Bäume in der Informatik

Ein Baum ist eine hierarchische Datenstruktur, die aus Knoten (Nodes) besteht. Jeder Knoten hat einen Elternknoten (außer der Wurzel) und kann mehrere Kindknoten haben.

Eigenschaften eines Baums:

- Ein spezieller Knoten, die *Wurzel* (Root), ist der Ausgangspunkt.
- Jeder Knoten (außer der Wurzel) hat genau einen Elternknoten.
- Die Knoten, die keine Kinder haben, werden *Blätter* (Leaves) genannt.
- Die *Höhe* eines Baums ist die maximale Anzahl von Kanten von der Wurzel zu einem Blatt.
- Der *Grad* eines Knotens ist die Anzahl seiner direkten Kinder.

Baumtypen

- 1. Binärbaum

  Ein Baum, bei dem jeder Knoten höchstens zwei Kinder hat (linkes und rechtes Kind).

- 2. Vollständiger Binärbaum

  Ein Binärbaum, bei dem alle Ebenen vollständig gefüllt sind, außer möglicherweise die letzte.

- 3. Balancierter Binärbaum

  Ein Binärbaum, bei dem die Höhe der beiden Teilbäume eines Knotens sich um höchstens 1 unterscheidet.

- 4. B-Baum

  Ein selbstbalancierender Suchbaum, der speziell für den Einsatz in Datenbanken optimiert ist.

Traversierungsmethoden

Traversierung bedeutet, alle Knoten des Baumes in einer bestimmten Reihenfolge zu besuchen.

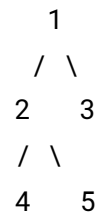
- **Tiefensuche (Depth-First Search, DFS)**

  DFS erkundet so tief wie möglich entlang eines Pfades, bevor sie zurückkehrt.

  DFS-Traversierungsarten:

  1. **Preorder (Wurzel-Links-Rechts):** Besuche die Wurzel, dann den linken Teilbaum und dann den rechten.
  2. **Inorder (Links-Wurzel-Rechts):** Besuche den linken Teilbaum, dann die Wurzel und dann den rechten Teilbaum.
  3. **Postorder (Links-Rechts-Wurzel):** Besuche den linken Teilbaum, dann den rechten Teilbaum und dann die Wurzel.

  Beispiel für Inorder-Traversierung:

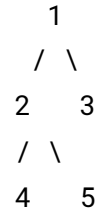
  

  Inorder: 4, 2, 5, 1, 3

- **Breitensuche (Breadth-First Search, BFS)**

  BFS erkundet den Baum Ebene für Ebene (von links nach rechts).

  Beispiel für BFS:

  BFS: 1, 2, 3, 4, 5

Anwendungen von Bäumen

- Dateisysteme
- Hierarchische Datenbanken
- Künstliche Intelligenz (Minimax-Baum)

- Netzwerke (Routing)

Entscheidungsbäume: Regression und Klassifikation

1. Entscheidungsbaum (Decision Tree)

Ein Entscheidungsbaum ist ein Modell, das zur Vorhersage von Zielwerten verwendet wird, indem es Daten in Entscheidungsbäume aufteilt. Dies geschieht durch eine Reihe von Entscheidungen basierend auf den Eingabedaten.

Wie funktioniert ein Entscheidungsbaum?

- **Knoten:** Jeder Knoten im Baum stellt eine Entscheidung dar. Im Fall der Klassifikation ist es eine Frage, die auf den Eingabedaten basiert. Im Fall der Regression ist es ein Schwellenwert, der die Daten in zwei Gruppen unterteilt.
- **Blätter:** Die Blätter des Baums enthalten die Vorhersage (z.B. eine Klasse bei Klassifikation oder ein Wert bei Regression).
- **Wurzeln:** Der oberste Knoten ist die Wurzel des Baums, von dem alle anderen Knoten abzweigen.

Schritte beim Aufbau eines Entscheidungsbaums:

1. **Auswahl der besten Feature**: Der Baum wird aufgebaut, indem er die Eingabedaten immer so aufteilt, dass die Entropie (bei Klassifikation) oder der MSE (Mean Squared Error, bei Regression) minimiert wird.
2. **Rekursive Aufteilung**: Der Baum teilt sich so lange auf, bis entweder eine bestimmte Tiefe erreicht ist oder die Splits keine Verbesserung der Vorhersage leisten.
3. **Entscheidungsregel**: Der Entscheidungsbaum trifft an jedem Knoten eine Entscheidung basierend auf einer Eingangsvariable. Beispielsweise könnte ein Baum für die Klassifikation von E-Mails als Spam oder Nicht-Spam bei einem Knoten fragen: “Enthält die E-Mail das Wort ‘Gratis’?” Bei der Regression könnte er fragen: “Liegt die Temperatur über 20 Grad?”

Vorteile von Entscheidungsbäumen:

- **Einfachheit**: Entscheidungsbäume sind leicht verständlich und interpretierbar.
- **Keine Normalisierung nötig**: Entscheidungsbäume benötigen keine Normalisierung der Eingabedaten.
- **Kann auch mit kategorialen Variablen umgehen**.

