Neuronale Netze in der Informatik an der Mittelschule

Ein spielerischer Einstieg ins Thema anhand eines selbst gebauten «Computers» aus Zündholzschachteln

André Nuber Leistungsnachweis Fachdidaktik GymInf 13.10.23

Bei dem vorliegenden Dokument handelt es sich um einen Leistungsnachweis im Rahmen der Fachdidaktik II, GymInf, 2023. Der Auftrag lautete, eine Methode zum Algorithmen-Entwurf vorzustellen. Künstliche neuronale Netzwerke, woraus zahlreiche Algorithmen resultieren, angefangen bei der Bild- oder Spracherkennung bis hin zur medizinischen Diagnostik, sind eine Form von Algorithmen-Entwurf. Entstanden ist daher eine Werkstatt rund um neuronale Netzwerke. Im Zentrum steht ein Computer aus Zündholzschachteln, der wie ein neuronales Netz trainiert wurde. Er ist nicht unschlagbar, gewinnt aber statistisch gesehen 86% der Spiele gegen starke Gegner/-innen.

Motivation

Spätestens seit der Einführung von ChatGPT (OpenAI, 2022) ist der Begriff der künstlichen Intelligenz (KI) in aller Munde. Die Lernenden sollen gemäss Lehrplan «Neuerungen in der Informatik offen begegnen» (EDK, 2017). Dies sollte beim vorliegenden Thema kein Problem sein – es ist ohnehin von sehr grossem Interesse. Die Umsetzung der Ziele der EDK allein sind schon Motivation genug, den Lernenden dieses Thema näherzubringen, geradezu zwingend erscheint diese Lehraufgabe mit Blick auf die Aktualität des Themas.

Im Rahmen eines Projektes im Ergänzungsfach «Anwendungen der Mathematik» hat mich ein Lernender auf «MENACE» aufmerksam gemacht, einen Computer aus Zündholzschachteln, der in der Lage ist, Tic-Tac-Toe zu spielen (Michie, 1963). Das Spezielle dabei ist: Dieser «Computer» verfolgt nicht einfach eine deterministische, perfekte Spielstrategie, sondern er ist wie ein neuronales Netz trainiert worden. Nach 130 Trainingsspielen gewinnt MENACE die meisten Spiele gegen «einfache» Gegner/-innen. Ab dem 220. Spiel erreicht es gegen «perfekte» Gegner/-innen in praktisch jedem Spiel ein Unentschieden (Michie, 1963).

Der besagte Lernende des Ergänzungsfachs hat auf meine Anregung hin einen ähnlichen Computer für eine einfache Version des NIM-Spiels gebaut (Preisig, 2022). Auf dem Tisch liegen acht Zündhölzer. Computer und Mensch haben abwechslungsweise die Möglichkeit, eines oder zwei zu ziehen. In seiner Version verlor derjenige, der gezwungen war, das letzte Hölzchen zu ziehen.

Sein Computer besteht aus je einer Zündholzschachtel für jede Spielkonfiguration. Darin befinden sich rote und weisse Kugeln. Ist der Computer an der Reihe, wird zufällig eine Kugel aus der Schachtel gezogen. Ist sie weiss, zieht der Computer ein Zündholz; ist sie rot, zieht er zwei. Nach dem Spiel folgt das Training: Hat der Computer verloren, verwirft er alle gezogenen Kugeln. Hat er gewonnen, werden sie zurückgelegt mit je einer zusätzlichen derselben Farbe.

Nach dieser Einführung folgen die Werkstatt-Posten, welche direkt zur Durchführung verwendet werden können. Einige davon widmen sich einer leicht komplexeren Variante des NIM-Spiels. Anhand dieses Spiels erwerben die Lernenden Kenntnisse über neuronale Netze und deren Training. Die Posten 1 und 2 sollten zuerst und in dieser Reihenfolge ausgeführt werden. Die Posten 3-6 sind in beliebiger Reihenfolge durchführbar. Posten 7 fasst das Ganze zusammen und ist daher als Abschluss gedacht.

Im Anschluss folgen Hintergrundinformationen für Lehrpersonen: eine Bauanleitung für die Zündholzschachtel-Computer, Musterlösungen zu den Posten und ein Trainingsprotokoll, das

wertvolle Einblicke ins Training des in Posten 3 verwendeten Computers gewährt. Viel Spass beim Spielen, Erforschen, Modifizieren, Implementieren und Unterrichten!

Voraussetzungen

- Die Lernenden können Pseudocode für einfache Algorithmen schreiben.
- Aus der Wahrscheinlichkeitstheorie sind grundsätzliche Kenntnisse bezüglich des Ziehens aus einer Urne bekannt. Die Lernenden können die Wahrscheinlichkeit bestimmen, mit der eine bestimmte Farbe aus einer Urne gezogen wird.
- Die Unterrichtssequenz ist für die 9.-10. Klasse konzipiert.
- Die Durchführung dauert rund 4 Lektionen.
- Die Lernenden kennen die Begriffe «Spiel- / Gewinn- / Verlustkonfiguration» und «Gewinnstrategie» (Hromkovic et al., 2023).

Lernziele

Rahmenlehrplan Informatik

- «Wie in anderen Fachgebieten auch, geht es im Informatikunterricht um die Vermittlung allgemeiner Kenntnisse, die auf andere – auch zukünftige – Anwendungsfälle übertragen werden können. Im Unterschied zu anderen Fachgebieten beziehen sich die Grundprinzipien der Informatik auf menschengemachte Systeme, sie können – anhand stufengerechter Beispiele – im Detail nachvollzogen und aktiv manipuliert oder sogar kreativ angewendet werden.»
- Der Lehrplan legt Wert auf «allgemeine, übertragbare und längerfristig gültige Konzepte und Kompetenzen [...], aktive Erfahrung im Umgang mit diesen Konzepten».
- Die Lernenden sollen «Algorithmen entwerfen, beurteilen und in einer Programmiersprache umsetzen».
- Ausserdem sollen sie «Neuerungen in der Informatik offen begegnen, z. B. neue Fachbereiche der Informatik kennen wollen».

