Gradient Decision Tree Boosting (GDTB)

By Bastian Berle & Ron Holzapfel

Idee

Gradient Decision Tree Boosting ist eine Ensemble Methode der Entscheidungsbäume. Sie kann für Regressions- und Klassifizierungsaufgaben eingesetzt werden. Dabei werden, basierend auf einem schwachen Baum, neue Modelle zur Kompensation der Fehler zuvor trainierter Bäume erstellt. Das Verfahren nutzt folgende Komponenten:

- Loss Function,
- Weak Learner, und
- Additive Model.

Vor- und Nachteile

Vorteile

- Flexibel einsetzbar für Klassifikations- und Regressionsprobleme
- Unterstützt unterschiedliche differenzierbare Fehlerfunktionen
- Komplexe Zusammenhänge in Trainingsdaten werden erlernt

Nachteile

- Optimierung der Hyperparameter in der Regel sehr rechenintensiv
- Sequentielles Training des Modells erlaubt keine Parallelisierung
- Überanpassung bei verrauschten Trainingsdatensätzen möglich

Python Code

```
from sklearn.ensemble import
    GradientBoostingRegressor
from sklearn.ensemble import
    GradientBoostingClassifier

GradientBoostingRegressor().fit(X, y)
GradientBoostingClassifier().fit(X, y)
```

Vergleich

AdaBoost Sowohl AdaBoost als auch GDTB nutzen als Basis einen weak learner und versuchen iterativ die Performance des weak learners durch Anpassung bei abweichenden Vorhersagen zu verbessern. Der Unterschied liegt im Umgang mit diesen Vorhersagen. Während AdaBoost die abweichenden Vorhersagen stärker gewichtet, eliminiert GDTB die Restfehler des vorherigen Modells durch ein ergänzendes.

Random Forest GDTB und Random Forests sind beide Ensembling Methoden, die Regression oder Klassifizierungen durchführen können, indem Ausgaben von mehreren Bäume verbunden werden. Im Gegensatz zu GDTB erstellt der Random Forest seine Bäume zufällig.

Algorithmus

Input: Trainingsdaten $\{(x_i, y_i)\}_{i=1}^N$ und differenzierbare Fehlerfunktion $L(y, f(x))$.

1. Schritt: Initialisiere konstantes Modell
$f_0(x) = \underset{\gamma}{arg\,min} \sum_{i=1}^N L(y_i, \gamma)$.

2. Schritt: Für $m = 1, 2, \ldots, M$:

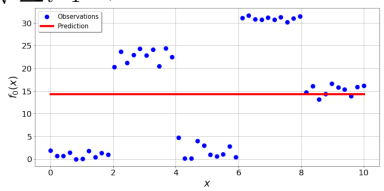
1. Berechne Pseude-Residuen:
   $r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)}$,
   $i = 1, 2, \ldots, N$.
2. Trainiere einen Entscheidungsbaum auf den Trainingsdaten $\{(x_i, r_{im})\}_{i=1}^N$.
   $\rightarrow$ Feature-Raum wird in die Regionen R_{jm} mit $j = 1, 2, \ldots, J_m$
3. Berechne Werte in Blättern:
   $\gamma_{jm} = \underset{\gamma}{arg\,min} \sum_{x_i \epsilon R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$,
   $j = 1, 2, \ldots, J_m$
4. Passe das Regressionsmodell an:
   $f_m(x) = f_{m-1}(x) + \alpha \sum_{j=1}^{J_m} \gamma_{jm} I(x \epsilon R_{jm})$

3. Schritt: Output $\hat{f}(x) = f_M(x)$

Vorgehen mit Beispiel

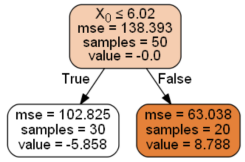
1. Initialisiere einen weak Learner (Beispiel: $f_0(x) = \frac{1}{N}\sum_{i=1}^N y_i$)

i	x_i	y_i	$f_0(x_i)$
1	0.0	1.88	14.3
2	0.2	0.64	14.3
3	0.4	0.69	14.3
⋮	⋮	⋮	⋮
50	10	16.1	14.3



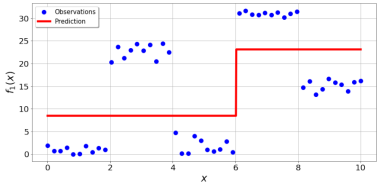
2. Berechne Residuen $r_{i1} = y_i - f_0(x_i)$ und trainiere neues Modell auf $\{(x_i, r_{i1})\}_{i=1}^N$, das aus Fehlern des alten lernt

i	x_i	y_i	$f_0(x_i)$	r_{i1}	$h_1(x_i)$
1	0.0	1.88	14.3	−12.42	−5.858
2	0.2	0.64	14.3	−13.66	−5.858
3	0.4	0.69	14.3	−13.61	−5.858
⋮	⋮	⋮	⋮	⋮	⋮
50	10	16.1	14.3	1.8	8.788



3. Kombiniere die weak Learner zu einem stärkeren Modell mit $f_1(x_i) = f_0(x_i) + h_1(x_i)$

i	x_i	y_i	$f_1(x_i)$
1	0.0	1.88	8.442
2	0.2	0.64	8.442
3	0.4	0.69	8.442
⋮	⋮	⋮	⋮
50	10	16.1	23.088



4. Wiederhole Schritt 2 und 3 bis gewählte Abbruchbedingung eintritt