Nachteile:

- **Überanpassung**: Entscheidungsbäume tendieren dazu, bei zu tiefen Bäumen zu überanpassen (Overfitting).
- **Instabilität**: Kleine Änderungen in den Eingabedaten können zu großen Veränderungen im Modell führen.

2. Regression mit Entscheidungsbaum

Die **Regressionsanalyse** mit Entscheidungsbäumen funktioniert ähnlich wie die Klassifikation, aber anstatt eine Klasse vorherzusagen, wird ein kontinuierlicher Wert geschätzt.

Beispiel:

Nehmen wir an, wir haben eine Reihe von Datenpunkten, bei denen die Eingabewerte (z.B. Temperaturen) und die Zielwerte (z.B. Energieverbrauch) bekannt sind. Ein Entscheidungsbaum zur Regression könnte den Temperaturwert als Eingabe verwenden und den Energieverbrauch als Vorhersagewert zurückgeben.

- Jeder Split im Baum minimiert die Varianz innerhalb der Gruppen (also die Differenz zwischen den Vorhersagen und den tatsächlichen Werten).
- Die Blätter des Entscheidungsbaums enthalten dann den Durchschnitt der Zielwerte der Eingabedaten, die in diesen Bereich fallen.

Beispiel:

Angenommen, der Entscheidungsbaum fragt: “Ist die Temperatur über 20 Grad?”, und teilt dann die Daten in zwei Gruppen auf:

- Wenn ja: Durchschnittlicher Energieverbrauch = 120 kWh.
- Wenn nein: Durchschnittlicher Energieverbrauch = 80 kWh.

Die Vorhersage des Modells basiert dann auf diesen Durchschnittswerten.

3. Klassifikation mit Entscheidungsbaum

Bei der **Klassifikation** wird ein Entscheidungsbaum verwendet, um die Eingabedaten in vordefinierte Kategorien oder Klassen zu unterteilen.

Beispiel:

Angenommen, wir möchten eine E-Mail als “Spam” oder “Nicht-Spam” klassifizieren. Der Entscheidungsbaum könnte beispielsweise folgende Entscheidungen treffen:

- **Wurzelknoten**: Enthält die E-Mail das Wort “Gratis”?
  - Ja: Gehe zum linken Knoten.
  - Nein: Gehe zum rechten Knoten.
- **Linker Knoten**: Enthält die E-Mail den Begriff “Gewonnen”?
  - Ja: Klassifiziere die E-Mail als Spam.
  - Nein: Klassifiziere die E-Mail als Nicht-Spam.

Hierbei wird an jedem Knoten des Entscheidungsbaums entschieden, ob eine E-Mail Spam ist oder nicht, basierend auf den Wörtern und Phrasen in der Nachricht.

4. Hyperparameter von Entscheidungsbäumen

Beim Trainieren eines Entscheidungsbaums gibt es mehrere Hyperparameter, die das Modell beeinflussen:

- **max_depth**: Maximale Tiefe des Baums. Ein zu hoher Wert kann zu Überanpassung führen, während ein zu niedriger Wert dazu führt, dass das Modell zu einfach ist.
- **min_samples_split**: Die Mindestanzahl an Datenpunkten, die in einem Knoten vorhanden sein müssen, um den Knoten weiter zu teilen.
- **min_samples_leaf**: Die Mindestanzahl an Datenpunkten, die in einem Blatt vorhanden sein müssen.
- **criterion**: Das Kriterium, das verwendet wird, um die “Qualität” eines Splits zu bewerten. Für Klassifikation wird in der Regel “Gini” oder “Entropie” verwendet, und für Regression “MSE” (Mean Squared Error).

5. Entscheidungsbaum für Klassifikation vs. Regression

- **Klassifikation**: Hier werden die Blätter des Baums mit den häufigsten Zielklassen (z.B. “Spam”, “Nicht-Spam”) beschriftet. Die Klassifikation basiert auf der Mehrheit der Datenpunkte in einem Blatt.
- **Regression**: Die Blätter sind mit dem Durchschnitt der Zielwerte für die Datenpunkte in diesem Blatt beschriftet.

Beispiel Code für einen Entscheidungsbaum in Python (Sklearn)

```
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, plo
from sklearn.datasets import load_iris
import numpy as np

# Beispiel für Klassifikation (Iris-Datensatz)
data = load_iris()
X = data.data
y = data.target

# Entscheidungsbaum für Klassifikation
clf = DecisionTreeClassifier(max_depth=3)
clf.fit(X, y)

# Beispiel Vorhersage
print(clf.predict([[5.1, 3.5, 1.4, 0.2]]))

# Beispiel für Regression
X = np.array([[1], [2], [3], [4]])
y = np.array([1, 4, 9, 16])

# Entscheidungsbaum für Regression
reg = DecisionTreeRegressor(max_depth=3)
reg.fit(X, y)

# Vorhersage
print(reg.predict([[5]]))

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 8))

plot_tree(reg, filled=True, ax=ax1)
ax1.set_title("Regressor Tree")

plot_tree(clf, filled=True, ax=ax2)
ax2.set_title("Classifier Tree")

plt.show()
```