieci
 - 1.1 **batch gradient descent** – aktualizacja modelu dopiero po przejściu przez wszystkie przykłady treningowe – kosztowne obliczeniowo i pamięciowo
 - 1.2 **stochastic gradient descent** – aktualizacja modelu po analizie pojedynczego przykładu – mniejsze wymagania obliczeniowe i pamięciowe, mogą pojawić się fluktuacje błędu
 - 1.3 **minibatch gradient descent** – praktyczny kompromis
2. Przejście przez wszystkie przykłady zbioru treningowego – epoka (**epoch**)
3. Przygody na szlaku z gradientem
 - 3.1 Lokalne minima (**local minima**)
 - 3.2 Punkt siodłowy (**saddle point**)
 - 3.3 Zanikający gradient (**vanishing gradient**)
 - 3.4 Eksplodujący gradient (**exploding gradient**)



1. Gradient descent – iteracyjne poprawianie sieci
 - 1.1 **batch gradient descent** – aktualizacja modelu dopiero po przejściu przez wszystkie przykłady treningowe – kosztowne obliczeniowo i pamięciowo
 - 1.2 **stochastic gradient descent** – aktualizacja modelu po analizie pojedynczego przykładu – mniejsze wymagania obliczeniowe i pamięciowe, mogą pojawić się fluktuacje błędu
 - 1.3 **minibatch gradient descent** – praktyczny kompromis
2. Przejście przez wszystkie przykłady zbioru treningowego – epoka (**epoch**)
3. Przygody na szlaku z gradientem
 - 3.1 Lokalne minima (**local minima**)
 - 3.2 Punkt siodłowy (**saddle point**)
 - 3.3 Zanikający gradient (**vanishing gradient**)
 - 3.4 Eksplodujący gradient (**exploding gradient**)





Rysunek: Gradient descent w typowych ilustracjach

Rysunek: Gradient descent w praktyce



źródło ilustracji: DALL-E

ask.iitis.pl



Rysunek: Gradient descent w typowych ilustracjach



Rysunek: Gradient descent w praktyce

źródło ilustracji: DALL-E

ask.iitis.pl

Learning rate



Mamy:

1. Adaptacja (aktualizacja) wag dla konkretnego przykładu (punktu danych)
2. Optymalizacja – iteracyjne przejście przez kolejne punkty danych i modyfikacje wag

Dalsze zagadnienia:

1. Krzywa błędu (**loss curve**)
2. Learning rate
3. Optymalizacja++



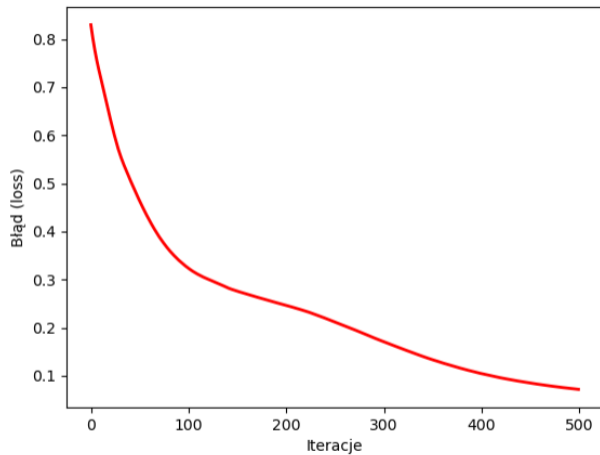
Mamy:

1. Adaptacja (aktualizacja) wag dla konkretnego przykładu (punktu danych)
2. Optymalizacja – iteracyjne przejście przez kolejne punkty danych i modyfikacje wag

Dalsze zagadnienia:

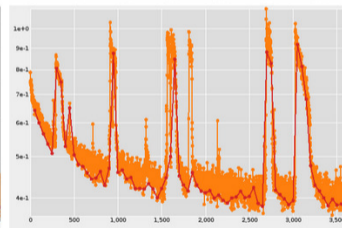
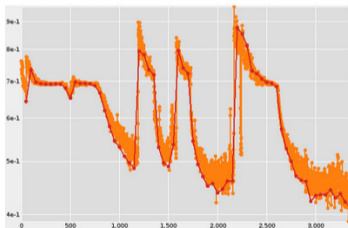
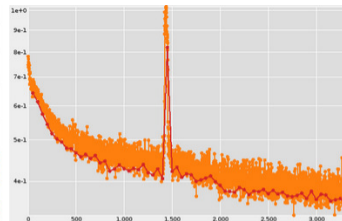
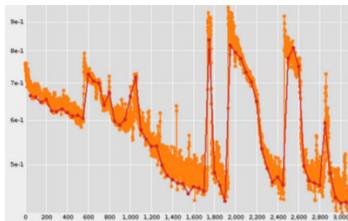
1. Krzywa błędu (**loss curve**)
2. Learning rate
3. Optymalizacja++





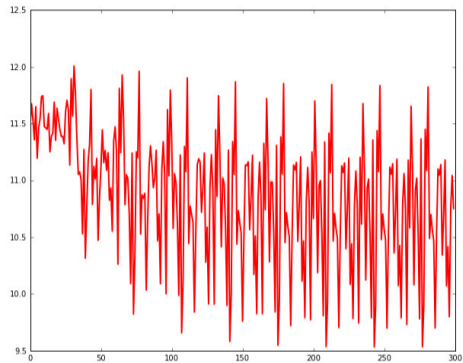
Rysunek: Przykładowa krzywa błędu





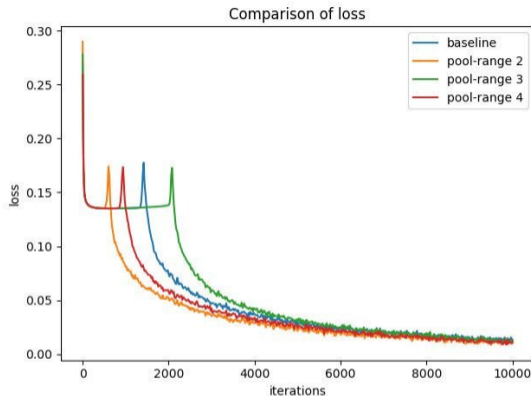
Rysunek: *Taming Spatial Transformer Networks*, contributed by Diogo. For the record, it's not supposed to look like that. (Pobrane z <https://lossfunctions.tumblr.com/>)





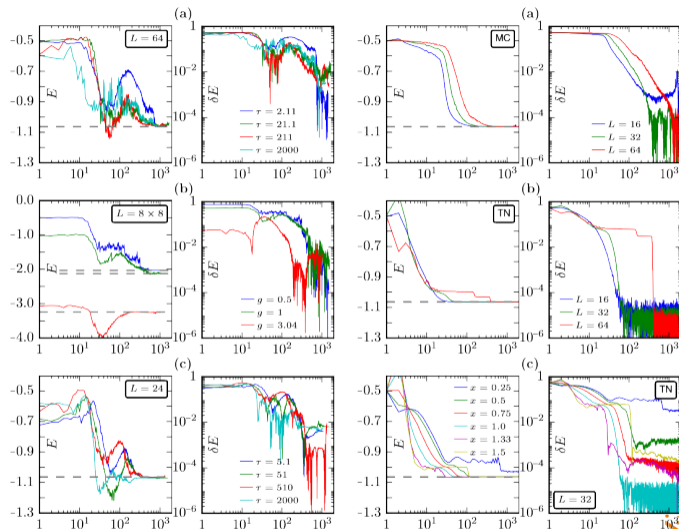
Rysunek: *A heart rate or a loss function? :) This one of a custom implementation of an RNN, graciously contributed by Ray Zhang. (Pobrane z <https://lossfunctions.tumblr.com/>)*





Rysunek: A highly amusing specimen from @_karfly . Truly baffles the mind. (Pobrane z <https://lossfunctions.tumblr.com/>)





redback.bjfu.ws: Why is my loss function going down and jumping?

J_Johnson: It's not always a straight path to the best fit. In fact, usually it's pretty bumpy.