(EDK, 2017)

Erweiterung des Lernzielkatalogs

Die Lernenden ...

- können einen deterministischen Algorithmus entwerfen, der eine Gewinnstrategie für das hier vorliegende NIM-Spiel beschreibt (T3),
- können deterministische und nichtdeterministische Algorithmen unterscheiden (T2),
- kennen die Begriffe «Neuron», «Gewichte», «neuronales Netz» (NN), «Topologie» (T1).
- verstehen, was Lernen für ein NN bedeutet (T2),
- verstehen, dass sich NN nichtdeterministisch verhalten (T3),
- erfahren, wie ein NN trainiert wird (T3),
- analysieren den Effekt verschiedener Trainings (gegen starken vs. zufälligen Gegner) (T4),
- können ein NN für ein einfaches Spiel entwerfen (T5).

T1-5: Taxonomiestufe nach Bloom et al. (1956)

Posten 1: Eine deterministische Gewinnstrategie

Das NIM-Spiel hat seinen Namen wohl vom germanischen Wort «nim» für den Imperativ «Nimm!» (ETH-Bibliothek, o. J.). Das Spiel läuft folgendermassen ab:

Spielregeln

- Ihr spielt zu zweit.
- Anfangs liegen 9 Zündhölzer auf dem Tisch.
- Ihr zieht abwechslungsweise 1, 2 oder 3 Zündhölzer.
- Wer das letzte Zündholz zieht, gewinnt das Spiel.

Eine Spielkonfiguration besteht also aus der Anzahl verbleibender Zündhölzer und der Person, die an der Reihe ist (z.B. 7 verbleibende Hölzer, B ist an der Reihe).

Spielt mindestens zehn Partien gegeneinander. Beginnt das Spiel abwechselnd. Widmet euch danach der folgenden Auswertung.

Auswertung

1. Diskutiert zu zweit: Wer kann das Spiel gewinnen: Person A, die das Spiel beginnt, oder Person B, die nicht beginnt?

2. Für eine der beiden Personen A bzw. B existiert eine Gewinnstrategie, also eine Strategie, mit der sie das Spiel gewinnen kann, egal, wie die andere Person spielt. Formuliere eine deterministische Gewinnstrategie für dieses Spiel in wenigen Sätzen. Eine deterministische Strategie legt in jeder Spielkonfiguration den Zug einer Person eindeutig fest (egal, was die andere Person tut).

3. Schreibe einen Pseudocode für einen Algorithmus, der deinen Spielzug – abhängig von der aktuellen Spielkonfiguration – beschreibt:

Algorithmus 1: NIM – ein deterministischer Spielzug

Posten 2: Aus Zündholzschachteln einen Computer bauen

Vor dir liegen zehn leere Zündholzschachteln. Alle Schachteln zusammen bilden einen «Computer». Neun Schachteln sind für die möglichen Spielkonfigurationen vorgesehen, eine für die Spielanleitung. Die nummerierten Schachteln 1-9 entsprechen je der Anzahl Zündhölzer, die zu einem bestimmten Zeitpunkt des Spiels noch zur Verfügung stehen. In den Schachteln befinden sich Kugeln, die das Verhalten des Computers bestimmen (siehe Tabelle). Befindet sich eine gelbe Kugel in der Schachtel, so zieht der Computer ein Zündholz, bei einer roten zwei, bei einer schwarzen Kugel zieht er drei Hölzer. Wenn der Computer am Zug ist, wird also die der Anzahl Zündhölzer entsprechende Schachtel geöffnet und nachgesehen, wie viele Hölzer der Computer zieht. Die Kugel wird in der Schachtel belassen.

Im Moment befinden sich noch keine Kugeln in den Schachteln. Lege in jede Schachtel je eine Kugel, und zwar so, dass der Computer das Spiel mit Sicherheit gewinnt, wenn er beginnen darf. Die Kugeln legen das Verhalten des Computers fest – der Mensch kann seine Strategie frei wählen. Die Zuordnung der Kugeln kann man als vollständige Beschreibung eines Algorithmus ansehen.

Halte deine Verteilung hier fest.

Schachtel	1	2	3	4	5	6	7	8	9
Kugel	1	2	3						

Was fällt dir auf?

Gibt es Spielkonfigurationen, bei denen der Computer verliert, wenn man optimal spielt? Welche sind es?

Wie kann der Computer diese Verlustkonfigurationen vermeiden?

Gibt es Spielkonfigurationen, die der Computer gar nie erreicht, wenn er das Spiel beginnt? Wenn ja, welche sind es?

Kannst du diesen Computer schlagen, wenn er das Spiel beginnt?

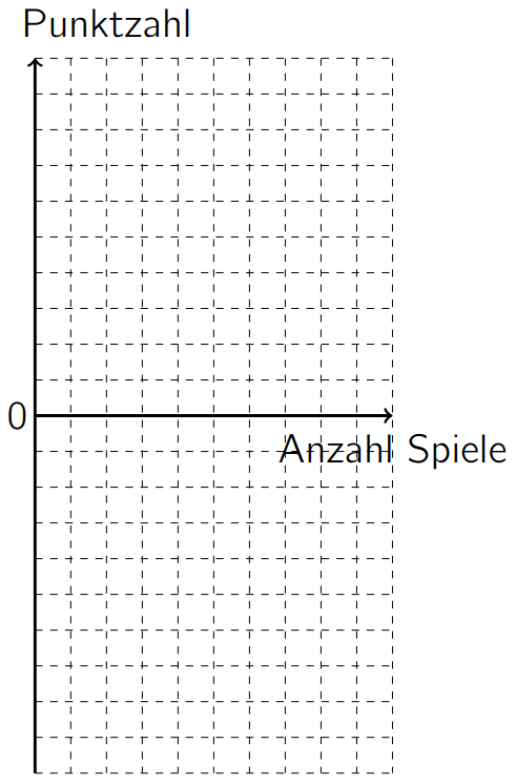
Warum wird dieses Spiel wohl ziemlich schnell langweilig?

