Blatt MLP

Rashad Gasimov 04.01.2026

NN.MLP.01: Perzeptron-Netze:

Der blau-graue Bereich entspricht

$(x1 \geq 2) \vee (x2 \geq 3)$

Es wird ein Netz mit drei Perzeptrons verwendet (Aktivierungsfunktion: sign:

1. **Perzeptron 1:**

$y1 = \text{sign}(x1 - 2)$

$w = (1,0),\ b = -2$

2. **Perzeptron 2:**

$y2 = \text{sign}(x2 - 3)$

$w = (0,1),\ b = -3$

3. **Perzeptron 3 (ODER):**

$y = \text{sign}(y1 + y2 + 1)$

$w = (1,1),\ b = 1$

Das Netz klassifiziert genau die blau-grauen Bereiche mit **+1**.

NN.MLP.02: Vorwärtslauf im MLP:

Das gegebene MLP besteht aus 25 Eingabezellen, zwei versteckten Schichten mit 64 bzw. 32 Neuronen und einer Ausgabeschicht mit 4 Neuronen (Bias-Zellen nicht mitgezählt).

Die Gewichtsmatrizen und Bias-Vektoren haben daher folgende Dimensionen:

$W^{[1]} \in \mathbb{R}^{64 \times 25},\ b^{[1]} \in \mathbb{R}^{64}$

$W^{[2]} \in \mathbb{R}^{32 \times 64},\ b^{[2]} \in \mathbb{R}^{32}$

$W^{[3]} \in \mathbb{R}^{4 \times 32},\ b^{[3]} \in \mathbb{R}^{4}$

Der Vorwärtslauf des Netzes erfolgt schichtweise. Zunächst wird die Eingabe mit der ersten Gewichtsmatrix multipliziert, der Bias addiert und anschließend die ReLU-Aktivierungsfunktion angewendet. Dasselbe geschieht in der zweiten versteckten Schicht. Die Ausgabe berechnet sich somit zu:

$a^{[1]} = \mathrm{ReLU}(W^{[1]}x + b^{[1]})$

$a^{[2]} = \mathrm{ReLU}(W^{[2]}a^{[1]} + b^{[2]})$

$y = \mathrm{ReLU}(W^{[3]}a^{[2]} + b^{[3]})$

Die Ausgabe ist ein Vektor mit vier Komponenten. Dieser kann beispielsweise als Ergebnis einer Klassifikation mit vier Klassen interpretiert werden, alternativ auch als Regressionsausgabe mit vier Zielwerten.

NN.MLP.03: Tensorflow Playground:

Logistische Regression

Zunächst wurde eine logistische Regression auf dem Gaussian-Datensatz trainiert. Da dieser Datensatz linear separierbar ist, konnte das Modell sehr schnell eine passende lineare Entscheidungsgrenze lernen. Sowohl Trainings- als auch Testkosten sind niedrig, eine Überanpassung ist nicht zu beobachten.

Bei den anderen Datensätzen wie Circle oder Spiral reicht die logistische Regression nicht aus. Die Entscheidungsgrenze bleibt linear und kann die Daten nicht korrekt trennen. Entsprechend bleiben die Kosten hoch und viele Punkte werden falsch klassifiziert.

MLP mit unterschiedlichen Netzarchitekturen

Mit zunehmender Anzahl an Neuronen und Schichten wird die Entscheidungsgrenze deutlich komplexer. Kleine Netze mit nur wenigen Neuronen zeigen vor allem beim Spiral-Datensatz ein klares Underfitting. Größere und tiefere Netze können sowohl den Kreis- als auch den Spiral-Datensatz immer besser modellieren und erzeugen stark nichtlineare Entscheidungsgrenzen.

Sehr tiefe Netze sind zwar leistungsfähig, reagieren aber auch empfindlicher auf Störungen im Datensatz.

Einfluss der Aktivierungsfunktion

Die Wahl der Aktivierungsfunktion hat einen klaren Einfluss:

ReLU trainiert am schnellsten und liefert klare, kantige Entscheidungsgrenzen.

tanh erzeugt glattere Grenzen, benötigt aber etwas mehr Trainingszeit.

Sigmoid konvergiert am langsamsten und zeigt bei tiefen Netzen Schwierigkeiten durch verschwindende Gradienten.

Insgesamt ist ReLU im Playground meist die effizienteste Wahl.

Einfluss von Noise (Noise-Level $= 15$)

Wird der Noise-Level erhöht, beginnen vor allem große Netze, sich stark an einzelne Trainingspunkte anzupassen. Die Trainingskosten werden sehr niedrig, während die Testkosten steigen. Die Entscheidungsgrenze wirkt unruhig und stark verzerrt. In diesem Fall kann eindeutig von Überanpassung gesprochen werden.

Einfachere Modelle zeigen hier oft ein stabileres Verhalten.

Trainings- und Testergebnisse

Nicht alle Datenpunkte können bei verrauschten Datensätzen korrekt klassifiziert werden. Besonders beim Spiral-Datensatz ist eine perfekte Trennung kaum möglich. Mit steigender Modellkomplexität nimmt zudem die Rechenzeit für die Entscheidungsgrenze zu, insbesondere bei Sigmoid-Aktivierungen.

Beobachtung der versteckten Schichten

In der ersten versteckten Schicht reagieren die Neuronen meist auf einfache, lineare Merkmale. In den späteren Schichten werden diese zu komplexeren Mustern kombiniert. Die Ausgaben der letzten Schicht sind daher deutlich abstrakter und stärker spezialisiert als die der ersten Schicht.

Fazit

Logistische Regression eignet sich nur für einfache, lineare Probleme. MLPs können auch komplexe Strukturen wie Kreise oder Spiralen lernen, benötigen dafür aber ausreichend Tiefe und Neuronen. Die Wahl der Aktivierungsfunktion und der Umgang mit Noise sind entscheidend, um Überanpassung zu vermeiden und gute Generalisierung zu erreichen.