(z: <https://discuss.pytorch.org/t/why-is-my-loss-function-going-down-and-jumping/170508>)

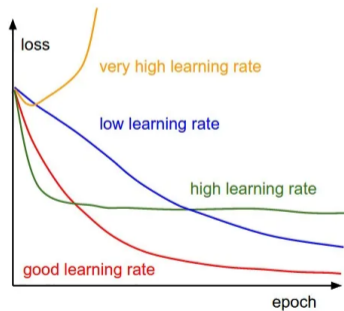
Kontrola procesu treningu:

1. Współczynnik szybkości uczenia (**learning rate**) γ , $w_{ijk}^{n+1} = w_{ijk}^n - \gamma \frac{\partial L}{\partial w_{ijk}}$



Learning rate:

- ▶ Typowe wartości 0.1 – 0.001
- ▶ Zbyt niskie wartości (zwykle) – wydłużony czas uczenia; wysokie – zatrzymanie uczenia, oscylacje
- ▶ Algorytmy: np. Adagrad, RMSprop, Adam...



- ▶ Algorytmy: np. Adagrad, RMSprop, Adam...
 - ▶ Pęd gradientu (**momentum**) – przejście przez punkty siodłowe (gdzie gradient ma niewielkie wartości) i lokalne minima
$$v^n = \mu v^{n-1} - \gamma \frac{\partial L}{\partial w_{ijk}} \quad w_{ijk}^{n+1} = w_{ijk}^n + v^n$$
 - ▶ Skalowanie (**learning rate annealing/decay**), np. $\gamma_n = \gamma_0 e^{-dn}$ – zmniejszanie wartości w trakcie uczenia



redback.bjfu.ws: Why is my loss function going down and jumping?

J_Johnson: It's not always a straight path to the best fit. In fact, usually it's pretty bumpy.

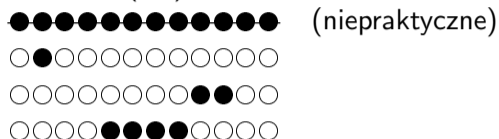
(z: <https://discuss.pytorch.org/t/why-is-my-loss-function-going-down-and-jumping/170508>)

Kontrola procesu treningu:

1. Współczynnik szybkości uczenia (**learning rate**) γ , $w_{ijk}^{n+1} = w_{ijk}^n - \gamma \frac{\partial L}{\partial w_{ijk}}$
2. Rozmiar podzbioru danych (**batch size**)



Batch size (BS):



popularne wartości np. 32, 64 / 128, 256

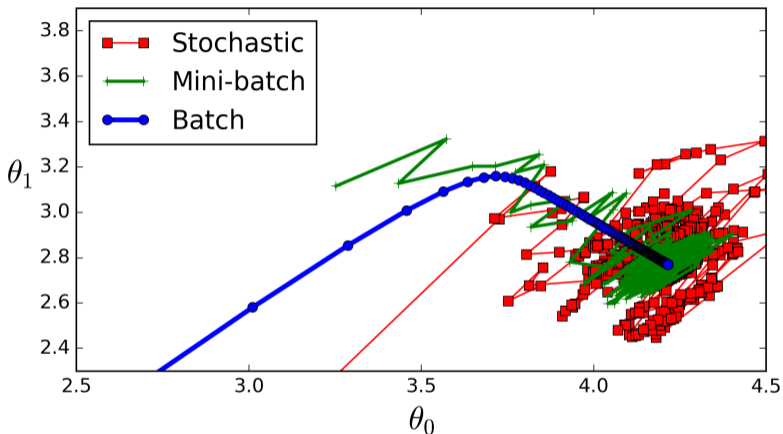
małe wartości np. 1, 2, 4, 8 – szybsze zmiany w sieci (więcej na epokę)

duże wartości wymagają więcej pamięci, mniej szumu

Przy małej wartości BS mogą pojawić się fluktuacje (widoczne w krzywej błędów), niekoniecznie negatywne – mogą uodparniać przed przetrenowaniem



Stochastic vs minibatch vs batch gradient descent



Źródło ilustracji: Odpowiedź *itdxer* w pytaniu *What is batch size in neural network?* CrossValidated StackExchange



Wartości gradientu, inicjalizacja – wartości początkowe



$$y_{11} = \max(20x - 10, 0)$$

$$y_{21} = \max(20y_{11} - 10, 0) \quad y_{31} = \max(20y_{21} - 10, 0) \quad y_{41} = \max(20y_{31} - 10, 0)$$

$$x = 1 \rightarrow y_{11} = 10 \quad y_{21} = 190 \quad y_{31} = 3790 \quad y_{41} = 75790$$

$$y_{11} = \max(0.2x, 0) \quad y_{21} = \max(0.2y_{11}, 0) \quad \dots$$

$$x = 1 \rightarrow y_{11} = 0.2 \quad y_{21} = 0.04 \quad y_{31} = 0.008 \quad y_{41} = 0.0016$$

- ▶ Wartości „przepełniają” przez sieć, podczas wyliczania wyjść neuronów (\rightarrow) i gradientów (\leftarrow)
- ▶ Wartości wyliczanych wyjść/gradientów wrażliwe na zakres wartości wag
- ▶ Eksplodujący gradient (**exploding gradients**) – duże modyfikacje do wag, niestabilności/NaN
- ▶ Zanikający gradient (**vanishing gradient**) – małe (lub brak) modyfikacji do wag, spowolniony/zatrzymany proces uczenia



$$y_{11} = \max(20x - 10, 0)$$

$$y_{21} = \max(20y_{11} - 10, 0) \quad y_{31} = \max(20y_{21} - 10, 0) \quad y_{41} = \max(20y_{31} - 10, 0)$$

$$x = 1 \rightarrow y_{11} = 10 \quad y_{21} = 190 \quad y_{31} = 3790 \quad y_{41} = 75790$$

$$y_{11} = \max(0.2x, 0) \quad y_{21} = \max(0.2y_{11}, 0) \quad \dots$$

$$x = 1 \rightarrow y_{11} = 0.2 \quad y_{21} = 0.04 \quad y_{31} = 0.008 \quad y_{41} = 0.0016$$

- ▶ Wartości „przeptywają” przez sieć, podczas wyliczania wyjść neuronów (\rightarrow) i gradientów (\leftarrow)
- ▶ Wartości wyliczanych wyjść/gradientów wrażliwe na zakres wartości wag
- ▶ Eksplodujący gradient (**exploding gradients**) – duże modyfikacje do wag, niestabilności/NaN
- ▶ Zanikający gradient (**vanishing gradient**) – małe (lub brak) modyfikacji do wag, spowolniony/zatrzymany proces uczenia



$$y_{11} = \max(20x - 10, 0)$$

$$y_{21} = \max(20y_{11} - 10, 0) \quad y_{31} = \max(20y_{21} - 10, 0) \quad y_{41} = \max(20y_{31} - 10, 0)$$

$$x = 1 \rightarrow y_{11} = 10 \quad y_{21} = 190 \quad y_{31} = 3790 \quad y_{41} = 75790$$

$$y_{11} = \max(0.2x, 0) \quad y_{21} = \max(0.2y_{11}, 0) \quad \dots$$

$$x = 1 \rightarrow y_{11} = 0.2 \quad y_{21} = 0.04 \quad y_{31} = 0.008 \quad y_{41} = 0.0016$$

- ▶ Wartości „przeptywają” przez sieć, podczas wyliczania wyjść neuronów (\rightarrow) i gradientów (\leftarrow)
- ▶ Wartości wyliczanych wyjść/gradientów wrażliwe na zakres wartości wag
- ▶ Eksplodujący gradient (**exploding gradients**) – duże modyfikacje do wag, niestabilności/NaN
- ▶ Zanikający gradient (**vanishing gradient**) – małe (lub brak) modyfikacji do wag, spowolniony/zatrzymany proces uczenia



$$y_{11} = \max(20x - 10, 0)$$

$$y_{21} = \max(20y_{11} - 10, 0) \quad y_{31} = \max(20y_{21} - 10, 0) \quad y_{41} = \max(20y_{31} - 10, 0)$$

$$x = 1 \rightarrow y_{11} = 10 \quad y_{21} = 190 \quad y_{31} = 3790 \quad y_{41} = 75790$$

$$y_{11} = \max(0.2x, 0) \quad y_{21} = \max(0.2y_{11}, 0) \quad \dots$$

$$x = 1 \rightarrow y_{11} = 0.2 \quad y_{21} = 0.04 \quad y_{31} = 0.008 \quad y_{41} = 0.0016$$