Solange du in jede Schachtel nur eine Kugel legst, verhält sich der Computer deterministisch: In einer bestimmten Spielkonfiguration (z. B. wenn noch 5 Hölzer liegen), verhält er sich immer gleich. Mit der Farbe der Kugel hast du den Computer «programmiert».

Posten 3: Ein starker Computer

Spiele 10 Partien gegen den hier vorliegenden Zündholzschachtel-Computer. Halte die Summe deiner erzielten Punkte im Diagramm fest (Sieg: +1 / Niederlage: -1). Die Spielanleitung findest du in der dafür vorgesehenen Zündholzschachtel.

Der Hauptunterschied zu Posten 2 ist folgender: In jeder Schachtel befinden sich nun mehrere Kugeln. Genauer findest du den Spielverlauf in der Anleitung.



NIM-Spiel: Anleitung

Der Computer und du nehmen abwechslungsweise 1, 2 oder 3 Zündhölzer. Gewonnen hat, wer das letzte Zündholz bekommt. Der Computer beginnt jede Partie.

Die Zahlen auf den Schachteln entsprechen der Anzahl verbleibender Zündhölzer. Wenn der Computer am Zug ist, gehe wie folgt vor:

1. Öffne die Schachtel mit der Nummer, die der verbleibenden Anzahl Zündhölzer entspricht.
2. Ziehe mit geschlossenen Augen eine Kugel aus der Schachtel.
3. Der Farbcode besagt, wie viele Zündhölzer er zieht.
4. Lege die Kugel sofort wieder zurück.

Wie gut spielt der Computer? Ist er unschlagbar?

Hat sich der Computer deterministisch verhalten, sprich, hat er sich in der gleichen Spielsituation (z.B. 5 verbleibende Hölzchen) immer gleich verhalten? Wähle eine geeignete Zündholzschachtel als Beispiel.

Beschreibe den Spielzug des Computers als Algorithmus (Pseudocode):

Algorithmus 2: NIM – ein nichtdeterministischer Spielzug

Posten 4: Ein neuronales Netz aus Zündholzschachteln

Spiele einige Partien gegen den Zündholzschachtel-Computer. Die Spielanleitung findest du in der dafür vorgesehenen Zündholzschachtel. Wichtig: Studiere sie genau! Lies insbesondere nach, was nach dem Ende jeder Partie geschehen soll!

1	2	3
4	5	6
7	8	9

NIM-Spiel: Anleitung

Der Computer und du nehmen abwechslungsweise 1, 2 oder 3 Zündhölzer. Gewonnen hat, wer das letzte Zündholz bekommt. Der Computer beginnt jede Partie.

Die Zahlen auf den Schachteln entsprechen der Anzahl verbleibender Zündhölzer. Wenn der Computer am Zug ist, gehe wie folgt vor:

1. Öffne die entsprechende Schachtel.
2. Ziehe mit geschlossenen Augen ein Tic Tac aus der Schachtel und lege es ins Feld (auf dem Blatt) mit derselben Nummer. Befindet sich kein Tic Tac mehr in der Schachtel, gibt der Computer die Partie auf.
3. Der Farbcode besagt, wie viele Zündhölzer er zieht.

Ende einer Partie:
Wenn du eine Partie gewonnen hast, darfst du die Tic Tacs essen. Wenn der Computer gewonnen hat, legst du sie in die Schachtel zurück, aus der du sie entnommen hast, zusammen mit je einem zweiten Tic Tac derselben Farbe. Ersatz Tic Tacs findest du in Schachtel 8.

Könnte es sein, dass dir der Tic Tac-Verzehr auf Dauer schadet bzw. dem Computer hilft? Nach jeder Partie werden die Tic Tacs umverteilt. Wie verändert dies die Spielweise des Computers?

Dieser Computer bildet ein künstliches neuronales Netz (KNN): Die Zündholzschachteln sind die Neuronen, die Tic Tacs sind die Verbindungen zwischen den Neuronen, da sie – zusammen mit der Interaktion des Menschen, der gegen den Computer spielt – von einer Spielkonfiguration zur nächsten führen. Beim Trainieren eines KNN werden die Verbindungen gestärkt bzw. geschwächt.

Sei `zuege` eine Liste mit neun Elementen, welche die Spielzüge des Computers beinhaltet. Beschreibe den Trainings-Algorithmus, sprich, was nach jeder Partie geschieht, mit Pseudocode:

Algorithmus 3: NIM – Training eines neuronalen Netzes

zuege = [0,0,3,0,1,0,0,0,1]; # Computer zog 3 Hölzer als noch 3 lagen, 1 als noch 5 lagen etc.

