Propagacja i propagacja wsteczna

Weronika Hryniewska

Propagacja

prediction = out

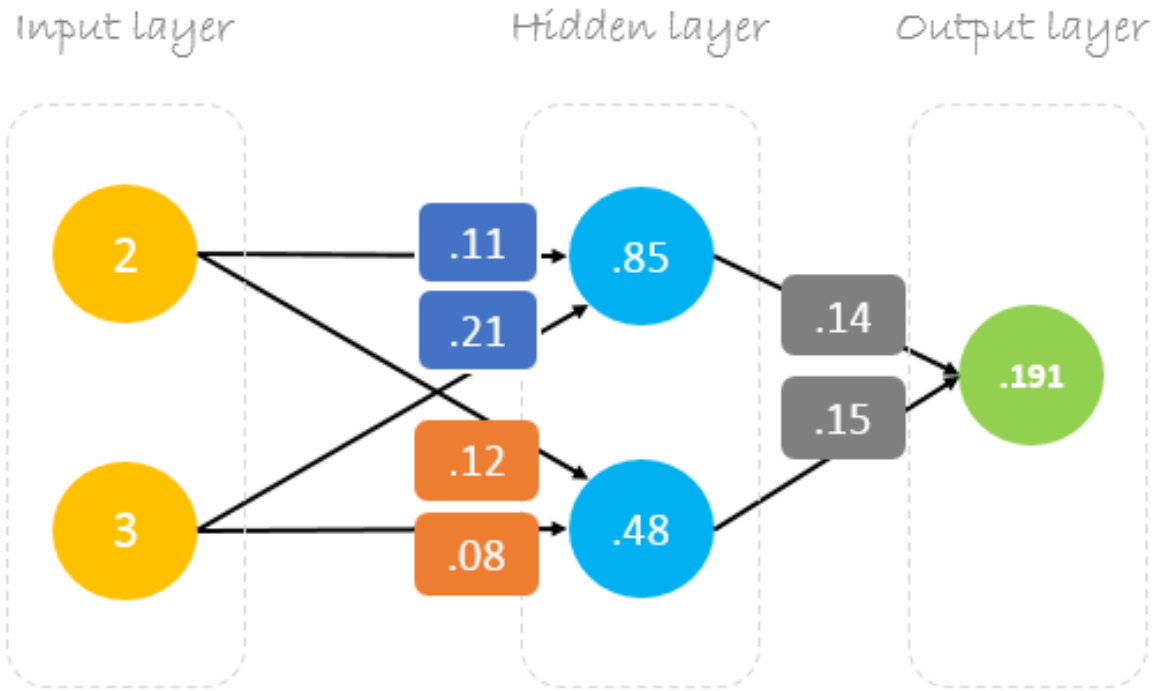
prediction = $(h_1)\, w_5 + (h_2)\, w_6$

$h_1 = i_1 w_1 + i_2 w_2$

$h_2 = i_1 w_3 + i_2 w_4$

prediction = $(i_1\, w_1 + i_2\, w_2)\, w_5 + (i_1\, w_3 + i_2\, w_4)\, w_6$

to change ***prediction*** value,
we need to change ***weights***



Forward Pass

$$[2 \quad 3] \cdot \begin{bmatrix} 0.11 & 0.12 \\ 0.21 & 0.08 \end{bmatrix} = [0.85 \quad 0.48] \cdot \begin{bmatrix} 0.14 \\ 0.15 \end{bmatrix} = [0.191]$$