- ▶ Wartości „przeptywają” przez sieć, podczas wyliczania wyjść neuronów (\rightarrow) i gradientów (\leftarrow)
- ▶ Wartości wyliczanych wyjść/gradientów wrażliwe na zakres wartości wag
- ▶ Eksplodujący gradient (**exploding gradients**) – duże modyfikacje do wag, niestabilności/NaN
- ▶ Zanikający gradient (**vanishing gradient**) – małe (lub brak) modyfikacji do wag, spowolniony/zatrzymany proces uczenia



$$y_{11} = \max(20x - 10, 0)$$

$$y_{21} = \max(20y_{11} - 10, 0) \quad y_{31} = \max(20y_{21} - 10, 0) \quad y_{41} = \max(20y_{31} - 10, 0)$$

$$x = 1 \rightarrow y_{11} = 10 \quad y_{21} = 190 \quad y_{31} = 3790 \quad y_{41} = 75790$$

$$y_{11} = \max(0.2x, 0) \quad y_{21} = \max(0.2y_{11}, 0) \quad \dots$$

$$x = 1 \rightarrow y_{11} = 0.2 \quad y_{21} = 0.04 \quad y_{31} = 0.008 \quad y_{41} = 0.0016$$

- ▶ Wartości „przepływają” przez sieć, podczas wyliczania wyjść neuronów (\rightarrow) i gradientów (\leftarrow)
- ▶ Wartości wyliczanych wyjść/gradientsów wrażliwe na zakres wartości wag
- ▶ Eksplodujący gradient (**exploding gradients**) – duże modyfikacje do wag, niestabilności/NaN
- ▶ Zanikający gradient (**vanishing gradient**) – małe (lub brak) modyfikacji do wag, spowolniony/zatrzymany proces uczenia



$$y_{11} = \max(20x - 10, 0)$$

$$y_{21} = \max(20y_{11} - 10, 0) \quad y_{31} = \max(20y_{21} - 10, 0) \quad y_{41} = \max(20y_{31} - 10, 0)$$

$$x = 1 \rightarrow y_{11} = 10 \quad y_{21} = 190 \quad y_{31} = 3790 \quad y_{41} = 75790$$

$$y_{11} = \max(0.2x, 0) \quad y_{21} = \max(0.2y_{11}, 0) \quad \dots$$

$$x = 1 \rightarrow y_{11} = 0.2 \quad y_{21} = 0.04 \quad y_{31} = 0.008 \quad y_{41} = 0.0016$$

- ▶ Wartości „przepływają” przez sieć, podczas wyliczania wyjść neuronów (\rightarrow) i gradientów (\leftarrow)
- ▶ Wartości wyliczanych wyjść/gradientów wrażliwe na zakres wartości wag
- ▶ Eksplodujący gradient (**exploding gradients**) – duże modyfikacje do wag, niestabilności/NaN
- ▶ Zanikający gradient (**vanishing gradient**) – małe (lub brak) modyfikacji do wag, spowolniony/zatrzymany proces uczenia



1. Przygotowanie danych

Normalizacja kontrola zakresu danych, np. $\hat{x}_{ij} = \frac{x_{ij} - \min \mathbf{x}_i}{\max \mathbf{x}_i - \min \mathbf{x}_i}$

Standaryzacja korekta względem średniej i wariancji $\hat{x}_{ij} = \frac{x_{ij} - \mu_i}{\sigma_i}$

2. Przygotowanie sieci

Inicjalizacja początkowe wartości wag losowane np. $w_{ijk} = \mathcal{N}(\mu = 0, \sigma = 0.01)$

Xavier Glorot uwzględnia średnią liczbę połączeń do neuronu

$$w_{ijk} = \mathcal{N}\left(\mu = 0, \sigma = \sqrt{\frac{1}{\text{fan}_{\text{avg}}}}\right) \text{ lub } w_{ijk} = \mathcal{U}\left(\sqrt{\frac{3}{\text{fan}_{\text{avg}}}}\right)$$

Kaming He rozwija inicjalizację XG pod kątem ReLU $w_{ijk} = \mathcal{N}\left(\mu = 0, \sigma = \sqrt{\frac{2}{\text{fan}_{\text{in}}}}\right)$

$$\text{lub } w_{ijk} = \mathcal{U}\left(\sqrt{\frac{6}{\text{fan}_{\text{avg}}}}\right)$$

3. Losowa inicjalizacja + optymalizacja lokalna = możliwe różne wyniki treningu w zależności od stanu początkowego



1. Przygotowanie danych

Normalizacja kontrola zakresu danych, np. $\hat{x}_{ij} = \frac{x_{ij} - \min \mathbf{x}_i}{\max \mathbf{x}_i - \min \mathbf{x}_i}$

Standaryzacja korekta względem średniej i wariancji $\hat{x}_{ij} = \frac{x_{ij} - \mu_i}{\sigma_i}$

2. Przygotowanie sieci

Inicjalizacja początkowe wartości wag losowane np. $w_{ijk} = \mathcal{N}(\mu = 0, \sigma = 0.01)$

Xavier Glorot uwzględnia średnią liczbę połączeń do neuronu

$$w_{ijk} = \mathcal{N}\left(\mu = 0, \sigma = \sqrt{\frac{1}{\text{fan}_{\text{avg}}}}\right) \text{ lub } w_{ijk} = \mathcal{U}\left(\sqrt{\frac{3}{\text{fan}_{\text{avg}}}}\right)$$

Kaming He rozwija inicjalizację XG pod kątem ReLU $w_{ijk} = \mathcal{N}\left(\mu = 0, \sigma = \sqrt{\frac{2}{\text{fan}_{\text{in}}}}\right)$

$$\text{lub } w_{ijk} = \mathcal{U}\left(\sqrt{\frac{6}{\text{fan}_{\text{avg}}}}\right)$$

3. Losowa inicjalizacja + optymalizacja lokalna = możliwe różne wyniki treningu w zależności od stanu początkowego



1. Przygotowanie danych

Normalizacja kontrola zakresu danych, np. $\hat{x}_{ij} = \frac{x_{ij} - \min \mathbf{x}_i}{\max \mathbf{x}_i - \min \mathbf{x}_i}$

Standaryzacja korekta względem średniej i wariancji $\hat{x}_{ij} = \frac{x_{ij} - \mu_i}{\sigma_i}$

2. Przygotowanie sieci

Inicjalizacja początkowe wartości wag losowane np. $w_{ijk} = \mathcal{N}(\mu = 0, \sigma = 0.01)$

Xavier Glorot uwzględnia średnią liczbę połączeń do neuronu

$$w_{ijk} = \mathcal{N}\left(\mu = 0, \sigma = \sqrt{\frac{1}{\text{fan}_{\text{avg}}}}\right) \text{ lub } w_{ijk} = \mathcal{U}\left(\sqrt{\frac{3}{\text{fan}_{\text{avg}}}}\right)$$

Kaming He rozwija inicjalizację XG pod kątem ReLU $w_{ijk} = \mathcal{N}\left(\mu = 0, \sigma = \sqrt{\frac{2}{\text{fan}_{\text{in}}}}\right)$

$$\text{lub } w_{ijk} = \mathcal{U}\left(\sqrt{\frac{6}{\text{fan}_{\text{avg}}}}\right)$$

3. Losowa inicjalizacja + optymalizacja lokalna = możliwe różne wyniki treningu w zależności od stanu początkowego

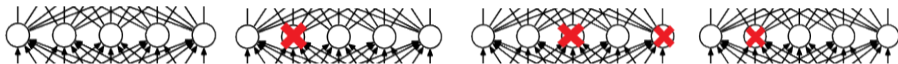


Regularyzacja



► Wsparcie treningu – kontrola wartości wag

Dropout z określonym prawdopodobieństwem (np. $p = 0.2$ albo $p = 0.5$) zerujemy wyjście z neuronu w trakcie treningu



Batch Normalization normalizacja i przeskalowanie wartości pomiędzy warstwami

$$y = \gamma \frac{x - \mu_B}{\sigma_B} + \beta \quad (\text{standaryzacja} + \text{przeskalowanie} + \text{przesunięcie})$$