Stelle die Ausgangssituation des Computers wieder her:

Posten 5: Tic-Tac-Toe

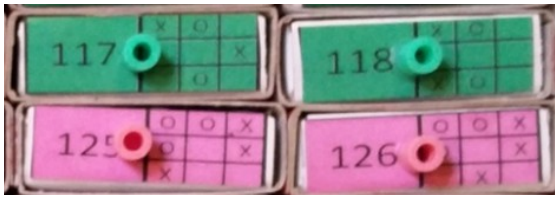


Abb. 1: MENACE – 4 der 304 Schachteln

Der Zündholzschachteln-Computer MENACE spielt Tic-Tac-Toe (siehe Abb. 1, 2). Er wurde auf die gleiche Weise trainiert wie unser Computer für das NIM-Spiel. Da Tic-Tac-Toe aber viel mehr mögliche Spielkonfigurationen enthält, sind (trotz berücksichtigter Symmetrien des Spielfelds) 304 Zündholzschachteln nötig! Unter diesem Link kannst du gegen eine Simulation von MENACE spielen. Im Abschnitt rechts siehst du stets, wie die Kugeln verteilt sind.

1. Wie sind die Kugeln anfangs verteilt?

   Kannst du erklären, warum dies eine sinnvolle Anfangsverteilung ist?

2. Spiele zehn Partien Tic-Tac-Toe unter diesem Link.

   Anzahl deiner Siege:

3. Trainiere den Computer: Lass ihn 30x gegen «Random» spielen. Im Dropdown-Menü kannst du diesen auswählen. Wie wurden nun die Kugeln umverteilt?

   Spiele wieder zehn Partien gegen den Computer.

   Anzahl deiner Siege:

4. Lade die Seite neu, sodass der Computer wieder in der Ausgangslage ist. Lass ihn nun 40x gegen «Perfect» spielen.

   Spiele selbst nochmals zehn Partien gegen den Computer.

   Anzahl deiner Siege:

Was kannst du abschliessend über die Stärke des Computers in den verschiedenen Szenarien sagen? Wie wirkt sich die Qualität des Testdatensatzes aus auf die Stärke des Computers?

Unter diesem Link findest du eine schöne (englischsprachige) Erklärung zu MENACE.


Abb. 2: MENACE – ein Computer aus Zündholzschachteln (Wylie, 2018)

Posten 6: Neuronale Netze

Wer weiss mehr über neuronale Netze als ein neuronales Netz?

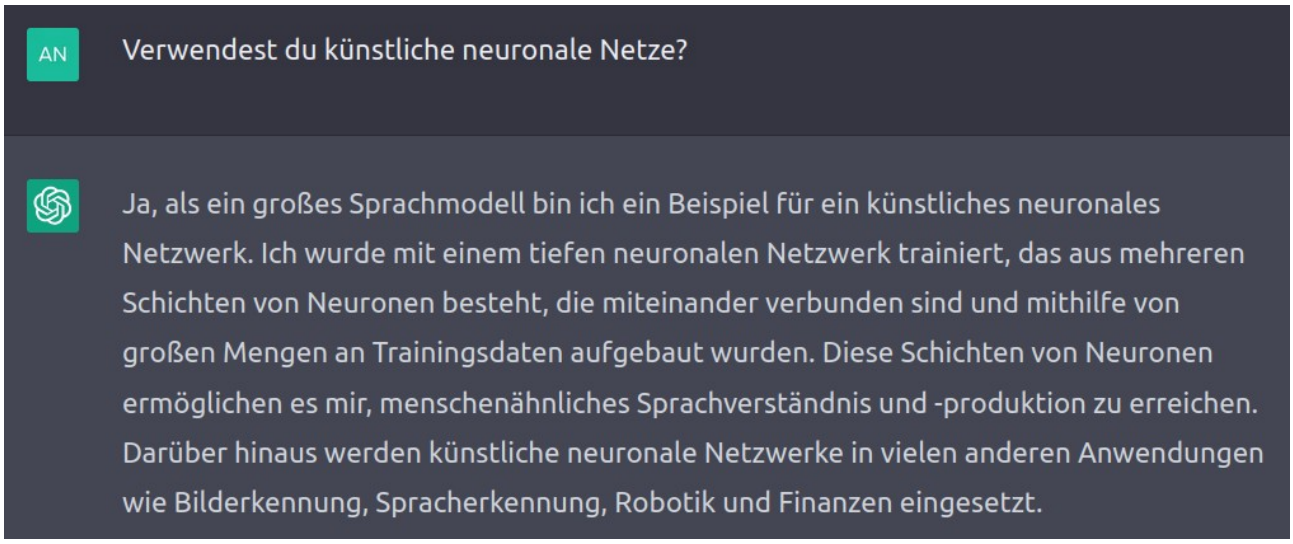


Abb. 3: ChatGPT Antwort zur Verwendung neuronaler Netze (ChatGPT, 2023)

Verwende ChatGPT oder eine Suchmaschine, um deine offenen Fragen zu künstlichen neuronalen Netzen zu klären. Halte Erklärungen zu den folgenden Begriffen fest. Beachte: So, wie unser Zündholzschachtel-Computer nicht perfekt spielt, sind auch die ChatGPT Ergebnisse nicht perfekt. Überprüfe sie stets mit unabhängigen Quellen und gib diese hier ebenfalls an.

Neuron:

Gewichte:

Neuronales Netz:

Topologie:

Anwendungen:

Posten 7: Ein künstliches neuronales Netz für das NIM-Spiel

Mit den Algorithmen 2-3 verfügst du über Bausteine, die du verwenden kannst, um ein neuronales Netz für das NIM-Spiel zu programmieren. Schreibe nun den Pseudocode für das ganze Spiel. Du darfst folgende Funktionen als gegeben betrachten:

```
computer_zieht_nicht_deterministisch(n, verteilung)
/* siehe Algorithmus 2
 * Parameter:
 *   n: Anzahl verbleibender Zündhölzer
 *   verteilung: Verteilung der Kugeln in den Schachteln
 * Returns: z: Anzahl zu ziehender Hölzer (1, 2 oder 3)
 */

training(verteilung_vorher, zuege, erfolg)
/* siehe Algorithmus 3
 * Parameter:
 *   verteilung_vorher: Verteilung der Kugeln vor der Partie
 *   zuege: Liste der Spielzüge, die der Computer ausgeführt hat
 *   erfolg: Erfolg (1) oder Misserfolg (0) des Computers
 * Returns:
 *   verteilung_nachher: Verteilung der Kugeln nach dem Training
 */

mensch_zieht(n)
/* Der Mensch wird aufgefordert, einen Spielzug zu machen.
 * Diese Funktion terminiert, wenn der Spielzug ausgeführt wurde.
 * Parameter: n: Anzahl verbleibender Zündhölzer vor dem Spielzug
 * Returns: z: Anzahl Zündhölzer, die der Mensch zieht
 */
```