Matrix multiplication

Details

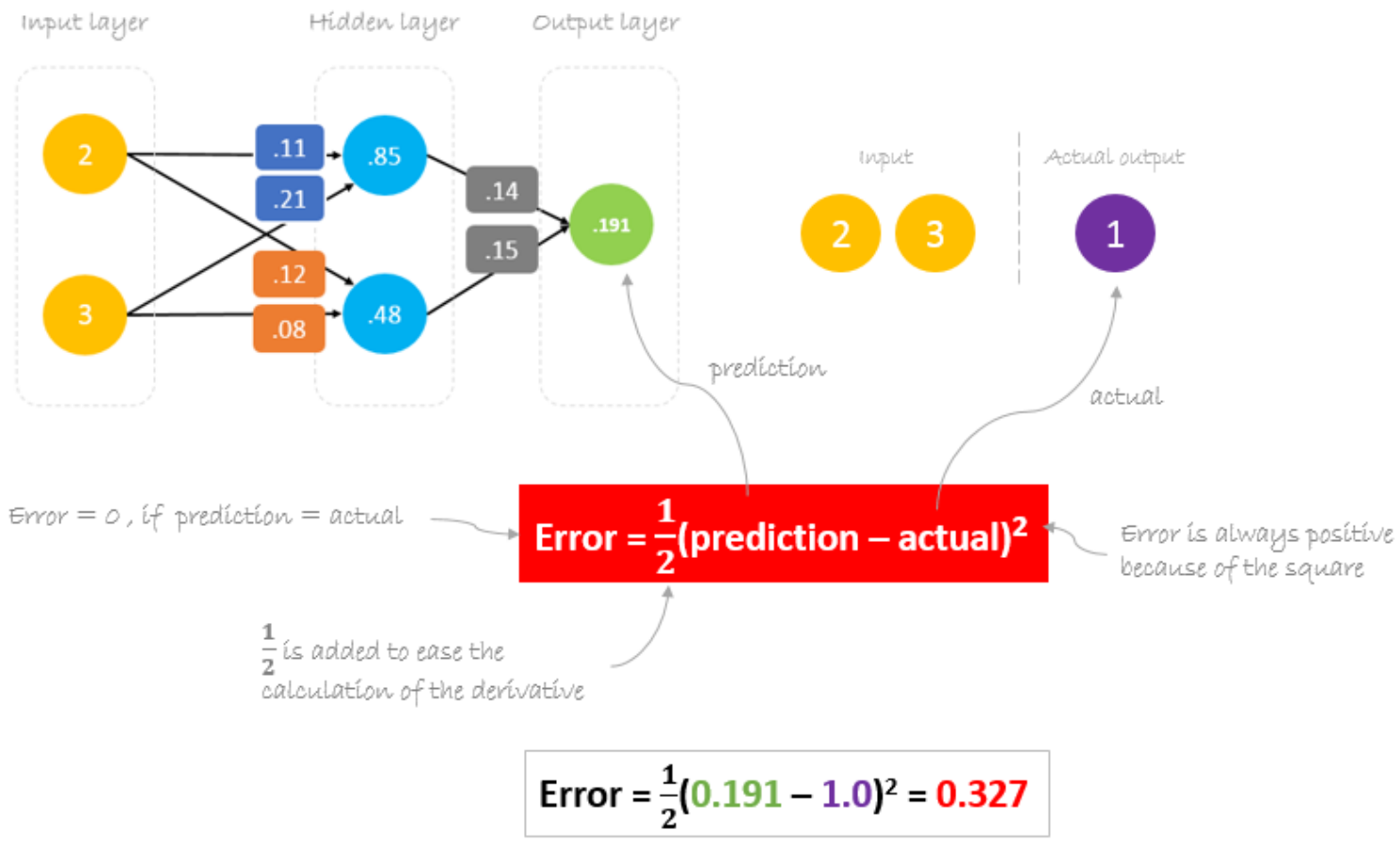
2 x .11 + 3 x .21 = .85

2 x .12 + 3 x .08 = .48

.85 x .14 + .48 x .15 = .191

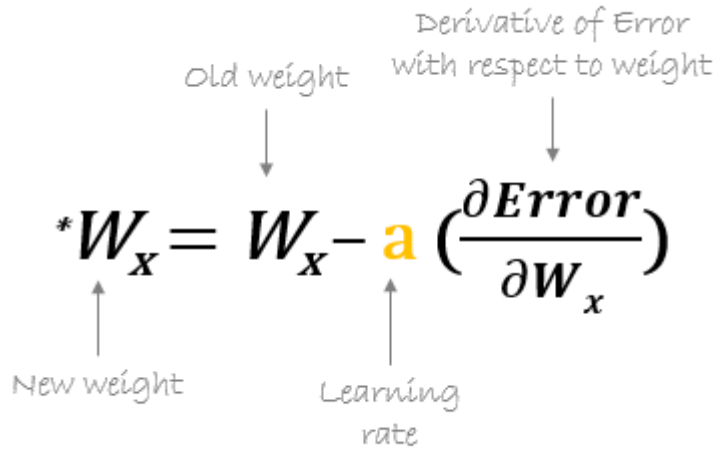
https://hmkcode.com/ai/backpropagation-step-by-step