L2/L1 regularization „kara” dla wysokich wartości wag

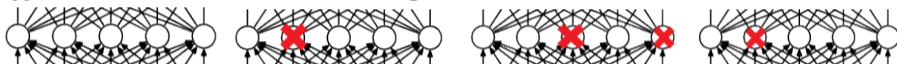
$$L(y^{\text{true}}, y^{\text{out}}) = (y^{\text{true}} - y_{21})^2 + \lambda \sum_{i=1}^n w_i^2$$

$$L(y^{\text{true}}, y^{\text{out}}) = (y^{\text{true}} - y_{21})^2 + \lambda \sum_{i=1}^n |w_i|$$



► Wsparcie treningu – kontrola wartości wag

Dropout z określonym prawdopodobieństwem (np. $p = 0.2$ albo $p = 0.5$) zerujemy wyjście z neuronu w trakcie treningu



Batch Normalization normalizacja i przeskalowanie wartości pomiędzy warstwami

$$y = \gamma \frac{x - \mu_B}{\sigma_B} + \beta \text{ (standaryzacja + przeskalowanie + przesunięcie)}$$

L2/L1 regularization „kara” dla wysokich wartości wag

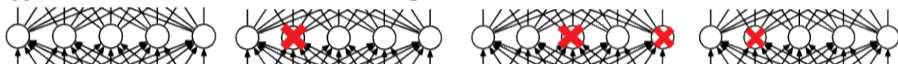
$$L(y^{\text{true}}, y^{\text{out}}) = (y^{\text{true}} - y_{21})^2 + \lambda \sum_{i=1}^n w_i^2$$

$$L(y^{\text{true}}, y^{\text{out}}) = (y^{\text{true}} - y_{21})^2 + \lambda \sum_{i=1}^n |w_i|$$



► Wsparcie treningu – kontrola wartości wag

Dropout z określonym prawdopodobieństwem (np. $p = 0.2$ albo $p = 0.5$) zerujemy wyjście z neuronu w trakcie treningu



Batch Normalization normalizacja i przeskalowanie wartości pomiędzy warstwami

$$y = \gamma \frac{x - \mu_B}{\sigma_B} + \beta \quad (\text{standaryzacja} + \text{przeskalowanie} + \text{przesunięcie})$$

L2/L1 regularization „kara” dla wysokich wartości wag

$$L(y^{\text{true}}, y^{\text{out}}) = (y^{\text{true}} - y_{21})^2 + \lambda \sum_{i=1}^n w_i^2$$

$$L(y^{\text{true}}, y^{\text{out}}) = (y^{\text{true}} - y_{21})^2 + \lambda \sum_{i=1}^n |w_i|$$



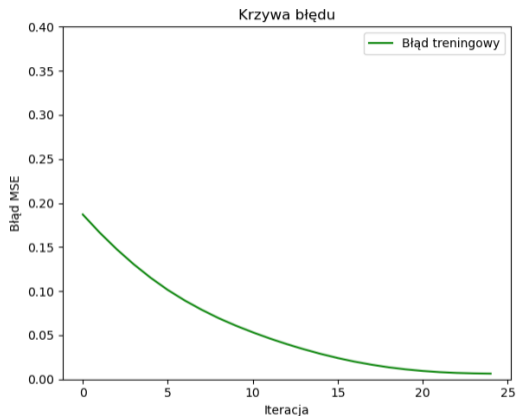
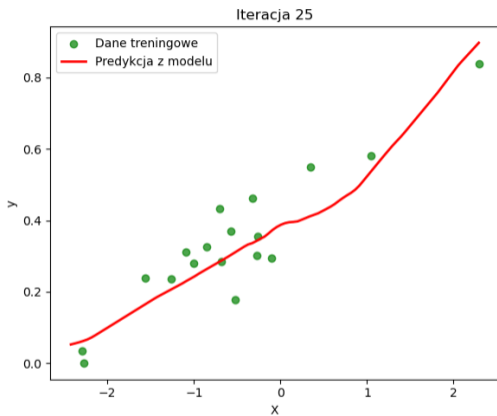
Mamy:

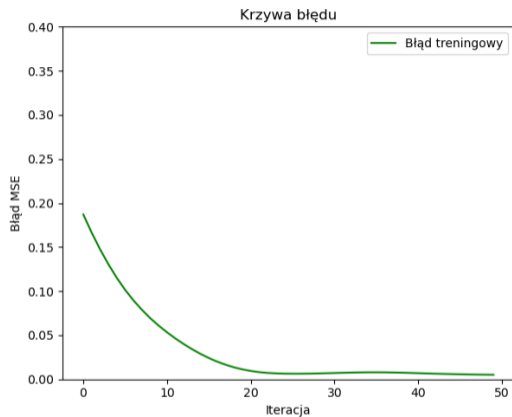
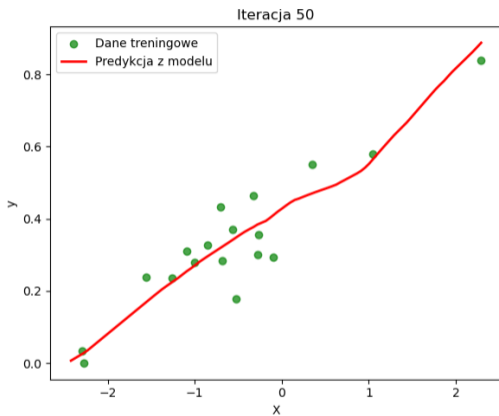
1. Adaptacja (aktualizacja) wag dla konkretnego przykładu (punktu danych)
2. Optymalizacja – iteracyjne przejście przez kolejne punkty danych i modyfikacje wag
3. Sterowanie procesem uczenia – learning rate, batch size, wybór optymalizatora
4. Inicjalizacja
5. Regularyzacja

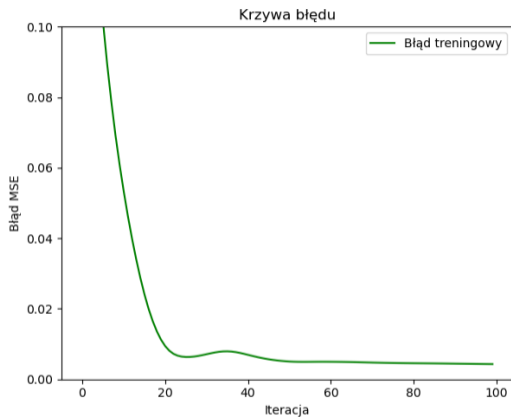
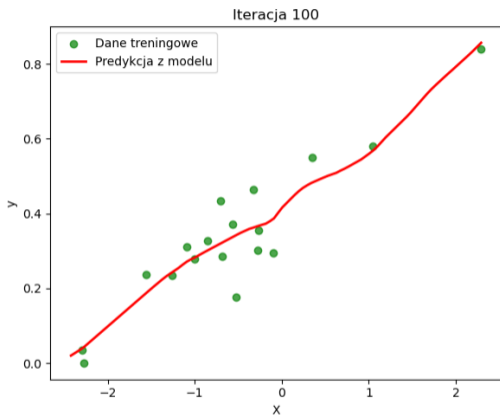


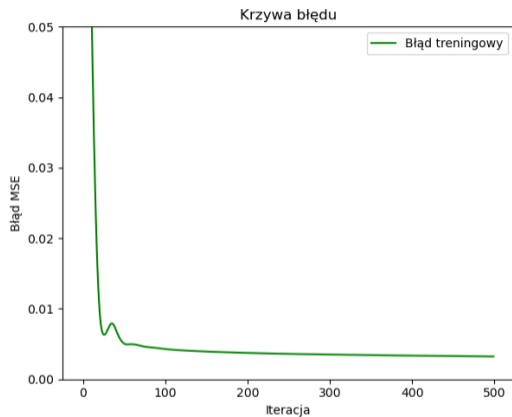
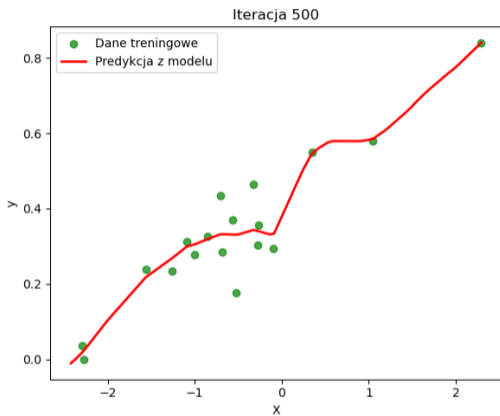
Zbiory testowe, walidacyjne, przetrenowanie, hiperparametry

