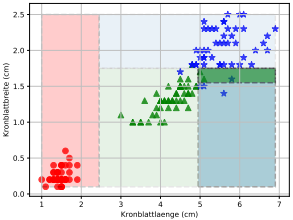
Regression mit dem CART Algorithmus und Pruning

Prof. Dr. Jörg Frochte

Maschinelles Lernen

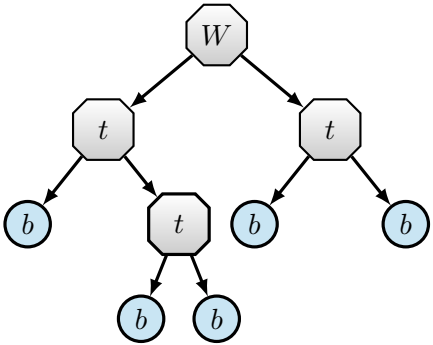


$$G = 1 - \sum_{i=1}^{c} N(i)^2$$

CART-Algorithmus

Wir haben den CART (Classification and Regression Trees) als einen Algorithmus zum Lernen eines binären Entscheidungsbaum kennengelernt.

- Bisher haben wir uns beim CART auf die Klassifikation konzentriert.
- Dieses Mal geht es um die Regression und ...
- ... das Problem der Überanpassung.



Unterschiede bei einer Regression

- Im Fall einer Regression besteht der Datensatz aus dem Featurevektor x und dem Funktionswert $y = f(x) \in \mathbb{R}$.
- Zwei Änderungen sind nötig für den Wechsel von der Klassifikation zur Regression:
  - Einmal brauchen wir einen Ersatz für die Gini Impurity und
  - zum anderen eine andere Art, den Wert an einem Blattknoten zu berechnen.
- Zunächst ersetzen wir die Gini Impurity durch ein Regressionsmaß wie z.B. den den mittleren quadratischen Fehler (MSE):

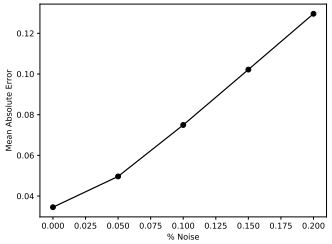
$$r_y = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2 \text{ mit } \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

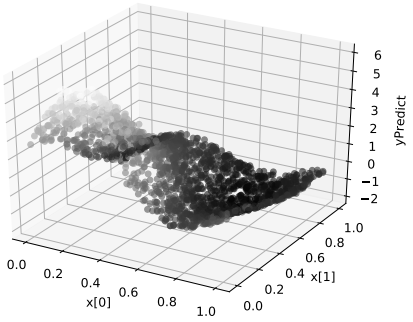
- Der Wert am Blattknoten ist einfach der Mittelwert $\bar{y}$. Über jeweils den Bereich, den ein Blattknoten abdeckt, ist die so enstandene Funktion also konstant.

- Anstatt dem MSE, könnte man als Optimierungskriterium auch die Residuenquadratsumme verwenden. Der Unterschied ist hierbei nur die Skalierung mit der Anzahl der verwendeten Beispiele.
- Das erste ist eine analytische Funktion

$$y = (\sin(2\pi x_0) + \cos(\pi x_1)) \cdot e^{1-x_0^2-x_1^2},$$

  die wir mit verschiedenen Stärken von additivem weißen Rauschen versehen.
- Rechts oben sehen wir die Regression mittels CART für unterschiedlich starkes Rauschen und `minLeafNodeSize=3`
- Statt einem konstanten Wert wie auf der letzten Folie wäre es auch möglich in jedem Knoten ein lineares Modell zu verwenden, wenn genug Sampels vorliegen.





