1 Künstliche neuronale Netzwerke

1.1 Allgemeine Beschreibung

Künstliche neuronale Netze sind dabei Modelle aus vielen binären Klassifizierern, die in Schichten gruppiert werden. Dabei gibt es einen Input-Layer, also eine Input-Schicht, in der die einzelnen Daten eingespeißt werden, die Hidden-Layer, also die verdeckten zu-trainierenden Schichten, und den Output-Layer, der das Ergebis zurüchgibt. Man spricht von tiefen neuronalen Netzen bei mehr als einer verdeckten Schicht. Dabei wird oft die One-Hot-Kodierung verwendet, also ein Merkmalsvektor $a \in \mathbb{R}$ der für jede Klasse eine Zeile hat und nur in der Zeile jeweils eine 1 hat, zu der der Merkmalspunkt gehört. In den anderen Zeilen des Vektors steht dann immer jeweils eine 0.

1.2 Mathematische Formulierung

Es erfolgt eine Eingabe der Trainingsdaten durch einen Inputlayer. Die Aktivierung erfolgt dann durch die Aktivierungsfunktion ϕ.

$$a_1^2 = \phi(a^{(1)} \cdot \omega^{(1)}) \tag{1}$$

Dabei bezeichnen $\omega^{T(1)} \in \mathbb{R}^{h \times [m+1]}$ die Gewichtungsmatrix und $a_1 \in \mathbb{R}^{[m+1] \times 1}$ ein Aktivierungsvektor. Die Gewichtsmatrix gibt jeweils die Gewichte der einzelnen Verbindungen zwischen zwei Schichten an. Damit gibt es es für jede zwei Schichten eine Matrix. Die einzelnen Punkte müssen aber alle miteinander verbunden sein, damit jede Stelle dieser Matrix einen Eintrag hat. Dabei ist h die Anzahl der verdeckten Schichten und $m + 1$ die Anzahl der Merkmale plus der Bias. Die Aktivierungsfunkton kann frei gewählt werden, es wird allerdings bei den meisten Fällen die logistische Aktivierungsfunktion, die Sigmoid-Funktion, genutzt.

$$\phi(z) = \frac{1}{1 + e^{-z}} \tag{2}$$

Für spätere Zwecke wird hier die Ableitung der Sigmoidfunktion explizit hergeleitet. Nach der Polynom- und Kettenregel gilt:

$$\frac{\partial \phi(z)}{\partial z} = -\frac{(-e^{-z})}{(1 + e^{-z})^2} \tag{3}$$

$$\frac{\partial \phi(z)}{\partial z} = \frac{1 + e^{-z}}{(1 + e^{-z})^2} - \frac{1}{(1 + e^{-z})^2} \tag{4}$$

Dies lässt sich weiter faktorisieren:

$$\frac{\partial \phi(z)}{\partial z} = \frac{1}{1+e^{-z}}(1 - \frac{1}{1+e^{-z}}) \tag{5}$$

$$\frac{\partial \phi(z)}{\partial z} = \phi(z)(1 - \phi(z)) \tag{6}$$

Nun verallgemeinern wir die Regel der Aktivierung:

$$a_1^{n+1} = \phi(a^{(n)} \cdot \omega^{T(n)}) \tag{7}$$

Die Straffungsfunktion wird ebenfalls aus der logistischen Regression übernommen.

$$J(w) = -\sum_{i=1}^{n} y^{(i)} log(a^i) + (1 - y^{(i)}) log(1 - a^{(i)}) \tag{8}$$

Da wir es hier allerdings nicht mehr mit einem Gewicht, sondern mit einem Gewichtsvektor zu tuen haben muss die Formel auf alle Gewichte angepasst werden.

$$J(w) = -\sum_{i=1}^{n} \sum_{j=1}^{t} y_j^{(i)} log(a_j^i) + (1 - y_j^{(i)}) log(1 - a_j^{(i)}) \tag{9}$$

Um nun diesen Term zu minimieren muss die Strafffunktion optimiert werden. Dies geschieht nach bekanntem Verfahren.
Zur Fehlermessung wird der Fehlern entweder Vorwärts oder Rückwärts berechnet. Die Rückwartsberechnung wird hier deswegen gewählt, da dort zur Berechnung der Ableitung ein Matrix-Vektor-Produkt anstatt eines Matrix-Matrix-Produktes ausgerechnet werden muss, was deutlich weniger rechenintensiv ist. Die Rückwärtsberechnung erfolgt zuerst über die Differenz der Ausgabe des Klassenvektors mit der vorhergesagten Bezeichnung.

$$\zeta^t = a^t - y \tag{10}$$

Danach berechnet sich der Fehler rekursiv zu:

$$\zeta^{n-1} = W^{T(n-1)} \zeta^n \frac{\partial \phi(z^{(n-1)})}{\partial z^{(n-1)}} \tag{11}$$

Die Ableitung dieser Funktion wurde bereits oben bestimmt. Daraus folgt.

$$\zeta^{n-1} = W^{T(n-1)}\zeta^n \phi(z^{n-1}(1 - \phi(z^{n-1})) \tag{12}$$

Mit dieser Formel berechnet sich die Gewichtsänderung nach dem bekannten Verfahren zu:

$$\Delta^l_{i,j} := \Delta^l_{i,j} + a^l_j \zeta^{l+1}_i \tag{13}$$

Dieses gesamte Verfahren zum Trainieren eines neuronalen Netzwerkes ist also die mehrdimensionale Variante des Adaline-Modells oder der logistitschen Regression. Diese verallgemeinerten Formeln gelten dann ebenfalls für nur eine Gewichtungsmatrix.