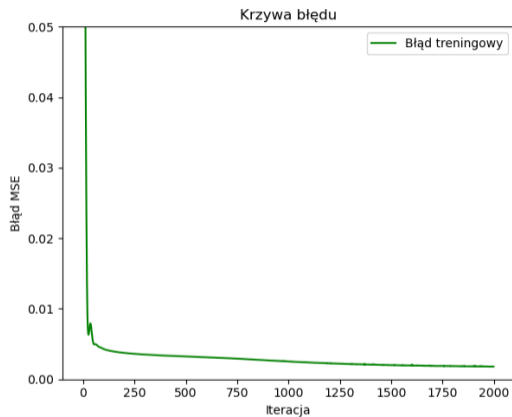
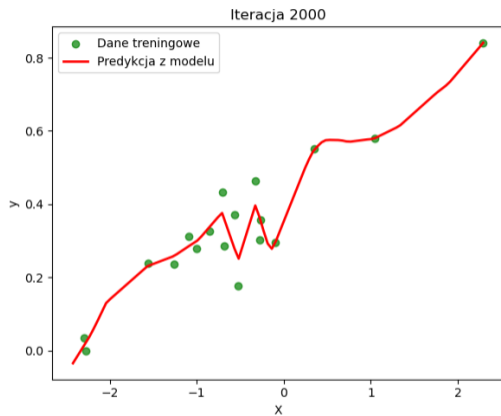


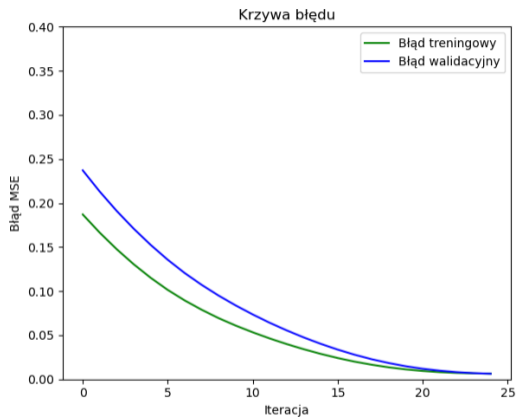
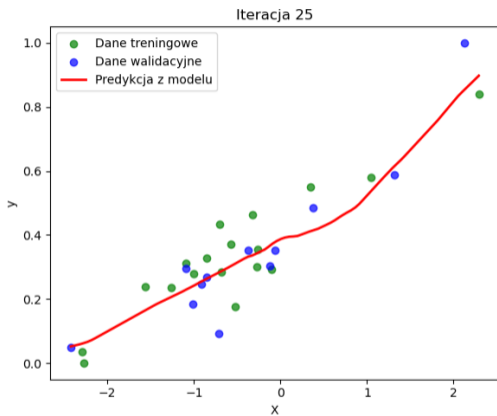


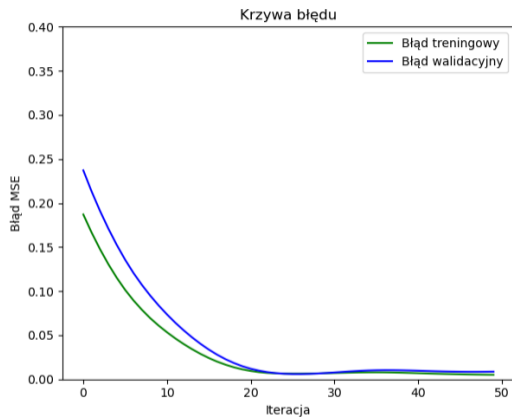
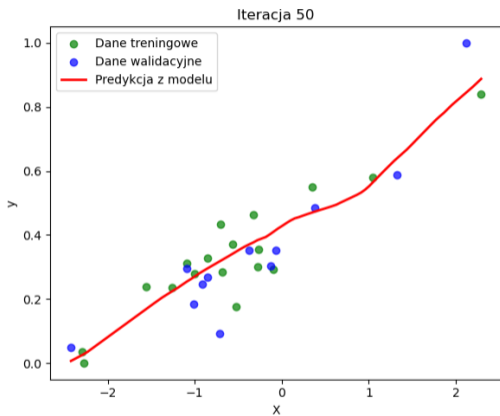


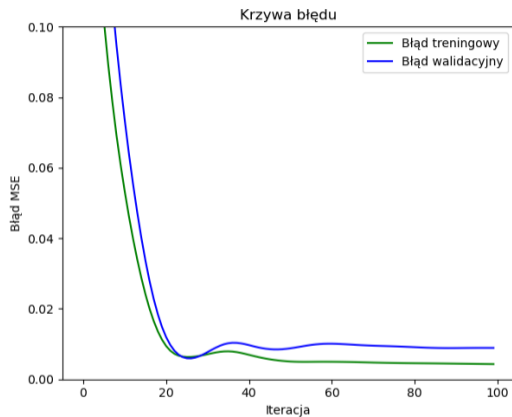
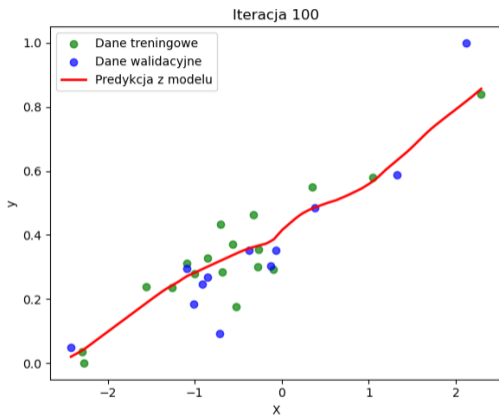


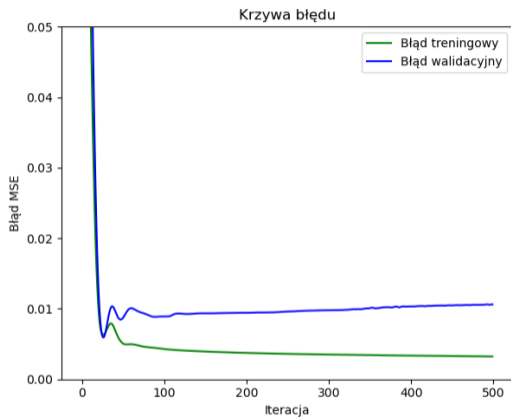
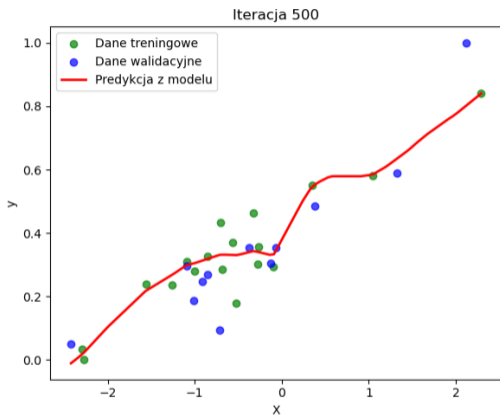


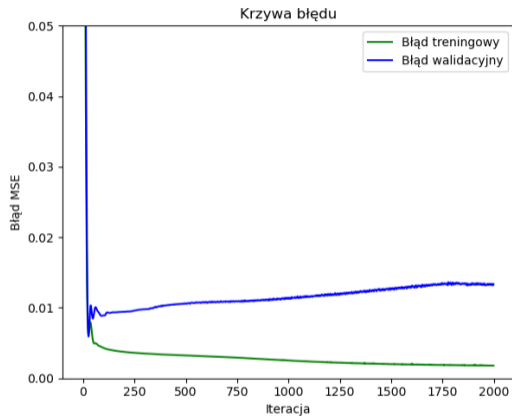
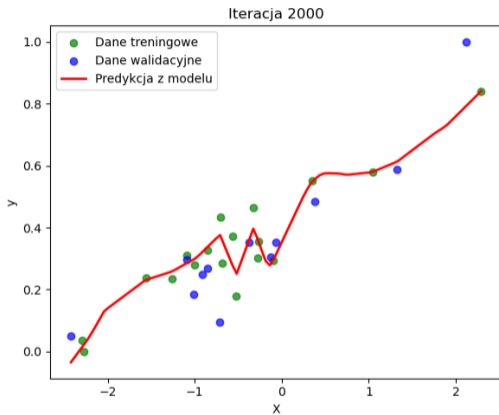












zbiór treningowy optymalizacja wag modelu

zbiór walidacyjny zabezpiecza przed przetrenowaniem, pozwala wybrać hiperparametry (np. learning rate 0.01 czy 0.001?), ułatwia decyzję o zakończeniu treningu, diagnostyka procesu treningu

zbiór testowy oszacowanie rzeczywistej wydajności (skuteczności, wyniku) wytrenowanego modelu

Zbiory rozłączne!

