Induktives Lernen anhand neuronaler Netze

Bernd Porr

25. April 2022

1 Induktives lernen

Bekannt ist ein Datensatz $\vec{x}_1, \ldots, \vec{x}_N$, der mit einer unbekannten Funktion

$$\vec{y} = f(\vec{x}) \tag{1}$$

in einen Datensatz $\vec{y}$ überführt wird. Das Ziel ist es, die Funktion f iterativ mit Hilfe von Beispielen $\vec{x}, \vec{y}$ zu lernen, so dass dann neue $\vec{x}_{N+1}, \ldots$ korrekt in $\vec{y} = f(\vec{x})$ überführt werden.

In den folgenden Abschnitten wird die Funktion f durch immer komplexere neuronale Netzwerke gelernt und approximiert.

2 Lineares Neuron

Abb. 1A zeigt ein einfaches lineares Neuron in einer einzigen Schicht. Die Aktivität des Neurons ist $y(n)$ und hat einen Index für dessen Position in der Schicht, wobei der Index i hier die Ausgangsschicht repräsentiert. n ist der Momentane Zeitpunkt als Zeit-Index in einem getakteten System. Die Eingangsaktivität $x_j(n) = y_j(n)$ wird mit ω_{ji} gewichtet und dann zum Ausgangssignal $y_i(n)$ summiert:

$$y_i(n) = \sum_j y_j(n) w_{ji} \tag{2}$$

Die Funktion f ist hier also eine einfache lineare Kombination von den Eingangssignalen, wobei die Gewichte ω_{ji} gelernt werden müssen, um die spezifische Funktion f zu approximieren.

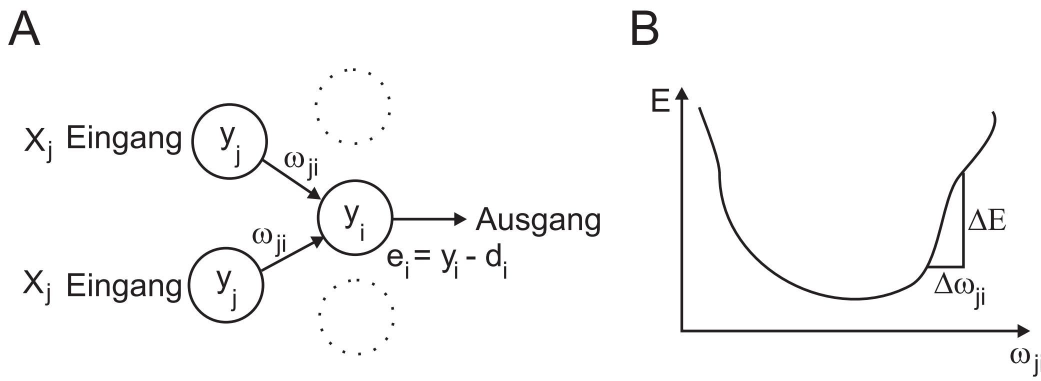

Abbildung 1: Einfaches lineares Neuron

Gelernt werden soll eine bestimmte Ausgangsaktivität in Abhängigkeit von einer Eingangsaktivität (Gl. 1). Das wird erreicht, indem ein Fehler $e_i(n)$ am Ausgang jedes Neurons etabliert wird, der dann die Gewichte w_{ji} schrittweise dorthin bewegt, so dass er im Mittel Null wird:

$$e_i(n) = y_i(n) - d_i(n) \tag{3}$$

wobei $d_i(n)$ der gewünschte Ausgangswert ist und $y_i(n)$ der Aktuelle. Ziel des Lernens ist es, dass das quadratische Mittel vom Fehler Null wird:

$$E = \frac{1}{2}e^2 \tag{4}$$

Der zentrale Trick ist nun, die Partielle Ableitung in Abhängigkeit vom Fehler E zu nehmen und damit die Gewichte zu verändern:

$$\Delta\omega_{ji} = -\mu\frac{\partial E}{\partial \omega_{ji}} \tag{5}$$

$$\omega_{ji} \leftarrow \omega_{ji} + \Delta\omega_{ji} \tag{6}$$

Warum macht das Sinn? In Abb. 1B ist die Abhängigkeit von einem Gewicht ω_{ji} gezeigt. Wenn in diesem Beispiel das Gewicht etwas erhöht wird, dann wird der Fehler E größer. Also war die Erhöhung kontraproduktiv und es ist besser, das Gewicht wieder zu verringern. Wenn aber umgekehrt eine Erhöhung von ω_{ji} eine Verringerung von E bewirkt, dann ist das Vorteilhaft also kann man das Gewicht erhoehen. Iterativ wird also der Fehler E verringert.

