1 Das MCP-Modell

Dieses sehr frühe Modell des maschinellen Lernens beherrscht eine binäre Klassifizerung der Objekte. Entweder ist das zu-klassifizierende Objekt ein Objekt dieser Klasse oder es ist ein Objekt einer anderen Klasse. Ein Modell dieser Art der Klassifizierung ist das MCP-Modell. Das MCP-Neuronenmodell basiert auf dem Modell einer Nervenzelle. Eine biologische Nervenzelle wird erst dann bei elektrischen Signalen aktiviert, wenn der Stromfluss einen gewissen Schwellenwert erreicht hat und mathematisch wird dies mit der Heaviside-Funktion modelliert.

$$\phi(z) = \begin{cases} 1, & z \geq 0 \\ -1, & z < 0 \end{cases} \tag{1}$$

Kann das zu-klassifizierende Objekt in quantitative Eigenschaften übersetzt und Diese normiert werden, kann hieraus eine Matrix $t \in \mathbb{R}^{n+1\times 1}$gemacht werden. Dabei wird jedes Feature, also jede beschriebene Eigenschaft einer Reihe zugeordnet. Damit beinhaltet jede Zeile die gesamten Informationen eines Datenpakets. Dabei ist x_i die i-te Eigenschaft des Objektes. In dieser Variante wird x_0 als 1 definiert. Dieser hat den Zweck in dem Skalarprodukt den Schwellenwert wieder auszugeben.

$$t = (1, x_1, ..., x_{n-1}, x_n) \tag{2}$$

Diese Matrix wird nun mit einem Gewichtsvektor $\omega \in R^{1\times n+1}$multipliziert. Dieser Gewichtsvektor ist ein Vektor , wobei jeder Eintrag ein individuelles Gewicht für jede Eigenschaft festlegt. Der nullte Eintrag ist hierbei der Schwellenwert z, der so gespeichert wird.

$$\omega = \begin{pmatrix} -z \\ y_1 \\ ... \\ y_{n-1} \\ y_n \end{pmatrix} \tag{3}$$

Für die Multiplikation gilt also nach den Regeln der linearen Algebra:

$$V^{m\times n}.W^{n\times k} = P^{m\times k} : \forall P_{g,h} : P_{g,h} = \sum_{1\leq i\leq n} V_{g,i}W_{i,h} \implies \omega{\cdot}t = -z + \sum_{1\leq i\leq n} x_i y_i \tag{4}$$

Damit lässt sich aus den Gewichten die Heaviside-Funktion neu definieren.

$$\phi(\omega \cdot t) = \begin{cases} 1, & \sum_{1 \leq i \leq n} x_i y_i \geq z \\ -1, & \sum_{1 \leq i \leq n} x_i y_i < z \end{cases} \tag{5}$$

Nun werden diese Schritte iterativ mit der Perzeptronen-Lernregel wiederholt und die Gewichte in ω angepasst. Die Berechnung des $k+1$-ten Gewichtes ist gegeben durch, wobei y das richtige Ergebnis ist und η die Lernrate ist:

$$w_{i,k+1} := w_{i,k} + \eta(y - \phi(\omega \cdot t))x_i \tag{6}$$

Wird hier also die richtige Eingabe durch die Heaviside-Funktion gegeben, ergbt sich automatisch eine Änderung des Gewichts von Null.

$$w_{i,k+1} := w_{i,k} + \eta(y - y)x_i \tag{7}$$

$$w_{i,k+1} := w_{i,k} \tag{8}$$

Ist das Ergebnis allerdings nicht das Richtige, so gibt es zwei Fälle. Wurde es als richtig vorhergesagt, war aber falsch, dann gilt:

$$w_{i,k+1} := w_{i,k} + \eta(-1 - 1)x_i \tag{9}$$

$$w_{i,k+1} := w_{i,k} - 2\eta x_i \tag{10}$$

Wurde es als falsch vorhergesagt, war aber richtig, dann gilt:

$$w_{i,k+1} := w_{i,k} + \eta(1 - -1)x_i \tag{11}$$

$$w_{i,k+1} := w_{i,k} + 2\eta x_i \tag{12}$$

Die jeweilige Änderung hängt also von der Lernrate und dem jeweiligen Wert der Eigenschaft ab. Umso größer beide Werte sind, umso größer und umso radikaler ist also die Änderung. Durch diesen Algorithmus für die Berechnung werden die Werte der Gewichte über die Iterationen verfeinert. Nach dem Trainieren kann nun mithilfe der Gewichte eine Eingabe bewertet werden. Durch die Verwendung validierender Daten wird das System getestet. Ist es dann erfolgreich genug, so wird das System angenommen.