- ▶ **Przetrenowanie** – przesadne dopasowanie do zbioru treningowego
- ▶ **Generalizacja** – dopasowanie do charakterystyki danych, którą zbiór treningowy reprezentuje



Zmienne procesu treningu:

1. Dane

- 1.1 Rozmiar zbioru, liczba przykładów – czy jest wystarczający?
- 1.2 Diagnostyka przykładów – podobieństwo/różnorodność? Outliery?

2. Model

- 2.1 Architektura – liczba warstw, liczba neuronów
- 2.2 Dobór hiperparametrów, regularyzacja
- 2.3 Rozmiar modelu względem rozmiaru zbioru danych

3. Trening

- 3.1 Dostępne zasoby obliczeniowe – przeszukiwanie przestrzeni hiperparametrów, sprawdzenie różnych stanów początkowych
- 3.2 Weryfikacja zbieżności na podzbiorze przykładów
- 3.3 Diagnostyka wyników



Co jeżeli zbiór danych jest niewystarczający? Augmentacja danych (**data augmentation**)

- ▶ Wprowadzenie szumu (różnego rodzaju)
- ▶ Edycja strukturalna (wycięcie fragmentu, obroty, skalowanie, ...)
- ▶ Składanie nowego przykładu z kilku przykładów
- ▶ ...

Źródło ilustracji: T. Kumar et al, Image Data Augmentation Approaches: A Comprehensive Survey and Future directions, arXiv:2301.02830



Co jeżeli zbiór danych jest niewystarczający? Augmentacja danych (**data augmentation**)

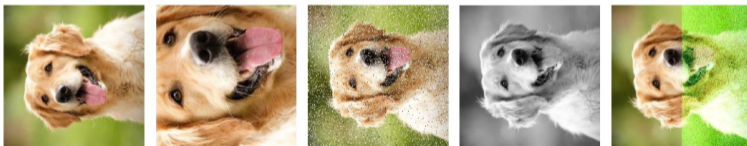
- ▶ Wprowadzenie szumu (różnego rodzaju)
- ▶ Edycja strukturalna (wycięcie fragmentu, obroty, skalowanie, ...)
- ▶ Składanie nowego przykładu z kilku przykładów
- ▶ ...

Źródło ilustracji: T. Kumar et al, Image Data Augmentation Approaches: A Comprehensive Survey and Future directions, arXiv:2301.02830



Co jeżeli zbiór danych jest niewystarczający? Augmentacja danych (**data augmentation**)

- ▶ Wprowadzenie szumu (różnego rodzaju)
- ▶ Edycja strukturalna (wycięcie fragmentu, obroty, skalowanie, ...)
- ▶ Składanie nowego przykładu z kilku przykładów
- ▶ ...



Źródło ilustracji: T. Kumar et al, Image Data Augmentation Approaches: A Comprehensive Survey and Future directions, arXiv:2301.02830



Konstrukcja sieci



Softmax



$$y_{11} = \max(20x - 10, 0) \quad y_{12} = \max(40x - 200, 0) \quad y_{21} = \max(y_{11} - y_{12}, 0)$$

Regresja – otrzymujemy wartość liczbową – jak dobry jest owoc w skali 0-10

A podział na „dobry” i „zły” – klasyfikacja?

Modyfikacja wyjścia sieci, tak żeby na końcu

- ▶ y_{21} wartość liczbowa określająca jak bardzo przykład należy do klasy „dobry”
- ▶ y_{22} [...] do klasy „zły”

Problematyczne uczenie, bo np.

- ▶ $y_{21} = 0.02, y_{22} = 0.6 \rightarrow$ „zły owoc”
- ▶ $y_{21} = 1.7, y_{22} = 0.6 \rightarrow$ „dobry owoc”



$$y_{11} = \max(20x - 10, 0) \quad y_{12} = \max(40x - 200, 0) \quad y_{21} = \max(y_{11} - y_{12}, 0)$$

Regresja – otrzymujemy wartość liczbową – jak dobry jest owoc w skali 0-10

A podział na „dobry” i „zły” – klasyfikacja?

Modyfikacja wyjścia sieci, tak żeby na końcu

- ▶ y_{21} wartość liczbowa określająca jak bardzo przykład należy do klasy „dobry”
- ▶ y_{22} [...] do klasy „zły”

Problematiczne uczenie, bo np.

- ▶ $y_{21} = 0.02, y_{22} = 0.6 \rightarrow$ „zły owoc”
- ▶ $y_{21} = 1.7, y_{22} = 0.6 \rightarrow$ „dobry owoc”



$$y_{11} = \max(20x - 10, 0) \quad y_{12} = \max(40x - 200, 0) \quad y_{21} = \max(y_{11} - y_{12}, 0)$$

Regresja – otrzymujemy wartość liczbową – jak dobry jest owoc w skali 0-10

A podział na „dobry” i „zły” – klasyfikacja?

Modyfikacja wyjścia sieci, tak żeby na końcu

- ▶ y_{21} wartość liczbowa określająca jak bardzo przykład należy do klasy „dobry”
- ▶ y_{22} [...] do klasy „zły”

Problematiczne uczenie, bo np.

- ▶ $y_{21} = 0.02, y_{22} = 0.6 \rightarrow$ „zły owoc”
- ▶ $y_{21} = 1.7, y_{22} = 0.6 \rightarrow$ „dobry owoc”



$$y_{11} = \max(20x - 10, 0) \quad y_{12} = \max(40x - 200, 0) \quad y_{21} = \max(y_{11} - y_{12}, 0)$$

Regresja – otrzymujemy wartość liczbową – jak dobry jest owoc w skali 0-10

A podział na „dobry” i „zły” – klasyfikacja?

Modyfikacja wyjścia sieci, tak żeby na końcu

- ▶ y_{21} wartość liczbowa określająca jak bardzo przykład należy do klasy „dobry”
- ▶ y_{22} [...] do klasy „zły”

Problematic learning, for example.

- ▶ $y_{21} = 0.02, y_{22} = 0.6 \rightarrow$ „zły owoc”
- ▶ $y_{21} = 1.7, y_{22} = 0.6 \rightarrow$ „dobry owoc”



$$y_{11} = \max(20x - 10, 0) \quad y_{12} = \max(40x - 200, 0) \quad y_{21} = \max(y_{11} - y_{12}, 0)$$

Regresja – otrzymujemy wartość liczbową – jak dobry jest owoc w skali 0-10

A podział na „dobry” i „zły” – klasyfikacja?

Modyfikacja wyjścia sieci, tak żeby na końcu

- ▶ y_{21} wartość liczbowa określająca jak bardzo przykład należy do klasy „dobry”
- ▶ y_{22} [...] do klasy „zły”

Problematyczne uczenie, bo np.