Approximation mit 20% Rauschen

Beispiel

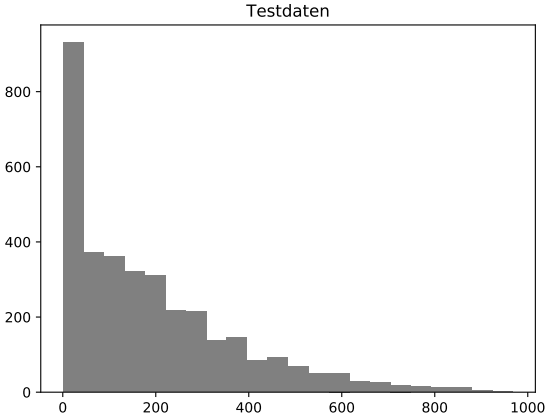
- Als Beispiel verwenden wir das **Bike Sharing Data Set** (`https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset`).
- Es besteht aus 17389 protokollierten Daten über das Ausleihverhalten von Fahrrädern in einer Großstadt.
- Der Datenbestand enthält Daten bzgl. des Wetters und des Ausleihverhaltens für Fahrräder protokolliert nach Uhrzeit, Feiertag usw.
- Die Originaldatei enthält die Informationen zum Datum bzw. der Uhrzeit teilweise redundant in einem Zeitstempel und einzelnen Spalteneinträgen.
- In dieser bereinigten Form haben wir 13 Merkmale, die in Tabelle auf der nächsten Seite notiert sind.

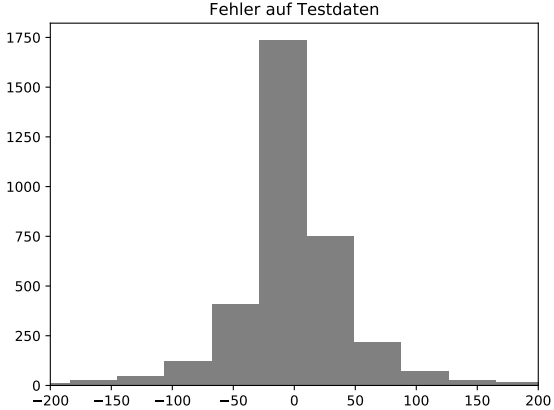
Merkmale des Bike Sharing Data Set (2013)

Nr.	Merkmal	Bedeutung	Wertebereich
0	season	Frühling(1), Sommer(2), Herbst(3), Winter(4)	$\{1, 2, 3, 4\}$
1	yr	Jahr 2011 (0) oder 2012 (1)	$\{0, 1\}$
2	mnth	Monat des Jahres	1 bis 12
3	day	Tag des Monats	1 bis 31
4	hr	Stunde des Tages	0 bis 23
5	holiday	Ist es ein Feiertag?	0 (False) oder 1 (True)
6	weekday	Welcher Wochentag	$\{1, 2, 3, 4, 5, 6, 7\}$
7	workingday	Kein Wochenende und kein Feiertag?	0 (False) oder 1 (True)
8	weathersit	Qualität des Wetters in Abstufungen	$\{1, 2, 3, 4\}$
9	temp	Normierte Temperatur	$[0, 1]$
10	atemp	Normierte gefühlte Temperatur	$[0, 1]$
11	hum	Normierte relative Luftfeuchtigkeit	$[0, 1]$
12	windspeed	Normierte Windgeschwindigkeit	$[0, 1]$
13	casual	Anzahl Fahrräder von Gelegenheitsradlern	$\in \mathbb{N}$
14	registered	Anzahl Fahrräder von registrierten Nutzern	$\in \mathbb{N}$
15	cnt	Gesamtanzahl verliehener Fahrräder	$\in \mathbb{N}$

- Dieser Datenbestand ist ein ganz realistischer Fall, in dem nicht alle Merkmale mit der von uns gesuchten Größe korrelieren müssen und sicherlich mehrere voneinander nicht statistisch unabhängig sind.
- Beispielsweise sind natürlich die Temperatur und die gefühlte Temperatur nicht statistisch unabhängig.
- Auch typisch ist, dass wir hier sehr unterschiedliche Arten von Merkmalen haben.
- Wie man sieht, sind die Merkmale 5 (holiday) und 7 (workingday) einer Nominalskala und 8 (temp) einer Ordinalskala zuzuordnen.
- Trotz der Tatsache, dass nicht alle Merkmale rational sind, müssen wir an unserem Code nichts ändern.
- In der Praxis ist man hier oft weit entspannter, man muss sich nur die Auswirkungen und Gefahren klar machen.