Obliczanie błędu



https://hmkcode.com/ai/backpropagation-step-by-step

Wsteczna propagacja błędu



$$\frac{\partial Error}{\partial W_6} = \frac{\partial Error}{\partial prediction} * \frac{\partial prediction}{\partial W_6}$$ ← chain rule

Error = $\frac{1}{2}$(prediction – actual)²

prediction = $(i_1 w_1 + i_2 w_2) w_5 + (i_1 w_3 + i_2 w_4) w_6$

h_2

$$\frac{\partial Error}{\partial W_6} = \frac{\frac{1}{2}(predictoin - actula)^2}{\partial prediciton} * \frac{\partial \left(i_1 w_1 + i_2 w_2\right) w_5 + \left(i_1 w_3 + i_2 w_4\right) w_6}{\partial W_6}$$

$$\frac{\partial Error}{\partial W_6} = 2 * \frac{1}{2}(predictoin - actula) \frac{\partial (predictoin - actula)}{\partial prediciton} * (i_1 w_3 + i_2 w_4)$$ ← $h_2 = i_1 w_3 + i_2 w_4$

$$\frac{\partial Error}{\partial W_6} = (predictoin - actula) * (h_2)$$

Δ = prediction – actual ← delta

$$\frac{\partial Error}{\partial W_6} = \Delta h_2$$

https://hmkcode.com/ai/backpropagation-step-by-step

$$\frac{\partial Error}{\partial W_1} = \frac{\partial Error}{\partial prediction} * \frac{\partial prediction}{\partial h_1} * \frac{\partial h_1}{\partial W_1}$$ ← chain rule

Error = $\frac{1}{2}$(prediction – actual)^2

prediction = $(h_1)\, w_5 + (h_2)\, w_6$

$h_1 = i_1 w_1 + i_2 w_2$

$$\frac{\partial Error}{\partial W_1} = \frac{\partial \frac{1}{2}(predictoin - actula)^2}{\partial prediciton} * \frac{\partial\, (h_1)\, w_5 + (h_2)\, w_6}{\partial h_1} * \frac{\partial\, i_1 w_1 + i_2 w_2}{\partial w_1}$$

$$\frac{\partial Error}{\partial W_1} = 2 * \frac{1}{2}(\mathbf{predictoin - actula}) \frac{\partial (predictoin - actula)}{\partial prediciton} * (w_5) * (i_1)$$

$$\frac{\partial Error}{\partial W_1} = (\mathbf{predictoin - actula}) * (w_5 i_1)$$

Δ = prediction – actual ← delta

$$\frac{\partial Error}{\partial W_1} = \Delta\, w_5 i_1$$

https://hmkcode.com/ai/backpropagation-step-by-step

updated weights

$$
\begin{aligned}
{}^{*}w_6 &= w_6 - a\,(h_2 \,.\, \Delta) \\
{}^{*}w_5 &= w_5 - a\,(h_1 \,.\, \Delta) \\
{}^{*}w_4 &= w_4 - a\,(i_2 \,.\, \Delta w_6) \\
{}^{*}w_3 &= w_3 - a\,(i_1 \,.\, \Delta w_6) \\
{}^{*}w_2 &= w_2 - a\,(i_2 \,.\, \Delta w_5) \\
{}^{*}w_1 &= w_1 - a\,(i_1 \,.\, \Delta w_5)
\end{aligned}
$$

$$
\begin{bmatrix} w_5 \\ w_6 \end{bmatrix} = \begin{bmatrix} w_5 \\ w_6 \end{bmatrix} - a\,\Delta \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} w_5 \\ w_6 \end{bmatrix} - \begin{bmatrix} a h_1 \Delta \\ a h_2 \Delta \end{bmatrix}
$$

$$
\begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} = \begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} - a\,\Delta \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} . \begin{bmatrix} w_5 & w_6 \end{bmatrix} = \begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} - \begin{bmatrix} a\,i_1 \Delta w_5 & a\,i_1 \Delta w_6 \\ a\,i_2 \Delta w_5 & a\,i_2 \Delta w_6 \end{bmatrix}
$$