- ▶ $y_{21} = 0.02, y_{22} = 0.6 \rightarrow$ „zły owoc”
- ▶ $y_{21} = 1.7, y_{22} = 0.6 \rightarrow$ „dobry owoc”



$$\text{Softmax: } \sigma(\mathbf{y})_i = \frac{e^{y_i}}{\sum_{j=1}^N e^{y_j}}$$

$$\mathbf{y}_a = [0.02, 0.6]$$

$$\sigma\left([0.02, 0.6]\right)_1 = \frac{e^{0.02}}{e^{0.02} + e^{0.6}} \approx 0.36$$

$$\sigma\left([0.02, 0.6]\right)_2 = \frac{e^{0.6}}{e^{0.02} + e^{0.6}} \approx 0.64$$

$$\sigma(\mathbf{y}_a) \approx [0.36, 0.64]$$

$$\mathbf{y}_b = [1.7, 0.6]$$

$$\sigma(\mathbf{y}_b) \approx [0.75, 0.25]$$

Szacowane prawdopodobieństwo przynależności do poszczególnych klas



$$\text{Softmax: } \sigma(\mathbf{y})_i = \frac{e^{y_i}}{\sum_{j=1}^N e^{y_j}}$$

$$\mathbf{y}_a = [0.02, 0.6]$$

$$\sigma\left([0.02, 0.6]\right)_1 = \frac{e^{0.02}}{e^{0.02} + e^{0.6}} \approx 0.36$$

$$\sigma\left([0.02, 0.6]\right)_2 = \frac{e^{0.6}}{e^{0.02} + e^{0.6}} \approx 0.64$$

$$\sigma(\mathbf{y}_a) \approx [0.36, 0.64]$$

$$\mathbf{y}_b = [1.7, 0.6]$$

$$\sigma(\mathbf{y}_b) \approx [0.75, 0.25]$$

Szacowane prawdopodobieństwo przynależności do poszczególnych klas



$$\text{Softmax: } \sigma(\mathbf{y})_i = \frac{e^{y_i}}{\sum_{j=1}^N e^{y_j}}$$

$$\mathbf{y}_a = [0.02, 0.6]$$

$$\sigma\left([0.02, 0.6]\right)_1 = \frac{e^{0.02}}{e^{0.02} + e^{0.6}} \approx 0.36$$

$$\sigma\left([0.02, 0.6]\right)_2 = \frac{e^{0.6}}{e^{0.02} + e^{0.6}} \approx 0.64$$

$$\sigma(\mathbf{y}_a) \approx [0.36, 0.64]$$

$$\mathbf{y}_b = [1.7, 0.6]$$

$$\sigma(\mathbf{y}_b) \approx [0.75, 0.25]$$

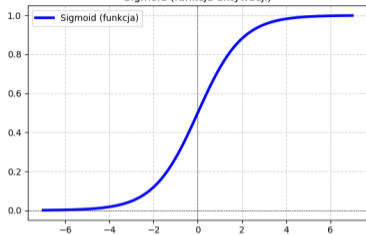
Szacowane prawdopodobieństwo przynależności do poszczególnych klas



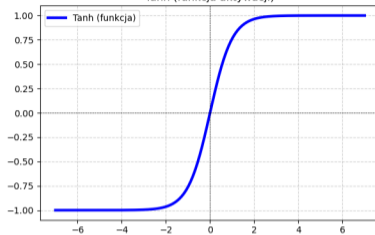
Funkcje aktywacji



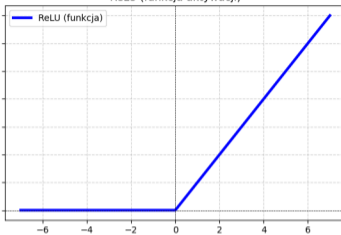
Sigmoid (funkcja aktywacji)



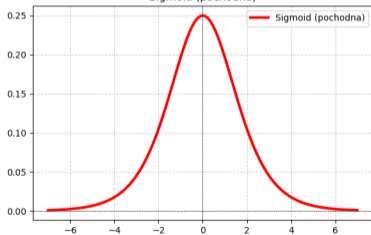
Tanh (funkcja aktywacji)



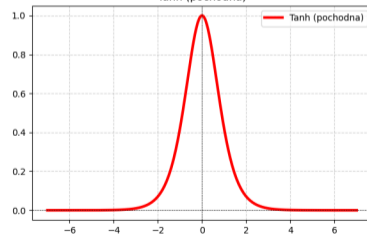
ReLU (funkcja aktywacji)



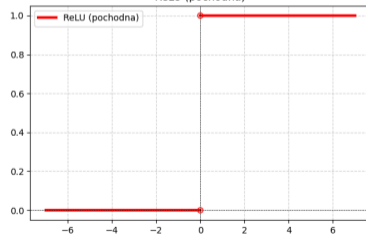
Sigmoid (pochodna)



Tanh (pochodna)



ReLU (pochodna)



Rozważmy zbiór danych (fragment zbioru Iris):

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]]
```

i neuron ReLU z wagami $[0.2, -0.4, 0.1, -0.5]$

$5.1 \times 0.2 + 3.5 \times -0.4 + 1.4 \times 0.1 + 0.2 \times -0.5 = 1.02 - 1.4 + 0.14 - 0.1 = -0.34 \dots$
... i tak dla każdego – neuron jest „martwy” (brak niezerowej wartości, zerowy gradient)



Rozważmy zbiór danych (fragment zbioru Iris):

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]]
```

i neuron ReLU z wagami $[0.2, -0.4, 0.1, -0.5]$

$$5.1 \times 0.2 + 3.5 \times -0.4 + 1.4 \times 0.1 + 0.2 \times -0.5 = 1.02 - 1.4 + 0.14 - 0.1 = -0.34 \dots$$

... i tak dla każdego – neuron jest „martwy” (brak niezerowej wartości, zerowy gradient)



$$\text{ReLU}(x) = \max(x, 0)$$

Alternatywy:

- ▶ Leaky ReLU $f(x) = \begin{cases} x & x > 0, \\ \alpha x & x \leq 0 \end{cases}$, np. $\alpha = 0.1$
- ▶ Parametric ReLU (α jest parametrem wyznaczanym w trakcie treningu)
- ▶ SiLU $f(x) = x \times \text{sigmoid}(x)$
- ▶ ELU $f(x) = \begin{cases} x & x > 0, \\ \alpha(e^x - 1) & x \leq 0 \end{cases}$, $\alpha \geq 0$ jest hiperparametrem
- ▶ Softplus $f(x) = \ln(1 + e^x)$
- ▶ ...



Funkcje błędu (loss)



- ▶ Mean Squared Error (MSE) $L(x, y) = (x - y)^2$
- ▶ Mean Absolute Error (MAE) $L(x, y) = |x - y|$
- ▶ Negative Log-Likelihood Loss function (NLL) $L(x, y) = -y \log x - (1 - y) \log(1 - x)$
- ▶ Cross-Entropy Loss (\approx softmax + NLL)
- ▶ Margin Ranking Loss $L(x_1, x_2, y) = \max(0, -y(x_1 - x_2) + m)$, (m margins)
- ▶ Kullback-Leibler Divergence Loss $L(x, y) = y(\log y - x)$
- ▶ ...



Zaawansowane architektury – sieci konwolucyjne

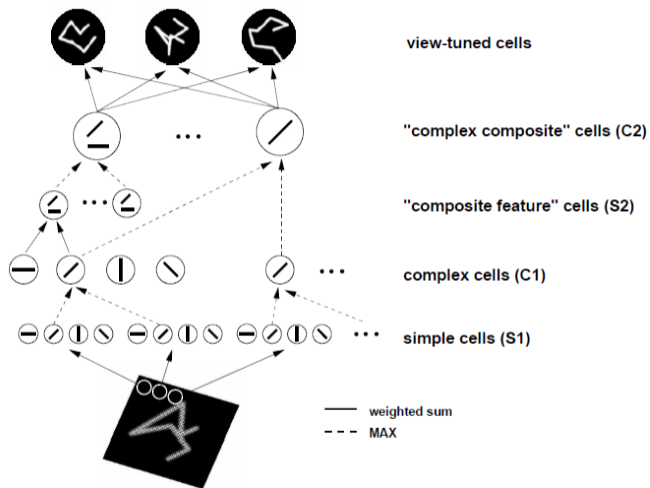


Problem: rozpoznawanie obiektów w obrazach

- ▶ Obiekt przesunięty, obrócony, przeskalowany, zdeformowany (operacja rzutowania)
- ▶ Cechy charakterystyczne – złożone, rzadkie, różne dla różnych rodzajów obiektów



Model HMAX:

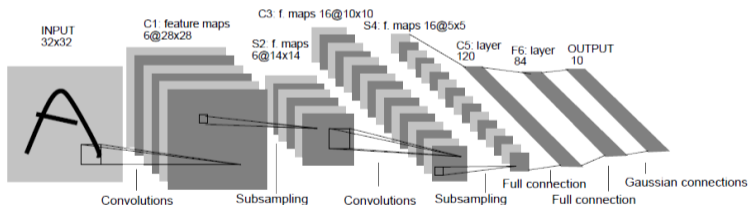


Źródło ilustracji: Riesenhuber, M. & Poggio, T. Hierarchical models of object recognition in cortex. Nat. Neurosci. 2, 1019-1025