Algorithmus 4: Künstliches neuronales Netz für das NIM-Spiel

Hintergrundinformationen für Lehrpersonen

Bauanleitung Zündholzschachtel-Computer

1. Kaufe 10 Zündholzschachteln pro benötigtem Computer (Posten 2-4 benötigen je einen). Zündhölzer gibt es typischerweise in 10er-Packungen in jedem Supermarkt.
2. Leere die Schachteln. Bewahre die Zündhölzer auf. Sie werden in den Posten 1-4 benötigt.
3. Füge die Schachteln mit Klebeband wie abgebildet zu einem Computer zusammen.
4. Lege die Anleitung (siehe Postenblatt) und 9 Zündhölzer in die dafür vorgesehene Schachtel.
5. Fülle die Kugeln wie in den Posten 3-4 beschrieben in die nummerierten Schachteln. Der Computer für Posten 2 bleibt leer.

Abb. 4: NIM-Computer aus Zündholzschachteln

Musterlösungen

Posten 1: Eine deterministische Gewinnstrategie

1. Diskutiert zu zweit: Wer kann das Spiel gewinnen: Person A, die das Spiel beginnt, oder Person B, die nicht beginnt, oder beide?

   Wenn A nach der Gewinnstrategie spielt, gewinnt A, egal welche Züge B macht.

2. Formuliere eine deterministische Gewinnstrategie für dieses Spiel in wenigen Sätzen [...].

   Die Gewinnkonfigurationen sind (B, 4i): Wenn B am Zug ist, soll ein Vielfaches von 4 Hölzchen auf dem Tisch liegen. Die Gewinnstrategie für Person A ist also, die Anzahl verbleibender Hölzchen stets auf ein Vielfaches von 4 zu reduzieren. Damit wird sichergestellt, dass Person A die Spielkonfiguration «4 verbleibende Hölzchen» erreicht. Zieht Person B dann 1-3 Hölzchen, so kann A mit Sicherheit das letzte ziehen.

3. *Schreibe einen Pseudocode für einen Algorithmus, der deinen Spielzug – abhängig von der aktuellen Spielkonfiguration – beschreibt:*

Algorithmus 1: NIM – deterministisch

Option A: kurz und elegant

```
ziehe n mod 4 Zündhölzer
    # n: Anzahl verbleibender Hölzer
```

Option B: erwartete Antwort der Lernenden

```
if n == 1: ziehe 1 Zündholz
else if n == 2: ziehe 2 Zündhölzer
else if n == 3: ziehe 3 Zündhölzer
else if n == 4: gib auf
else if n == 5: ziehe 1 Zündholz
...
```

Das Unschöne am hier vorgestellten Algorithmus 1 ist, dass er nur für genau diese Art des NIM-Spiels funktioniert. Er müsste stark angepasst werden, wenn die Spielregeln verändert werden, und er ist kaum auf andere Spiele übertragbar. Hier bietet es sich an, weitere allgemeinere Algorithmen zu thematisieren (z.B. Minimax, vgl. Wikipedia, 2023b), die auch auf andere Spiele wie z.B. Tic-Tac-Toe übertragbar sind.

Ziel der hier vorliegenden Unterlagen ist aber ein anderes: Die grundlegenden Ideen künstlicher neuronaler Netze sollen eingeführt werden. Damit verfügen die Lernenden über ein allgemeines Konzept des Algorithmen-Entwurfs, das problemlos auf viele weitere Problemstellungen und Spiele übertragbar ist. Demonstriert wird dies exemplarisch am hier vorliegenden NIM-Spiel.

Posten 2: Aus Zündholzschachteln einen Computer bauen

Halte deine Verteilung hier fest:

Schachtel	1	2	3	4	5	6	7	8	9
Kugel	1	2	3	(1/2/3)	1	2	3	(1/2/3)	1

Aus den Spielkonfigurationen 4 und 8 gibt es keine Gewinnstrategie. Dort kann das Spiel aufgegeben werden oder 1 bis 3 Hölzer gezogen werden – der Computer verliert das Spiel, wenn der Mensch optimal spielt. In einem deterministischen Algorithmus muss jedenfalls auch in diesen Konfigurationen eindeutig geklärt werden, was der Computer tun soll.

Was fällt dir auf?

Das Muster scheint sich zu wiederholen: 1, 2, 3, Verlust, 1, 2, 3, Verlust, ...

Gibt es Spielkonfigurationen, bei denen der Computer verlieren wird? Welche sind es?

Wenn der Computer am Zug ist und ein Vielfaches von 4 Hölzchen verbleibt, verliert er das Spiel.

Wie kann der Computer diese Verlustkonfigurationen vermeiden?

Die Verlustkonfigurationen können vermieden werden, indem der Computer (der jede Partie beginnt) stets so viele Hölzchen zieht, dass der Mensch in in diesen Konfigurationen landet.

Gibt es Spielkonfigurationen, die der Computer nie erreicht, wenn er das Spiel beginnt? Wenn ja, welche sind es?

Die Konfiguration (Computer, 8) wird bei 9 Starthölzchen nie erreicht: Ziehen der Computer und der Mensch je mindestens ein Hölzchen, verbleiben noch höchstens 7 Hölzchen.