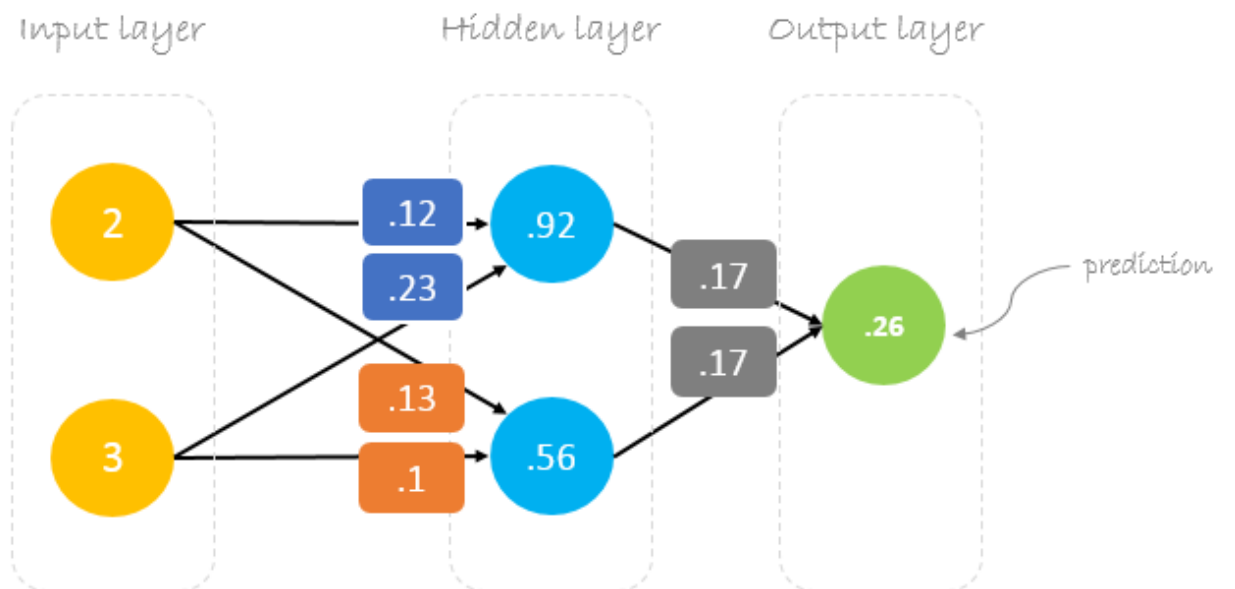
https://hmkcode.com/ai/backpropagation-step-by-step

$\Delta = 0.191 - 1 = -0.809$ ← **Delta** = prediction - actual

$a = 0.05$ ← **Learning rate,** we smartly guess this number

$$\begin{bmatrix} w_5 \\ w_6 \end{bmatrix} = \begin{bmatrix} 0.14 \\ 0.15 \end{bmatrix} - 0.05(-0.809) \begin{bmatrix} 0.85 \\ 0.48 \end{bmatrix} = \begin{bmatrix} 0.14 \\ 0.15 \end{bmatrix} - \begin{bmatrix} -0.034 \\ -0.019 \end{bmatrix} = \begin{bmatrix} 0.17 \\ 0.17 \end{bmatrix}$$

$$\begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} = \begin{bmatrix} .11 & .12 \\ .21 & .08 \end{bmatrix} - 0.05(-0.809) \begin{bmatrix} 2 \\ 3 \end{bmatrix} . \begin{bmatrix} 0.14 & 0.15 \end{bmatrix} = \begin{bmatrix} .11 & .12 \\ .21 & .08 \end{bmatrix} - \begin{bmatrix} -0.011 & -0.012 \\ -0.017 & -0.018 \end{bmatrix} = \begin{bmatrix} .12 & .13 \\ .23 & .10 \end{bmatrix}$$



Matrix multiplication

$$[2 \quad 3] . \begin{bmatrix} 0.12 & 0.13 \\ 0.23 & 0.10 \end{bmatrix} = [0.92 \quad 0.56] . \begin{bmatrix} 0.17 \\ 0.17 \end{bmatrix} = [0.26]$$