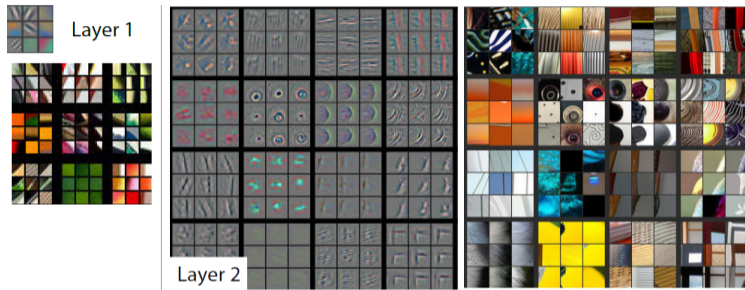
Architektura konwolucyjna:

- ▶ Filtry konwolucyjne (uczone/trenowane)
- ▶ Operacja „zbierania” (pooling)
- ▶ Pary konwolucja+pooling powtórzone kilka razy
- ▶ Na końcu kilka warstw sieci MLP



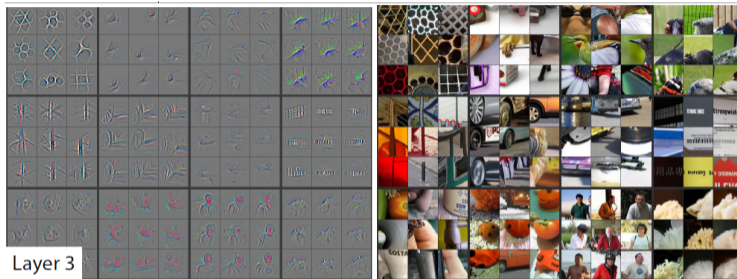
Źródło ilustracji: Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. (1998). "Gradient-based learning applied to document recognition" (PDF). Proceedings of the IEEE. 86 (11): 2278–2324





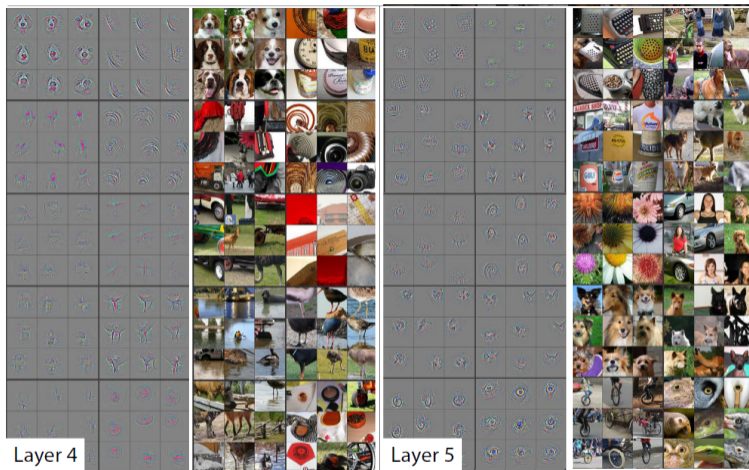
Źródło ilustracji: Zeiler, M.D., Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds) Computer Vision – ECCV 2014





Źródło ilustracji: Zeiler, M.D., Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds) Computer Vision – ECCV 2014

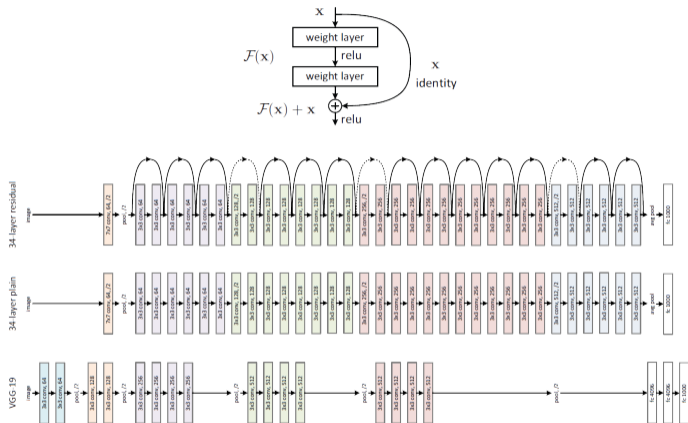




Źródło ilustracji: Zeiler, M.D., Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds)

Computer Vision – ECCV 2014

Sieci ResNet



Źródło ilustracji: K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern

Recognition (CVPR), 2016, pp. 770-778

Zaawansowane architektury – sieci transformers



ChatGPT 4o ▾

 Share

PR

Why is a Raven Like a Writing Desk?



"Why is a raven like a writing desk?" is a riddle posed by the Mad Hatter in Lewis Carroll's "Alice's Adventures in Wonderland." This famous riddle is known for its lack of a definitive answer. Carroll himself later suggested an answer in a preface to an 1896 edition of the book: "Because it can produce a few notes, though they are very flat; and it is nevar put with the wrong end in front!" This answer is a pun, as "nevar" is "raven" spelled backward, though it was later corrected to "never" in subsequent editions, losing the pun.

The riddle has been a subject of much debate and has inspired numerous answers and interpretations over the years, both serious and humorous. It reflects the whimsical and nonsensical tone of the book, where logic and conventional answers are often turned on their head.



Problem „sekwencja w sekwencję” (sequence to sequence), z odległymi zależnościami między słowami

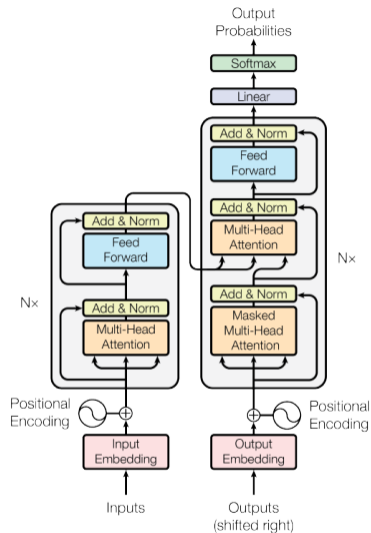
Mary had a little lamb.

In the hospital, Mary had a little lamb.

Krótką (i fragmentaryczną) historia prób rozwiązania:

- ▶ Sieci rekurencyjne
- ▶ Sieci LSTM (Long-Short Term Memory)
- ▶ Sieci transformers

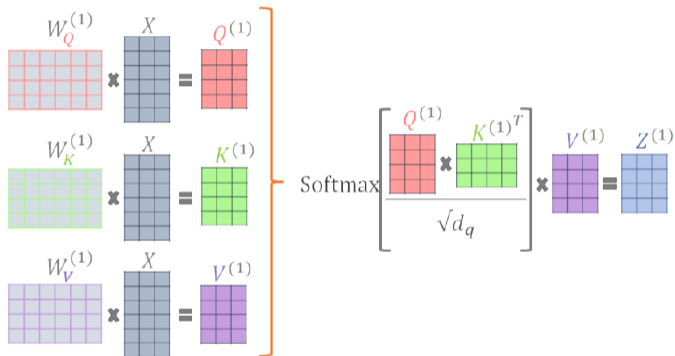




Źródło ilustracji: Vaswani, A. et al (2017). "Attention is All you Need". Advances in Neural Information Processing Systems. 30

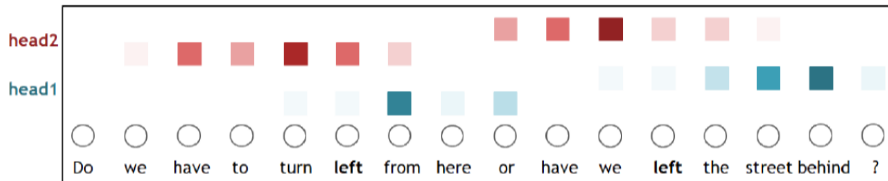


$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$



Źródło ilustracji: Ahmed, S., Nielsen, I.E., Tripathi, A. et al. Transformers in Time-Series Analysis: A Tutorial. Circuits Syst Signal Process 42, 7433-7466 (2023).





Źródło ilustracji: Ahmed, S., Nielsen, I.E., Tripathi, A. et al. Transformers in Time-Series Analysis: A Tutorial. Circuits Syst Signal Process 42, 7433-7466 (2023)

