4. Übungsblatt

Aufgabe 19 Trainieren von Schwellenwertelementen

Geben Sie den Ablauf des Lernvorgangs mit Online Training (Delta-Regel) eines Schwellenwertelementes für die Boole'sche Funktion $x_1 \rightarrow x_2$ an! Am besten mit Hilfe einer Tabelle, die Spalten für die Werte von x_1, x_2, $d = x_1 \rightarrow x_2$, $\vec{x} \cdot \vec{w}$, y, e, $\Delta\theta$, Δw_1, Δw_2, θ, w_1 und w_2 enthält. Verwenden Sie als Anfangsbelegung des (erweiterten) Gewichtsvektors $\vec{w} = (0, 0, 0)$ und als Lernrate 1. Geben Sie eine geometrische Interpretation des Lernergebnisses an!

Hinweis : Die Aufgabe kann gerne programmiert und visualisiert werden.

Aufgabe 20 Gradientenverfahren

Suchen Sie das Minimum der Funktion $f(x_1, x_2) = 2x_1^2 - 2x_1 + x_2^2 - x_2$ mithilfe des Gradientenverfahrens. Nutzen Sie die Anfangsnäherung $(x_1, x_2) = (0, 0)$ und die Schrittweite $\gamma = 0.2$.

Hinweis : Die Aufgabe kann gerne programmiert und visualisiert werden.

Aufgabe 21 Funktionsapproximation

a) Geben Sie ein mehrschichtiges Perzeptron mit ca. 10 Neuronen an, das die Funktion $y = x^2$ im Intervall $[0.5, 4.5]$ durch eine Treppenfunktion annähert.

b) Wie kann man diese Näherung verbessern? (Geben Sie zwei Möglichkeiten an.)

Aufgabe 22 Funktionsapproximation

Wir betrachten die Indikatorfunktion der rationalen Zahlen über der Menge der reellen Zahlen (auch als Dirichlet-Funktion bekannt), d.h. die Funktion

$$f : \mathbb{R} \rightarrow \{0, 1\}, \qquad x \mapsto \begin{cases} 1, & \text{falls } x \in \mathbb{Q}, \\ 0, & \text{sonst.} \end{cases}$$

a) Kann diese Funktion durch ein neuronales Netz (mehrschichtiges Perzeptron) beliebig genau angenähert werden?

b) Was zeigt das Ergebnis der Teilaufgabe a) über die Berechnungsfähigkeiten neuronaler Netze?

Aufgabe 23 Bonus: Berechnung von Gradienten

Recherchieren Sie wie Numerische Differentiation zur Berechnung eines Gradienten funktioniert.

Zeigen sie am Beispiel $x_1^2 + x_2$, wie die Verfahren an der Stelle $(1, 1)$ angewendet werden.

Aufgabe 24 Bonus: Automatisches Differenzierungsframework (2)

Dieses mal wollen wir unser angefangenes Automatisches Differenzierungsframework um einige Operationen erweitern. Für alle Funktionen gibt es wie beim letzten mal geeignete Tests um die Korrektheit zu überprüfen.

a) Erweitere die Tensor Klasse um folgende Methoden:

- square(self) : Methode, die die Elemente im Tensor jeweils quadriert
- mean(self): Methode, die den Mittelwert aller Elemente im Tensor bestimmt
- transpose(self) : Methode, die die letzten beiden Dimensionen transponiert.
- reshape(self,shape) : Methode, die den Tensor in eine andere Form bringt.
- ___getitem___(self, key) : Methode die das Verwenden der Tensoren mit dem [] Operator ermöglicht.

b) Im nächsten Schritt wollen wir unsere Methoden so anpassen, dass sie broadcasting unterstützen.

- Implementieren Sie die broadcast(numpyArray a, numpyArray b) Methode. Diese Methode soll die gebroadcasteten Tensoren und Funktionen die den Gradienten auf die ursprüngliche Dimensionen reduzieren kann.
- Passen Sie die Operationen : ___add___(), ___sub___(), ___mul___() an, sodass diese broadcasting bei der Benutzung unterstützen.