Als nächsten Schritt muss nun die Lernregel hergeleitet werden. Das geschieht, indem die Gln. 2, 3 und 4 in Gl. 5 einsetzt werden:

$$\begin{aligned}
\Delta\omega_{ji} &= -\mu\frac{1}{2}\frac{\partial(d_i(n)-y_i(n))^2}{\partial\omega_{ji}} && (7)\\
&= -\mu\frac{1}{2}\frac{\partial\left(d_i(n)-\sum_j y_j(n)w_{ji}\right)^2}{\partial\omega_{ji}} && (8)\\
&= \mu\underbrace{\left(d_i(n)-\sum_j y_j(n)w_{ji}\right)}_{-e_i(n)}\cdot y_j(n) && (9)\\
&= -\mu\underbrace{\frac{\partial E}{\partial y_i}}_{-e_i(n)}\underbrace{\frac{\partial y_i}{\partial\omega_{ji}}}_{y_j(n)} && (10)\\
&= \mu\cdot e_i(n)\cdot y_j(n) && (11)
\end{aligned}$$

Die Lernregel ist also einfach die Multiplikation vom Fehler am Ausgang mit der Aktivität am Eingang. Dieses Produkt verändert dann das zugehörige Gewicht.

3 Mehrschichtiges lineares Netzwerk

Abb. 2 zeigt nun ein Netzwerk mit mehreren Schichten. Wie das Signal vom Eingang zum Ausgang durch die verschiedenen Schichten läuft, ist einfach auszurechnen. Auch der Fehler e_i in der Ausgangsschicht kann weiterhin einfach als Differenz zwischen y_i und d_i ausgerechnet werden (siehe Gl. 3). Das Problem ist, wie die *internen* Fehler e_j und e_k ausgerechnet werden können. Der interne Fehler für ω_{kj} ist erstmal genauso definiert wie der Fehler am Ausgang (siehe Gl. 10):

$$\frac{\partial E}{\partial\omega_{kj}} = \underbrace{\frac{\partial E}{\partial y_j}}_{\text{Trick!}}\frac{\partial y_j}{\partial\omega_{kj}} \tag{12}$$

aber der ist ja nicht direkt verfügbar. Der Trick ist nun den Term $\frac{\partial E}{\partial y_j}$ mit Hilfe von der Aktivität y_i am Ausgang auszudrücken und dann einfach die beiden Terme

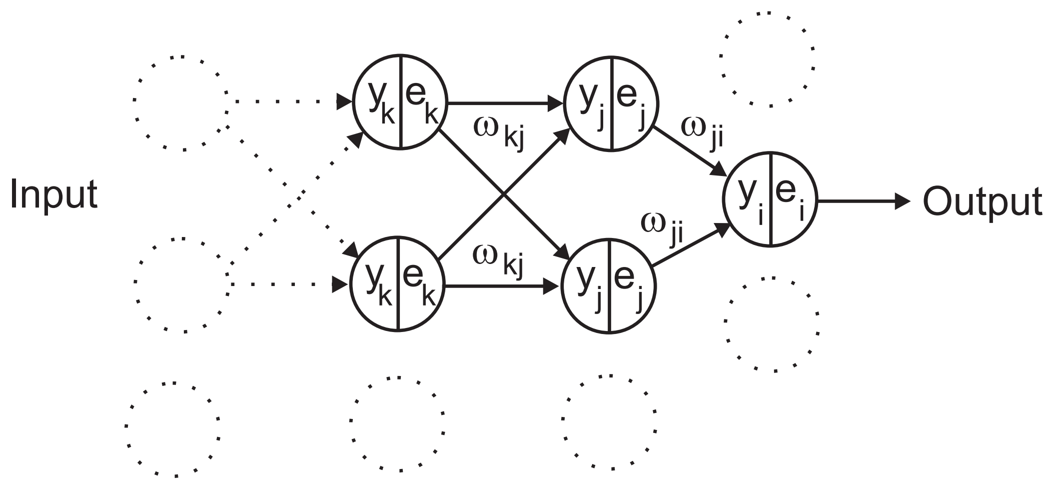

Abbildung 2: Mehrschichtiges Feedforward Netz

mit Gl. 2 und Gl. 10 zu identifizieren:

$$\frac{\partial E}{\partial y_j} = \sum_i \underbrace{\frac{\partial E}{\partial y_i}}_{-e_i} \cdot \underbrace{\frac{\partial y_i}{\partial y_j}}_{\omega_{ji}} \tag{13}$$