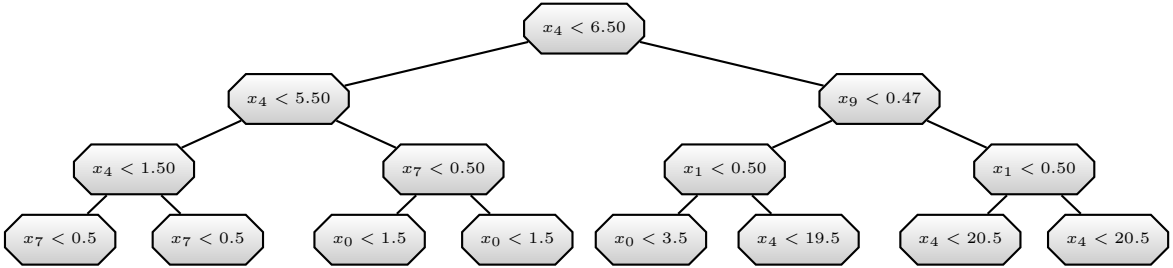
Ergebnisse für den CART-Baum





Wir erhalten hier einen mittleren Fehler von ca. 31 Fahrrädern, um die sich die Vorhersage auf der Testmenge verschätzt.

Erste Ebenen des CART Entscheidungsbaumes



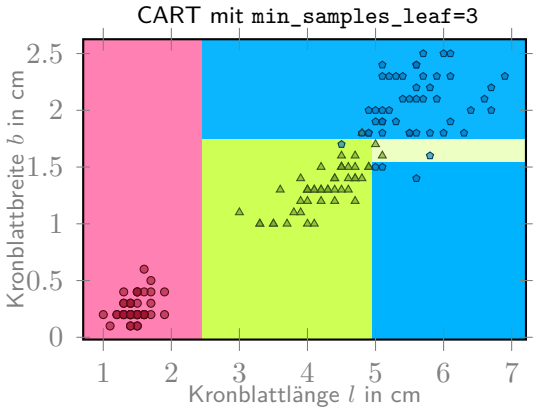
- Wie man am oberen Ende des Baumes sieht, ist die Uhrzeit – Merkmal 4 – ein sehr wichtiger Aspekt.
- Ansonsten spielen die Jahreszeit (Nr. 0), das Jahr (Nr. 1), Feiertag: ja oder nein (Nr. 7) und die Temperatur (Nr. 9) eine große Rolle.
- Andere Merkmale kommen erst weiter unten als Feinabstimmung im Baum vor.

Einordnung der Ergebnisse

- Wie man sieht, schwankt die Anzahl der entliehenen Fahrräder beträchtlich.
- Im Mittel ist unser Ergebnis eigentlich sehr gut, was jedoch daran liegt, wie wir unser Testset gebildet haben.
- Unsere Testmenge wurde zufällig aus der Gesamtmenge der Daten gezogen.
- Das bedeutet, dass zur Trainingsmenge zum Beispiel die ausgeliehenen Fahrräder an einem bestimmten Tag um 09:00, 10:00 und 12:00 gehören und die Testmenge den Wert um 11:00 enthält.
- Eine solche Interpolation auf zeitlichen Daten ist wesentlich leichter als eine Extrapolation.

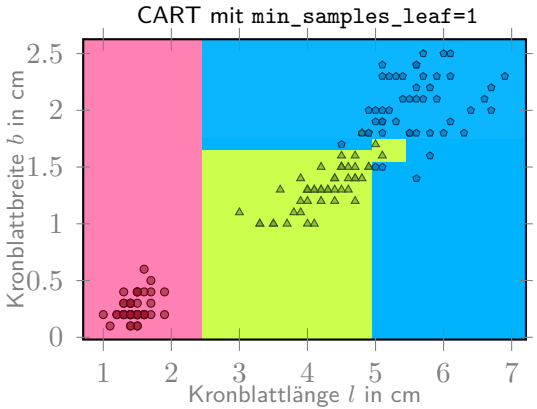
Overfitting

- Ohne Einschränkung, wann CART aufhören soll, bricht der Algorithmus erst ab, wenn keine Verbesserung mehr erreicht werden kann.



Overfitting

- Ohne Einschränkung, wann CART aufhören soll, bricht der Algorithmus erst ab, wenn keine Verbesserung mehr erreicht werden kann.
- Der Baum passt sich den Trainingsdaten bestmöglichst an.
- Bei verrauschten oder schwer zu trennenden Daten ist das nicht gewollt und nennt sich *overfitting*.