Kannst du diesen Computer schlagen, wenn er das Spiel beginnt?

Nein, dieser deterministische Computer ist unschlagbar, solange er jede Partie beginnt.

Warum wird dieses Spiel wohl ziemlich schnell langweilig?

Gegen einen unschlagbaren Computer, der sich immer gleich verhält und jedes Spiel gewinnt, wird jegliches Spiel wohl ziemlich schnell langweilig und frustrierend.

Posten 3: Ein starker Computer

Wie gut spielt der Computer? Ist er unschlagbar?

Dieser Computer ist nicht unschlagbar. Er gewinnt aber gegen eine starke Gegnerin dennoch 86% der Partien. Eine starke Gegnerin kennt die perfekte Spielstrategie. Sie weiss insbesondere, dass sie das Spiel gewinnt, wenn sie so zieht, dass noch 4 Zündhölzer verbleiben. Eine ausführlichere Analyse findet man im Trainingsprotokoll.

Hat sich der Computer deterministisch verhalten, sprich, hat er sich in der gleichen Spielsituation (z.B. fünf verbleibende Hölzchen) immer gleich verhalten? Wähle eine geeignete Zündholzschachtel als Beispiel.

Ein Blick in Schachtel 7 zeigt: Manchmal zieht der Computer in dieser Konfiguration zwei, manchmal drei Hölzer. Er verhält sich also nicht immer gleich und somit nichtdeterministisch.

Beschreibe den Spielzug des Computers als Algorithmus (Pseudocode):

Algorithmus 2: NIM – ein nichtdeterministischer Spielzug

öffne Schachtel n;
ziehe eine Kugel;
Anzahl zu ziehender Hölzer z = Wert der Kugel;

Etwas konkreter:

```
werte = [1,2,3];                    # 1, 2 oder 3 Hölzer können gezogen werden
gewichte = verteilung[n];           # gewichte: Verteilung der Kugeln in Schachtel n, z.B. [3, 6, 1]
z = waehle_zufaellig(werte, gewichte);      # aus werte wird zufällig einer ausgewählt
return z;
```

In Python kann die Funktion `random.choices(werte, weights=[3,6,1], k=1)[0]` *verwendet werden, um «waehle_zufaellig(...)» zu implementieren.*

Posten 4: Ein neuronales Netz aus Zündholzschachteln

Könnte es sein, dass dir der Tic Tac-Verzehr auf Dauer schadet bzw. dem Computer hilft? [...]

Wenn du ein Tic Tac isst, verringerst du die Wahrscheinlichkeit, dass der Computer diesen Zug erneut ausführt. Ist es das letzte einer Farbe, dann wird der Computer diesen Zug nie mehr ausführen. Da du die Tic Tacs nur essen darfst, wenn du gewonnen hast, wird der Computer also Züge, die zu seinem Verlust geführt haben, mit geringerer Wahrscheinlichkeit (oder gar nicht mehr) ausführen.

Analog funktioniert die Belohnung: Wenn der Computer gewonnen hat, verdoppelst du die Tic Tacs der entsprechenden Farbe. Daher wird er die Gewinnzüge mit erhöhter Wahrscheinlichkeit ausführen.

Je länger du den Computer belohnst oder bestrafst, desto besser wird er. Der Computer aus Posten 3 wurde so trainiert. Er gewinnt statistisch gesehen rund 86% der Partien, auch wenn du noch so gut spielst. Diese Zahl hängt aber stark vom Training ab. Details zum Training des

Computers in Posten 3 sind im «Trainingsprotokoll» zu finden. Wäre der Computer mit einem schwächeren Gegner trainiert worden, wäre auch der Computer nach 30 Partien bedeutend schwächer. Dies lässt sich in Posten 5 mit Simulationen für ein anderes Spiel, Tic-Tac-Toe, erfahren.

Algorithmus 3: NIM – Training eines neuronalen Netzes

```
zuege = [0,0,3,0,1,0,0,0,1];   # Computer zog 3 Hölzer als noch 3 lagen, 1 als noch 5 lagen etc.
for i = 0 bis laenge(zuege)-1:
    if erfolg:                 # wenn der Computer gewonnen hat
        erhöhe verteilung[i][zuege-1] um 1
    else:
        if verteilung[i][zuege-1] > 0:
            vermindere verteilung[i][zuege-1] um 1
```

Posten 5: Tic-Tac-Toe

1. Wie sind die Kugeln anfangs verteilt?

Alle Züge sind gleich wahrscheinlich. Die Symmetrien wurden berücksichtigt, z.B. enthält die «box for the first move» nur Kugeln in je einem Eck-, einem Randfeld und in der Mitte (Scroggs, 2016).

Je weiter das Spiel fortgeschritten ist, desto weniger Kugeln pro Spielzug enthält die Schachtel in der Anfangsverteilung.

Kannst du erklären, warum dies eine sinnvolle Anfangsverteilung ist?

Je weiter das Spiel fortgeschritten ist, desto stärker sollen Fehler bestraft werden. Macht der Computer z.B. einen 7. Zug, der zum Verlust geführt hat, soll dieser nie wiederholt werden. Die eine Kugel wird sofort verworfen. Dies heisst aber noch lange nicht, dass der Computer den Startzug komplett verwerfen soll, da damit allenfalls auch ein Sieg möglich gewesen wäre.

Was kannst du abschliessend über die Stärke des Computers in den drei verschiedenen Szenarien sagen? Wie wirkt sich die Qualität des Testdatensatzes aus auf die Stärke des Computers?