Gl. 13 wieder in Gl. 12 substituiert gibt dann:

$$\frac{\partial E}{\partial \omega_{kj}} = -\left(\sum_i e_i \omega_{ji}\right) \frac{\partial y_j}{\partial \omega_{kj}} \tag{14}$$

mit

$$y_j = \sum_k y_k(n) w_{kj} \tag{15}$$

führt das zu:

$$\frac{\partial E}{\partial \omega_{kj}} = -\left(\sum_i e_i \omega_{ji}\right) \underbrace{\frac{\partial \left(\sum_k y_k(n) w_{kj}\right)}{\partial \omega_{kj}}}_{y_k} \tag{16}$$

$$= -\left(\sum_i e_i \omega_{ji}\right) y_k \tag{17}$$

Die Änderung des internen Gewichtes ω_{kj} folgt dann folgender Gleichung:

$$\Delta\omega_{kj} = \mu \cdot y_k \cdot \underbrace{\sum_i e_i \omega_{ji}}_{e_j} \tag{18}$$

Wobei nun e_j der interne Error ist und damit ist es möglich, alle internen Fehler rückwärts vom Ausgang zum Eingang zu berechnen.

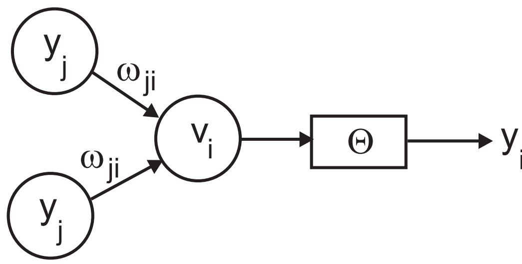

Abbildung 3: Neuron mit nichtlinearer Characteristik

4 Nicht-lineare Netzwerke

Nur wenn Netzwerke nicht-linear sind, machen tiefere Schichten einen Sinn, da jedes lineare Netzwerk immer auf eine Schicht reduziert werden kann. Erst die Nichtlinearität rechtfertigt tiefe ("deep") mehrschichtige Netzwerke. Abb. 3 zeigt ein Neuron von solch einem mehrschichtigen Netzwerk, wo die lineare Summe dann durch eine Nichtlinearität geschickt wird:

$$y_i(n) = \Theta\left(\underbrace{\sum_j y_j(n) w_{ji}}_{v_i}\right) \tag{19}$$

Was aendert sich in diesem Fall an den Lernregeln? Sehr wenig. Wenn man sich Gl. 10 ansieht, merkt man, dass einfach ein weiterer Term durch die Kettenregel hinzukommt:

$$\frac{\partial E}{\partial \omega_{ji}} = \frac{\partial E}{\partial y_i} \underbrace{\frac{\partial y_i}{\partial v_i}}_{\Theta'} \frac{\partial v_i}{\partial \omega_{ji}} \tag{20}$$

wo Θ' einfach die Ableitung der nicht-linearen Aktivierungsfunktion Θ ist. Das heißt, dass der Fehler für die Ausgangsschicht jetzt folgendermaßen ausgerechnet wird:

$$\Delta\omega_{ji} = \mu \cdot \Theta'(y_i) \cdot y_j \cdot e_i \tag{21}$$

und der interne Fehler so:

$$\Delta\omega_{kj} = \mu \cdot \Theta'(y_j) \cdot y_k \cdot \underbrace{\sum_i e_i \omega_{ji}}_{e_j} \tag{22}$$

Strikt muss die nicht-lineare Funktion Θ differenzierbar sein. Es hat sich aber herausgestellt, dass der Einweggleichrichter (Rectifiying Linear Unit = ReLU)

$$\Theta(v) = \begin{cases} 0, & \text{falls } v < 0 \\ v, & \text{ansonsten} \end{cases} \tag{23}$$

für viele Klassifizierungen am besten ist. Dessen Ableitung hat keine eindeutige Lösung am Ursprung so dass man sich dort entscheiden muss, ob sie eins oder Null ist. Andere populäre nicht-lineare Funktionen sind $\Theta(v) = \tanh(v)$ oder die sigmoide Funktion: $\Theta(v) = \frac{1}{1+e^{-v}}$.