Overfitting & MLP

1. NN.MLP.01: Perzeptron-Netze

Gegeben

Zwei Eingabevariablen $x_1$und x_2.
Die Aktivierungsfunktion aller Perzeptrons ist die sign-Funktion:

$$\text{sign}(z) = \begin{cases} +1 & \text{für } z \geq 0 \\ -1 & \text{für } z < 0 \end{cases}$$

Ziel ist es, die in der Abbildung blau-grau dargestellten Bereiche mit $+1$zu klassifizieren.

Perzeptron 1

Detektiert den Bereich oberhalb der Geraden $x_2 = 3$

$$y_1 = \text{sign}(0 \cdot x_1 + 1 \cdot x_2 - 3) = \text{sign}(x_2 - 3)$$

- $y_1 = +1$, falls $x_2 \geq 3$
- $y_1 = -1$, sonst

Perzeptron 2

Detektiert den Bereich rechts der Geraden $x_1 = 2$

$$y_2 = \text{sign}(1 \cdot x_1 + 0 \cdot x_2 - 2) = \text{sign}(x_1 - 2)$$

- $y_2 = +1$, falls $x_1 \geq 2$
- $y_2 = -1$, sonst

Perzeptron 3

Detektiert den Bereich unterhalb der Geraden $x_2 = 4$

$$y_3 = \text{sign}(-1 \cdot x_2 + 4) = \text{sign}(4 - x_2)$$

- $y_3 = +1$, falls $x_2 \leq 4$
- $y_3 = -1$, sonst

Ausgabeneuron

Verknüpft die drei Perzeptrons mit einer UND-Operation

$$y = \text{sign}(1 \cdot y_1 + 1 \cdot y_2 + 1 \cdot y_3 - 2.5)$$

Funktionsweise des Netzes

Das Netz gibt $y = +1$aus, wenn alle folgenden Bedingungen erfüllt sind:

$$\begin{array}{|ll|}\hline x_2 & \geq 3 \\ x_1 & \geq 2 \\ x_2 & \leq 4 \\ \hline\end{array}$$

Damit wird exakt der blau-graue Bereich der Abbildung klassifiziert.

In allen anderen Fällen gilt:

$$y = -1$$

2. NN.MLP.02: Vorwärtslauf im MLP

Gegebene Architektur

- Eingangsschicht: 25 Neuronen
- Erste versteckte Schicht: 64 Neuronen
- Zweite versteckte Schicht: 32 Neuronen
- Ausgabeschicht: 4 Neuronen
- Aktivierungsfunktion: ReLU in allen Schichten

$$\text{ReLU}(z) = \max(0, z)$$

1. Dimensionen der Gewichtsmatrizen und Bias-Vektoren

Schicht	Gewichtsmatrix	Dimension	Bias-Vektor	Dimension
Eingang → 1. versteckte Schicht	$W^{[1]}$	64×25	$b^{[1]}$	64×1
1. → 2. versteckte Schicht	$W^{[2]}$	32×64	$b^{[2]}$	32×1
2. versteckte → Ausgabeschicht	$W^{[3]}$	4×32	$b^{[3]}$	4×1

2. Vorwärtslauf (Forward Pass) in Matrixnotation

Sei der Eingabevektor:

$$x \in \mathbb{R}^{25\times 1}$$

Schicht 1:

$$z^{[1]} = W^{[1]}x + b^{[1]} \in \mathbb{R}^{64\times 1}$$
$$a^{[1]} = \mathrm{ReLU}(z^{[1]})$$

Schicht 2:

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]} \in \mathbb{R}^{32\times 1}$$
$$a^{[2]} = \mathrm{ReLU}(z^{[2]})$$

Schicht 3 (Ausgabe):

$$z^{[3]} = W^{[3]}a^{[2]} + b^{[3]} \in \mathbb{R}^{4\times 1}$$
$$a^{[3]} = \mathrm{ReLU}(z^{[3]})$$

Netzwerkausgabe:

$$a^{[3]} \in \mathbb{R}^{4\times 1}$$

3. NN.MLP.03: Tensorflow Playground

1. Logistische Regression auf verschiedenen Datensätzen

Auf dem Gaussian-Datensatz (linear separierbar) lernt die logistische Regression eine nahezu perfekte lineare Entscheidungsgrenze. Die Trainings- und Testkosten sinken schnell gegen Null, und die Daten können korrekt klassifiziert werden.

Auf nichtlinearen Datensätzen wie Circle, Spiral, XOR oder Moons bleibt die Entscheidungsgrenze stets linear. Eine korrekte Trennung ist nicht möglich, die Kosten bleiben hoch, und es tritt Underfitting auf. Eine Überanpassung entsteht hier nicht, da das Modell zu einfach ist.

2. MLP mit verschiedenen Architekturen und Aktivierungsfunktionen

Mit steigender Anzahl an versteckten Schichten und Neuronen wird die Entscheidungsgrenze komplexer und nichtlinearer.
Für den Circle-Datensatz reichen meist eine einzelne versteckte Schicht mit 3–5 Neuronen aus.
Der Spiral-Datensatz benötigt deutlich tiefere Netze, um die stark nichtlineare Struktur abzubilden.

Die Wahl der Aktivierungsfunktion hat Einfluss auf Form und Geschwindigkeit:

- ReLU konvergiert am schnellsten und erzeugt scharfe Entscheidungsgrenzen.
- tanh liefert glattere Grenzen, trainiert etwas langsamer.
- Sigmoid ist am langsamsten und anfällig für vanishing gradients.

Die Trainingskosten sinken mit wachsender Modellkomplexität stark, die Testkosten jedoch nur bis zu einem gewissen Punkt. Bei sehr komplexen Netzen, insbesondere beim Spiral-Datensatz, kann es zu Overfitting kommen.
Einfache Netze trainieren schneller als tiefe Netze.

Die Ausgaben der ersten versteckten Schicht zeigen einfache Transformationen der Eingabedaten, während die letzte versteckte Schicht abstraktere Merkmale lernt und stärker auf die endgültige Klassifikation spezialisiert ist.

3. Experimente mit Noise-Level 15

Durch das Erhöhen des Noise-Levels auf 15 wird die Trennbarkeit der Daten deutlich schlechter. Die Entscheidungsgrenze wird unregelmäßiger, insbesondere bei tiefen Netzen mit vielen Neuronen, die versuchen, auch das Rauschen zu lernen.

Die Trainingskosten sinken bei komplexen Modellen stark, während die Testkosten deutlich höher bleiben oder wieder ansteigen. In diesem Fall liegt Überanpassung (Overfitting) vor. Einfachere Modelle zeigen einen geringeren Abstand zwischen Trainings- und Testkosten und generalisieren robuster.

Die Berechnung der Entscheidungsgrenze dauert bei tiefen Netzen länger. Modelle mit ReLU konvergieren schneller als solche mit tanh oder Sigmoid.

Eine perfekte Klassifikation aller Datenpunkte ist bei Noise nicht möglich, da sich die Klassen überlappen. Komplexe Netze können Trainingsdaten zwar gut anpassen, versagen jedoch häufiger auf Testdaten.

Die erste versteckte Schicht lernt einfache Merkmale, während die letzte versteckte Schicht abstraktere und stärker spezialisierte Aktivierungen zeigt.