Details

2 x .12 + 3 x .23 = .85

2 x .13 + 3 x .10 = .48

.92 x .17 + .56 x .17 = .26

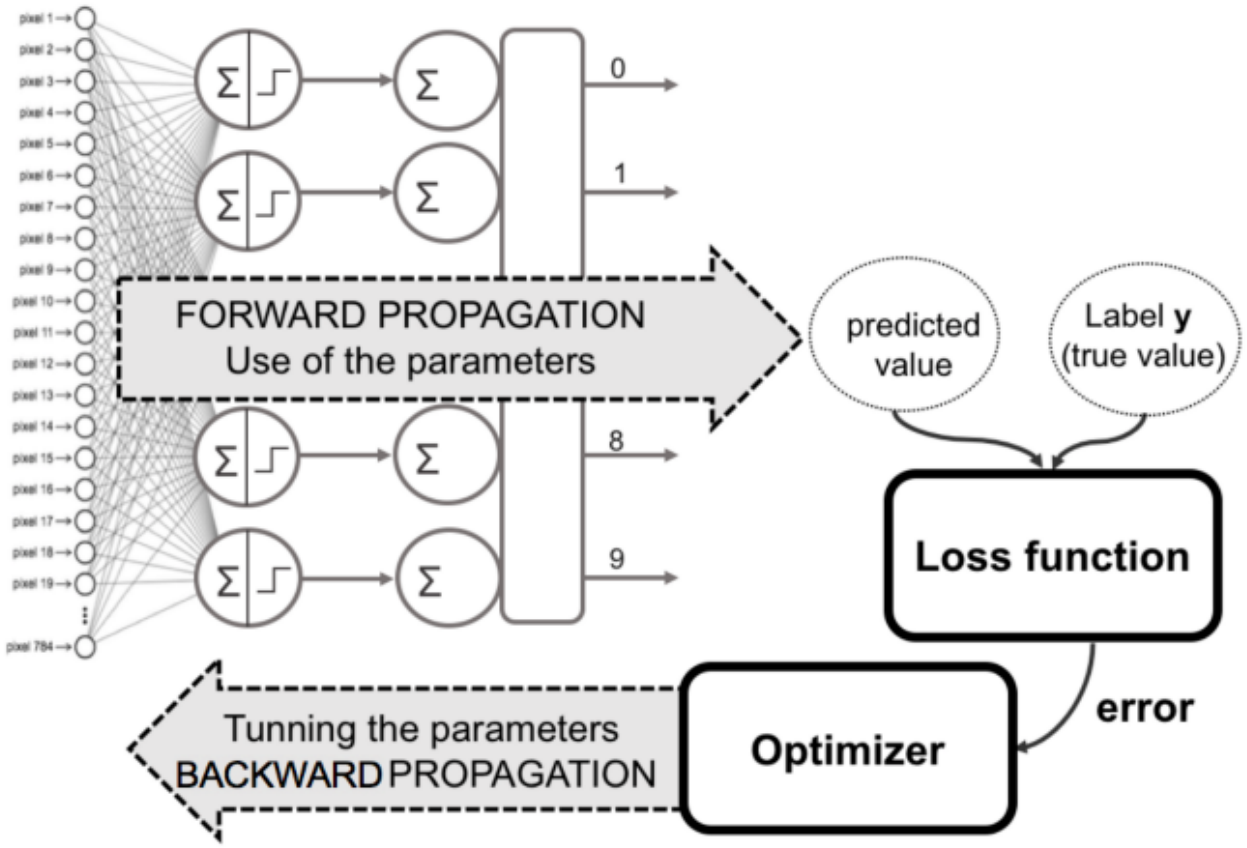
https://hmkcode.com/ai/backpropagation-step-by-step

Komponenty propagacji wstecznej

Weronika Hryniewska

```
model.compile(
loss='categorical_crossentropy',
optimizer='sgd',
metrics=['accuracy']
)
```

x



https://ichi.pro/pl/proces-uczenia-sie-glebokiej-sieci-neuronowej-99606345524817

Funkcja straty

- Wyliczana przez ostatnią warstwę sieci
- Stara się minimalizować błąd poprzez zmianę wag neuronów
- Musi być różniczkowalna

Przykłady funkcji straty

- Regression Loss Functions
  - Mean Squared Error Loss
  - Mean Squared Logarithmic Error Loss – wartość docelowa ma szeroki zakres wartości
  - Mean Absolute Error Loss – duże lub małe wartości oddalone od wartości średniej
- Binary Classification Loss Functions
  - Binary Cross-Entropy – preferowana, wartości docelowe w zbiorze {0, 1}.
  - Hinge Loss – głównie do SVM, wartości docelowe w zbiorze {-1, 1}.
  - Squared Hinge Loss
- Multi-Class Classification Loss Functions
  - Multi-Class Cross-Entropy Loss – preferowana, wartości docelowe w zbiorze {0, 1, ..., n}, gdzie każdej klasie przypisana jest unikalna wartość całkowita
  - Sparse Multiclass Cross-Entropy Loss – klasyfikacja z dużą ilością etykiet
  - Kullback Leibler Divergence Loss - rekonstrukcja oryginalnych danych wejściowych