Je stärker der Gegner, mit dem der Computer trainiert, desto schneller sind die Fortschritte des Computers. Zwar sind Trainingseffekte auch gegen schwache oder zufällig spielende Gegner erkennbar, allerdings sind viel mehr Trainingspartien nötig. Die Stärke des Trainingsgegners entspricht der Qualität des Testdatensatzes. Diese ist also für ein effizientes Training entscheidend.

Posten 6: Neuronale Netze

Neuron: *Neuronen berechnen im Allgemeinen aus Eingangsgewichten mithilfe einer Aktivierungsfunktion ein Ausgangssignal. In unserem Beispiel entsprechen die Zündholzschachteln den Neuronen, die ausgeben, wie viele Zündhölzer gezogen werden sollen.*

Gewichte: *Die Gewichte entsprechen der Stärke der Verbindung zwischen zwei Neuronen. In unserem Beispiel bestimmt die Farbe der Kugel (zusammen mit dem Spielzug des Menschen) die Verbindung zwischen zwei Spielkonfigurationen. Die relative Anzahl der Kugeln entspricht daher dem Gewicht einer Verbindung.*

Neuronales Netz: *Ein neuronales Netz besteht aus Neuronen und den Verbindungen dazwischen.*

Topologie: *Mit der Topologie ist die Struktur eines neuronalen Netzes gemeint.*

Anwendungen: *Sprachmodelle (Large Language Models), Mustererkennung in Bildern uvm.*

Posten 7: Ein künstliches neuronales Netz für das NIM-Spiel

Der folgende Algorithmus beschreibt eine Partie inkl. Training. Für die vollständige Implementierung muss zusätzlich die Anfangsverteilung der Kugeln definiert werden. Ausserdem könnte man die Partie in ein `while True` *packen, damit man beliebig viele Partien spielen kann.*

Algorithmus 4: Künstliches neuronales Netz für das NIM-Spiel

```
n = 9;                          # Anzahl verbleibender Zündhölzer
zuege = [0,0,0,0,0,0,0,0,0];    # speichert die Züge des Computers
am_zug = 1;                     # 0: Mensch am Zug, 1: Computer am Zug

while n>0:
    if am_zug:
        z = computer_zieht_nicht_deterministisch(n, verteilung)
        zuege[n-1] = z;
        am_zug = 0;
    else:
        z = mensch_zieht(n);
        am_zug = 1;
    vermindere n um z;

# Spielende
if am_zug:    # Wäre nun der Computer am Zug, hat der Mensch das letzte Zündholz gezogen.
    erfolg = 0
    Ausgabe: «Der Mensch hat gewonnen.»
else:
    erfolg = 1
    Ausgabe: «Der Computer hat gewonnen.»
verteilung = training(verteilung, zuege, erfolg)
```

Trainingsprotokoll

Für die Lehrperson, die diese Unterlagen verwendet, ist es ratsam, selbst einen Computer zu trainieren, indem sie viele Partien gegen ihn spielt. Dabei soll sie sich den folgenden Fragen stellen:

- Wie sieht eine sinnvolle Anfangsverteilung aus?
- Welche Strategie verfolgt der Mensch, der gegen den Computer spielt?
- Sind zusätzliche Regeln nötig bzw. sinnvoll?

Den Lernenden würde ich diese Arbeit weitgehend ersparen – sie ist mühsam und zeitaufwendig. Es folgen die wesentlichen Erkenntnisse aus dem Training des Computers in Posten 3:

- Die Anfangsverteilung (siehe Abbildung) entspricht derjenigen des Computers in Posten 4:
  - Wenn der Computer das letzte Zündholz ziehen kann, macht er dies. Die Schachteln 1-3 enthalten daher nur eine einzige Kugel einer Farbe. Diese Konfiguration hat sich in früheren Trainings ohnehin nach nur gerade 7 Partien eingestellt.
  - Die Schachteln 4-6 enthalten je von jeder Farbe eine Kugel, Schachtel 7 je 2, Schachtel 9 je 3. Verlustzüge sollen umso härter bestraft werden, je näher sie am Spielende sind (also je kleiner die Nummer der Schachtel ist).
  - Schachtel 8 enthält keine Kugeln: Diese Konfiguration kann nicht erreicht werden, wenn der Computer das Spiel beginnt.

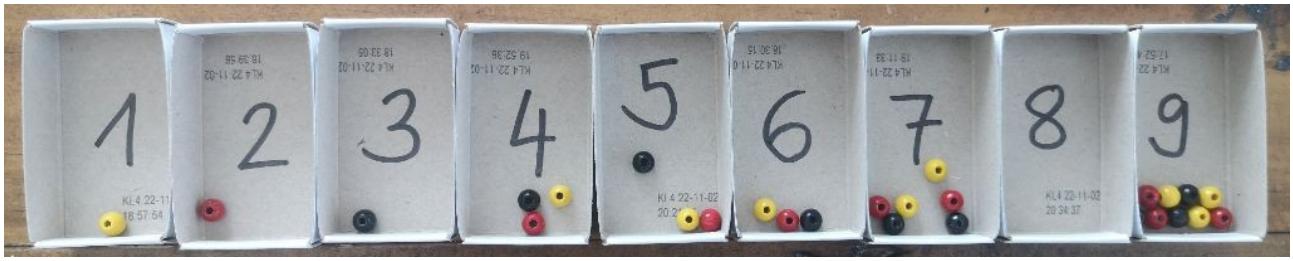


Abb. 5: Anfangsverteilung der Kugeln vor Trainingsbeginn des Computers.

- Gespielt wurden 30 Partien gegen einen starken Gegner, der die optimale Spielstrategie anwendet. Dieser zieht, wenn möglich, *n* mod 4 Kugeln. Entspricht die Anzahl Kugeln einem Vielfachen von 4, so werden zufällig 1 bis 3 Kugeln gezogen.
- Ähnliche Trainings wurden gegen einen Zufallsgegner durchgeführt. Die Trainingseffekte sind dann zwar auch erkennbar, es sind aber viel mehr Trainingspartien nötig. Hier wird nur das Training gegen den «starken Gegner» weiter thematisiert.

