Projektarbeit NN

Florian Braun

Inhaltsverzeichnis

Fehlerfunktion MAE

- Durchschnitt des Betrages der Differenz
- Änderung des Fehlers ist linear
- Beispiel: $y1 = \{0\}$, $\hat{y}1 = \{1\}$, $y2 = \{0\}$, $\hat{y}2 = \{2\}$
- $E1 = |0-1| = 1$
- $E2 = |0-2| = 2$
-

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i|$$

Vorteile: geringer Rechenaufwand

Fehlerfunktion MSE

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

Quadratische Abweichung

Änderung des Fehlers ist quadratisch

- Beispiel: *y1 = {0}, ŷ1 = {1}, y2 = {0}, ŷ2 = {2}*
- *E1 = (0 - 1)^2 = 1*
- *E2 = (0- 2)^2 = 4*

Aufwändiger als MAE durch Quadrierung

Abweichungen haben größeren Einfluss als bei MAE

Fehlerfunktion RMSE

- Ähnlich wie MSE
- Änderung über $\hat{y}$ ist geringer
- Beispiel: $y1 = \{0\}$, $\hat{y}1 = \{1\}$, $y2 = \{0\}$, $\hat{y}2 = \{2\}$
- *E1 = \(0 - 1)^2 = 1*
- *E2 = \(0- 2)^2 = 2*

Hoher Rechenaufwand

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

Delta-Regel

Bestimmt die Änderung für ein Gewicht bei einem einzelnen Perzeptron

w_{ij}_neu = w_{ij}_alt + w_delta

w_delta = f’ * -n * e_j * o_i

n: Lernrate

e : Fehler eines Neurons

o : Ausgabe eines Neurons

f’ : Ableitung der Aktivierungsfunktion

Backpropagation

Bestimmt die Änderung für ein Gewicht bei einem MLP

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = -\eta \delta_j o_i$$

mit

$$\delta_j = \begin{cases} \varphi'(\text{net}_j)(o_j - t_j) & \text{falls } j \text{ Ausgabeneuron ist,} \\ \varphi'(\text{net}_j) \sum_k \delta_k w_{jk} & \text{falls } j \text{ verdecktes Neuron ist.} \end{cases}$$

Dabei ist

Δw_{ij} die Änderung des Gewichts w_{ij} der Verbindung von Neuron i zu Neuron j,

η eine feste Lernrate, mit der die Stärke der Gewichtsänderungen bestimmt werden kann,

δ_j das Fehlersignal des Neurons j, entsprechend zu $\frac{\partial E}{\partial \text{net}_j}$,

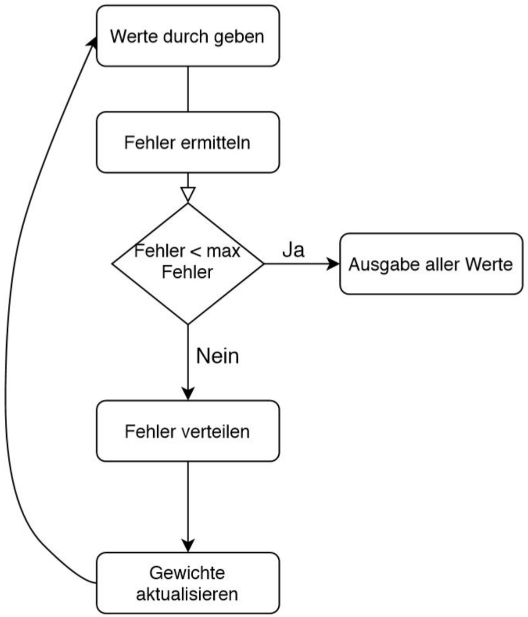
t_j die Soll-Ausgabe des Ausgabeneurons j,

o_i die Ausgabe des Neurons i,

o_j die Ist-Ausgabe des Ausgabeneurons j und

k der Index der nachfolgenden Neuronen von j.

Funktion des Neuronalen Netzes im Programm



Funktion des Neuronalen Netzes im Programm

```python
def train(self, inp, expected):
    self.output = self.pass_values(expected, inp)
    total_error = self.calculate_total_error(self.output, expected)
    self.total_error.append(total_error)
    self.back_propagation(total_error)
    self.update_weights()
    self.clear_e()
```

Funktion des Neuronalen Netzes im Programm

```python
def back_propagation(self, total_error):
    key_list = list(self.neurons.keys())
    key_list.reverse()
    for key in key_list:
        for neuron in self.neurons[key]:
            if key == key_list[0]:
                neuron.e = neuron.activation_function_derivatives(neuron.scalar_product()) * total_error

            for input in neuron.get_input_cnts():
                neuron2 = input.get_input_neuron()
                neuron2.e += neuron.e * input.get_weight() * neuron2.activation_function_derivatives(
                    neuron2.scalar_product())
```

Funktion des Neuronalen Netzes im Programm

```python
def update_weights(self):
    for key in list(self.neurons.keys()):
        for neuron in self.neurons[key]:
            for cnt in neuron.get_input_cnts():
                inp = cnt.get_input_neuron().generate_output()
                cnt.update_weight(-neuron.e * self.eta * inp)
```

Quellen

https://medium.com/@george.drakos62/how-to-select-the-right-evaluation-metric-for-machine-learning-models-part-1-regrression-metrics-3606e25beae0

https://de.wikipedia.org/wiki/Backpropagation