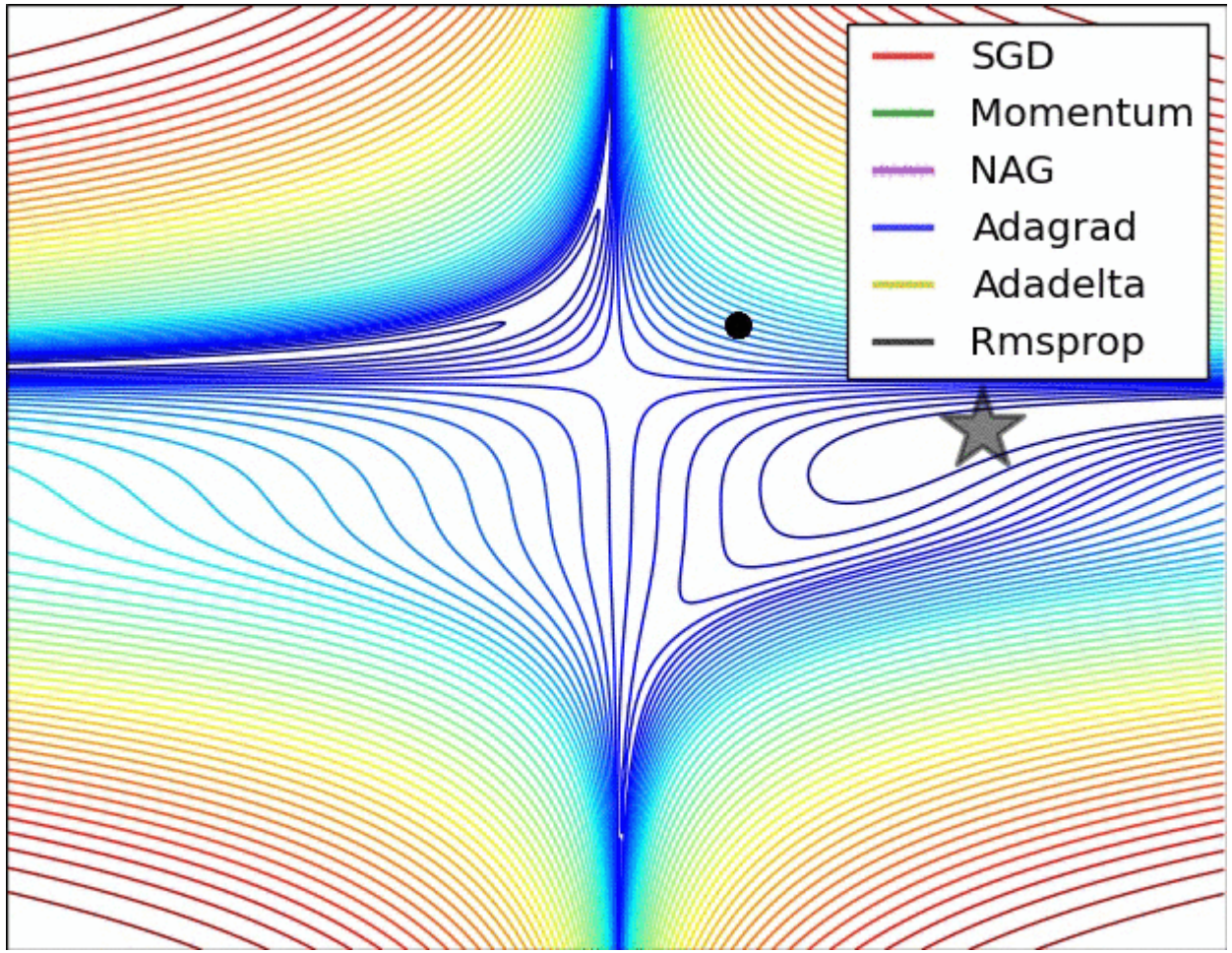
https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks

Optymalizator

- Algorytm aktualizujący wagi
- uwagi mogą być aktualizowane po każdym przykładzie (online) lub po grupie przykładów (offline, batch)

Przykłady optymalizatorów

- SGD
- Adagrad
- RMSprop
- AdaDelta
- Adam (RMSprop + Momentum)
- AdaMax
- Nadam



www.dlology.com/blog/quick-notes-on-how-to-choose-optimizer-in-keras