Analyse:

- Nach 30 Partien hat sich die abgebildete Verteilung eingestellt (Abb. 6).
- Bereits nach 4 Partien hat der Computer keine Kugeln mehr in Schachtel 4. Er gibt diese Position auf.
- Nach 11 Partien befinden sich nur noch gelbe Kugeln in Schachtel 9. Diese führen zu 8 verbleibenden Hölzern, dies ist eine Gewinnstellung.
- Nach 17 Partien befinden sich nur noch rote Kugeln in Schachtel 6. Diese führen zu 4 verbleibenden Hölzern, dies ist wiederum eine Gewinnstellung.
- Das folgende Baumdiagramm (Abb. 7) wurde verwendet, um die Wahrscheinlichkeit zu berechnen, dass der Computer das Spiel gewinnt. Sie beträgt:
  $$P(\text{Computer gewinnt}) = 1 \cdot \frac{1}{3} \cdot \frac{12}{13} + 1 \cdot \frac{1}{3} \cdot 1 + 1 \cdot \frac{1}{3} \cdot \frac{4}{6} \approx 0.863 = 86.3\%$$

Abb. 6: Verteilung der Kugeln nach 30 Partien – dieser Computer wird in Posten 3 verwendet.

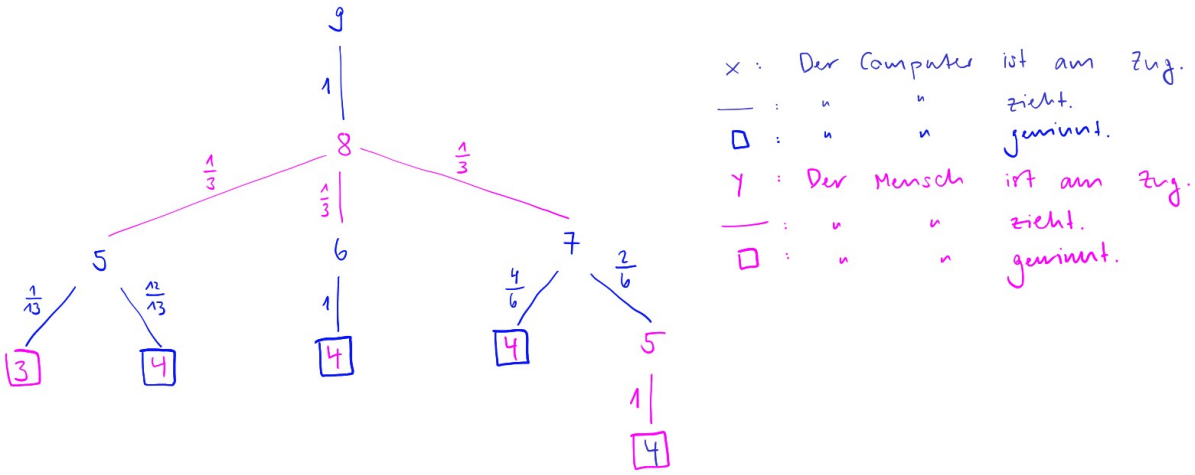


Abb. 7: Baumdiagramm zur Ermittlung der Gewinnchance des Computers.

Literaturverzeichnis

Bloom, B. S., Engelhart, M. D., Furst, E. J., Hill, W. H. & Kratwohl, D. R. (1956). *Taxonomy of educational objectives: The classification of educational goals.* Vol. Handbook I: Cognitive domain. New York: David McKay Company.

ChatGPT (2023): *Verwendest du künstliche neuronale Netze?* Abgerufen am 3.3.23 unter https://chat.openai.com/chat.

EDK (2017): Rahmenlehrplan für die Maturitätsschulen: Informatik. Abgerufen am 3.3.2023 unter https://www.edk.ch/de/themen/gymnasium.

ETH-Bibliothek (o. J.): Nim. Abgerufen am 24.4.23 unter https://library.ethz.ch/standorte-und-medien/plattformen/virtuelle-ausstellungen/alles-ist-spiel/nim.html.

Hromkovič, J. et al. (2023): *INFORMATIK – Algorithmen und KI.* In Vorbereitung.

Michie, D. (1963): *Experiments on the mechanization of game-learning. Part I. Characterization of the model and its parameters*. The Computer Journal, Vol. 6, Issue 3, p. 232-236. Abgerufen am 03.03.2023 unter https://academic.oup.com/comjnl/article/6/3/232/360077?login=false.

OpenAI (2023): *Introducing ChatGPT*. Abgerufen am 3.3.23 unter https://openai.com/blog/chatgpt.

Preisig, R. (2022): *ZÜSLE – Die Zündholzspiel lernende Einheit.* Projekt im Rahmen des Ergänzungsfachs «Anwendungen der Mathematik». ISME St. Gallen.

Scroggs, M. (2016): *MENACE*. Abgerufen am 3.3.23 unter https://www.mscroggs.co.uk/menace/.

Wikipedia (2023a): *Künstliches neuronales Netz*. Abgerufen am 3.3.23 unter https://de.wikipedia.org/wiki/K%C3%BCnstliches_neuronales_Netz.

Wikipedia (2023b): *Minimax-Algorithmus*. Abgerufen am 26.4.23 unter https://de.wikipedia.org/wiki/Minimax-Algorithmus.

Wylie, C. (2018): *How 300 Matchboxes Learned to Play Tic-Tac-Toe Using MENACE.* Abgerufen am 03.03.2023 unter https://opendatascience.com/menace-donald-michie-tic-tac-toe-machine-learning/.