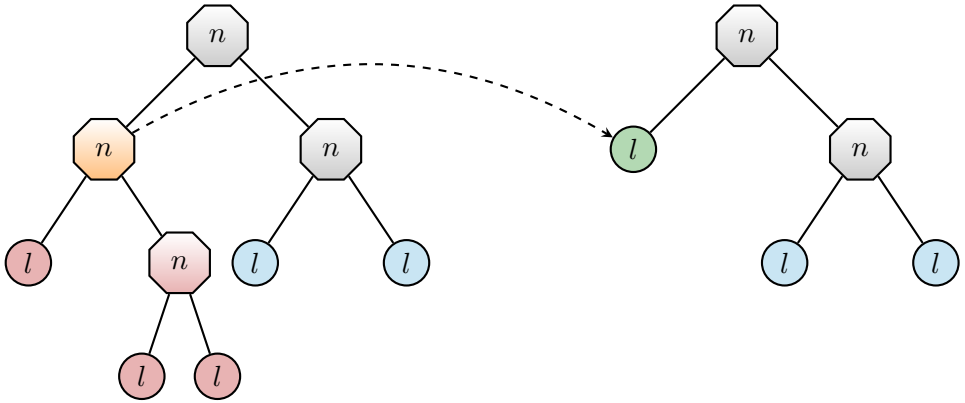
Pre-Pruning

- Overfitting wirkt sich im Allgemeinen negativ auf die Verallgemeinerung von Daten aus– anders gesagt: Die Trainingsdaten werden auswendig gelernt.
- Das gilt für beiden Anwendungsgebiete also Regression und Klassifikation.
- Unser Ziel ist es also, diese unsinnige Verästelung zu reduzieren und den Baum auf die notwendige Komplexität zurechtzustutzen.
- Dies nennt man **Pruning**. Man unterscheidet zwischen **Pre-Pruning** und **Post-Pruning**.
- Beim **Pre-Pruning** wird eigentlich nichts gestutzt, sondern beim Aufstellen des Baumes zu starke Verästelung vermieden.
- Typische Beispiele sind die maximale Tiefe des Baumes, Mindestverbesserungsraten oder Mindestgrößen für die Blätter (Anzahl der Samples)
- Wir haben also schon etwas *Pre-Pruning*

Post-Pruning

- Beim **Post-Pruning** werden an einem fertigen Baum Knoten durch Blätter ersetzt, um die Komplexität zu verbessern.
- Die Beurteilung, was zurückgeschnitten wird, geschieht auch hierbei über eine Validierungsmenge.
- Ein einfacher und trotzdem sehr effektiven Ansatz ist der **Reduced-Error Ansatz**.
- Hierbei testet man einen Knoten innerhalb des Baumes darauf, wie sich der Fehler auf der Validierungsmenge entwickelt, wenn dieser Knoten durch ein Blatt ersetzt wird.
- Das Blatt wird dabei nach den allgemeinen Regeln für den Baum gebildet, also zum Beispiel nach einer Mehrheitsentscheidung für die Klassifizierung oder eine Mittelwertbildung für die Regression.
- Verbessert sich der Baum durch diesen Rückschnitt auf der Validierungsmenge oder ist der Fehlerzuwachs in einem tolerierbaren Maß, so wird der unter t liegende Teilbaum abgeschnitten und durch ein Blatt ersetzt.

Post-Pruning illustriert



Komplexität

Nun bleibt die Frage wie der Algorithmus für größere Datenmengen in der Theorie skaliert? Für diese Frage muss man das Lernen und die Auswertung des Baumes unterscheiden.

- Sei n_S die **Anzahl Samples in der Trainingsmenge** und n_F die **Anzahl Merkmale**.
- Wir nehmen an, dass ein perfekt ausbalancierter Baum entsteht.
- Die Komplexität für eine Auswertung liegt bei $\mathcal{O}(\log(n_S))$, weil die Tiefe des Baums mit n_S logarithmisch wächst.
- Die Komplexität für das Training beträgt:

$$\mathcal{O}(n_F \cdot n_S^2 \cdot \log(n_S)) \text{ mit zusätzlichen Annahmen jedoch } \mathcal{O}(n_F \cdot n_S \cdot \log(n_S))$$