Übungsblatt 04: Decision Tree Learner

Bonus: Starke vs. Schwache KI

1. Ist ein System wie ChatGPT “intelligent”? Was ist der Kern des Systems?

- ChatGPT wirkt intelligent, weil es in natürlicher Sprache kommuniziert, und abhängig vom Kontext reagiert und Wissen aus vielen Bereichen anwendet, aber es versteht nicht was es sagt. Es simuliert Intelligenz auf Basis von Wahrscheinlichkeiten - es wählt Wörter, die mit hoher Wahrscheinlichkeit als nächstes passen, basierend auf Mustern in den Trainingsdaten.
- Der Kern des Systems beruht auf Mustererkennung und Statistik

2. Gibt es Systeme, die intelligent sind? Was tun diese, wie arbeiten sie?

- Man unterscheidet generell zwischen schwacher KI, also KI die nur spezialisierte Aufgaben lösen kann, und starker KI, also KI die Menschen denken und verstehen können. Wenn Intelligenz als menschenähnliches Denken definiert sind, gibt es solche AGIs noch nicht.

3. Brauchen wir wirklich Intelligenz in Systemen? Reicht auch schwache KI, d.h. reichen intelligent *wirkende* Systeme?

- Während schwache Intelligenz für viele Aufgaben ausreichen kann (z.B. Spracherkennung oder medizinische Diagnosen) können ohne starke KI Menschen in einigen Rollen und Jobs nicht ersetzt werden, da diese eine generallisierte Wahrnehmung und Entscheidungsfindung der Umwelt benötigen. Allerdings birgt dies auch probleme, da intelligente Systeme vielleicht nicht nur die Aufgaben erfüllen wollen, die ihnen geboten werden. Sie könnten lügen und eigene, unabhängige Ziele verfolgen.

4. Absicht vs. Auswirkung: vorteilhafte Anwendungen vs. Unfälle (Robustheit und falsche Korrelationen, Fairness, Sicherheit) vs. Missbrauch (Spam, Betrug, Spear-Phishing, Desinformation) vs. doppelte Verwendung (“*dual use*”: Raketen, Kernkraft, Genbearbeitung, ...) vs. *Bias* (Voreingenommenheit)

- KI kann beispielsweise durch Erkennung von Mustern auf MRT Scans schneller, effizienter und genauer Krankheiten frühzeitig feststellen als ein Arzt es könnte, jedoch kann durch die undurchschaubare und statistische Natur der KI auch eine falsche Diagnose gestellt werden, weswegen es in diesem Beispiel ein großes Potenzial bietet, jedoch durch mehrere Instanzen kontrolliert werden sollte, um das Risiko von fehldiagnosen zu mindern. Auch Fehler in den Testdaten (wie z.B. Testdaten die hauptsächlich aus Menschen mit heller Haut bestehen) können zu Mängeln in der Auswertung führen, weshalb diese ebenfalls weiterführend überprüft werden sollten (in diesem Beispiel könnten so Hautkrankheiten bei Menschen mit dunkler Haut fehldiagnostiziert werden). Generell kann man sagen, wenn auch anfällig für Fehler überwiegt das Potenzial der Technologie ihre Risiken bei weitem. Unabhängig davon allerdings gilt dass die Technologie möglich ist und so lange sie möglich ist wird sie auch erforscht und entwickelt werden. Man kann diese Weiterentwicklung nur verlangsamen, nicht aber aufhalten und sollte sie deshalb begrüßen und in die richtigen Bahnen lenken, um positive Eigenschaften zu begünstigen und Fehler sowie Missbrauch und Biases in den Trainingsdaten und der letzendlichen KI zu minimieren.

A01: Entscheidungsbäume mit CAL3 und ID3

- ID3

```
import math
from collections import Counter, defaultdict

class ID3DecisionTree:
    def __init__(self):
```

```
        self.tree = None

    def entropy(self, data):
        labels = [row[-1] for row in data]
        counts = Counter(labels)
        total = len(data)
        return -sum((count/total) * math.log2(count/total) for count in counts.
values())

    def info_gain(self, data, attr_index):
        total_entropy = self.entropy(data)
        values = defaultdict(list)
        for row in data:
            values[row[attr_index]].append(row)
        subset_entropy = sum((len(v)/len(data)) * self.entropy(v) for v in valu
es.values())
        return total_entropy - subset_entropy

    def best_attribute(self, data):
        attrs = range(len(data[0]) - 1)
        gains = {a: self.info_gain(data, a) for a in attrs}
        return max(gains, key=gains.get)

    def fit(self, data, attributes):
        labels = [row[-1] for row in data]
        if len(set(labels)) == 1:
            return labels[0]
        if len(attributes) == 0:
            return Counter(labels).most_common(1)[0][0]

        best = self.best_attribute(data)
        tree = {attributes[best]: {}}
        for value in set(row[best] for row in data):
            subset = [row for row in data if row[best] == value]
            if not subset:
                tree[attributes[best]][value] = Counter(labels).most_common(1)
```

```
[0][0]
            else:
                new_attrs = attributes[:best] + attributes[best+1:]
                new_data = [row[:best] + row[best+1:] for row in subset]
                tree[attributes[best]][value] = self.fit(new_data, new_attrs)
        self.tree = tree
        return tree
```

```
from ID3DecisionTree import ID3DecisionTree
data = [
    ['>=35', 'hoch', 'Abitur', 'O'],
    ['<35', 'niedrig', 'Master', 'O'],
    ['>=35', 'hoch', 'Bachelor', 'M'],
    ['>=35', 'niedrig', 'Abitur', 'M'],
    ['>=35', 'hoch', 'Master', 'O'],
    ['<35', 'hoch', 'Bachelor', 'O'],
    ['<35', 'niedrig', 'Abitur', 'M'],
]
attributes = ['Alter', 'Einkommen', 'Bildung']

tree = ID3DecisionTree()
decision_tree = tree.fit(data, attributes)
print(decision_tree)
```

A02: Pruning

- Schrittweise vereinfachung von folgendem Baum

  $x_3(x_2, (x_1(C, A), x_1(B, A)), x_1(x_2(C, B), A))$

1. Absorption

   $x_3(x_2(A, A), x_1(x_2(C, B), A))$

2. Idempotenz

   $x_3(A, x_1(x_2(C, B), A))$

3. Absorption

   $x_3(A, A)$

4. Idempotenz

   A