Boosting

Autor und Herausgeber

Alexander Jung*, Department of Computer Science, Aalto University, Espoo, Finland (ORCID 0000-0001-7538-0990)
Editors: Konstantina Olioumtsevits (ORCID 0000-0003-2697-8659), Ekkehard Schnoor (ORCID 0000-0001-8252-1499), Aalto University, Espoo, Finland
*Corresponding author: alex.jung@aalto.fi

Zusammenfassung

Boosting ist eine Ensemble-Methode, die einfache Basislerner, auch schwache Lerner genannt, sequenziell kombiniert. Bei jeder iteration verwendet ein Basislerner einen modifizierten Trainingsdatensatz, der aus dem ursprünglichen Trainingsdatensatz und den aktuellen predictions konstruiert wird. Das gradient boosting ersetzt die Labels durch negative Verlust-partial derivatives, während das adaptive boosting (AdaBoost) die Labels beibehält und die weights der Datenpunkte ändert. Die resultierende Hypothese ist eine additive Kombination der gelernten Korrekturen.

Definition

Die Wetterstation Krems an der Donau (Österreich) zeichnet stündlich die Lufttemperatur auf. Jeder Tag ist ein Datenpunkt mit dem Messwert von 06:00 (UTC) als Merkmal x und dem Messwert von 15:00 (UTC) als Label y (Fig. 1). Die Abbildung zeigt auch eine Hypothese, die von einem Entscheidungsbaum der Tiefe eins gelernt wurde. Dieser Baum ist zu einfach, um die Beziehung zwischen x und y zu erfassen. Boosting verbessert die Hypothese, indem es eine weitere Hypothese hinzufügt, die ihre prediction-Fehler korrigiert. Die beiden Kurven entsprechen den ersten beiden iterations dieses Prozesses.

Das Boosting ist eine Ensemble-Methode, die Basislerner sequenziell kombiniert, um eine genauere Hypothese zu erhalten (Hastie et al., 2009, Ch. 10). Die Basislerner sind typischerweise einfache Modelle, wie flache Entscheidungsbäume, und werden auch schwache Lerner genannt.

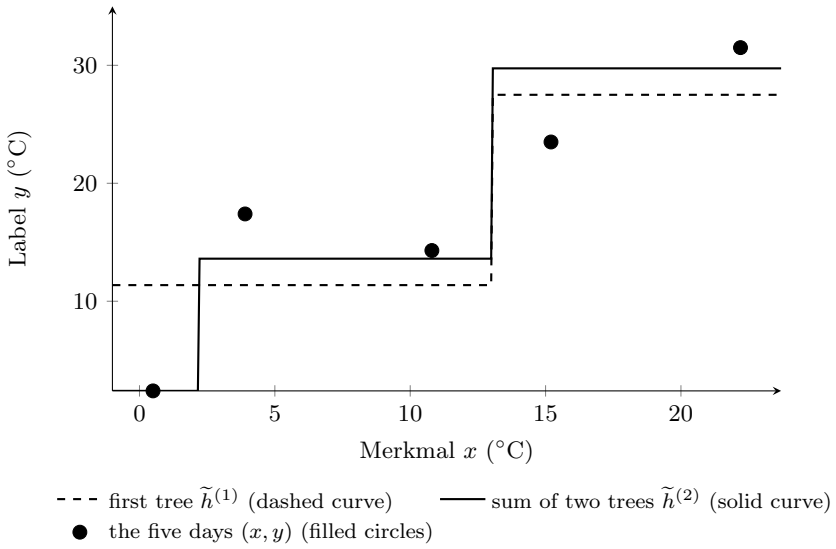


Abbildung 1: Zwei iterations des Boosting an fünf Tagen Wetterdaten in Krems an der Donau. Die gestrichelte Kurve ist die Hypothese, die vom ersten Entscheidungsbaum der Tiefe eins gelernt wurde. Der zweite Entscheidungsbaum wird aus seinen Residuen gelernt. Das Hinzufügen dieses Entscheidungsbaum ergibt die durchgezogene Kurve und senkt den mittleren quadratischen prediction-Fehler von 31.5 auf 11.4. Daten generiert von `pythondemos/boosting.py`

Im Gegensatz zu Ensemble-Methoden wie Bagging und stacking trainiert Boosting seine Basislerner sequenziell. Bei iteration t lernt ein Basislerner eine Hypothese $\widehat{h}^{(t)}$ aus einem modifizierten Trainingsdatensatz $\widetilde{\mathcal{D}^{(\text{train})}}^{(t)}$. Dieser Trainingsdatensatz wird aus dem ursprünglichen Trainingsdatensatz $\mathcal{D}^{(\text{train})}$ und den predictions von $\widetilde{h}^{(t-1)}$ konstruiert (Fig. 2).

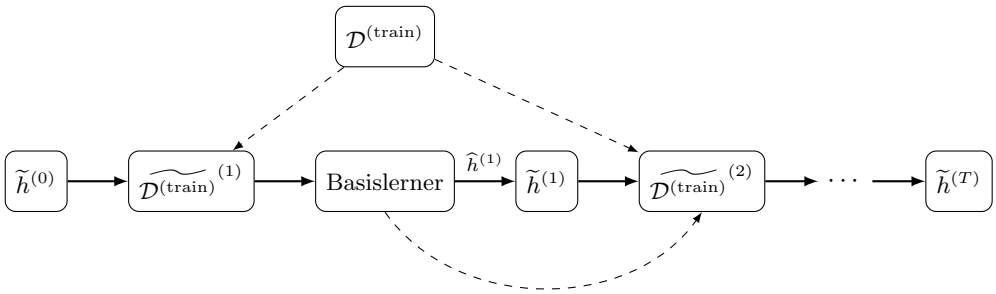


Abbildung 2: Boosting konstruiert jeden modifizierten Trainingsdatensatz $\widetilde{\mathcal{D}^{(\text{train})}}^{(t)}$ aus dem ursprünglichen Trainingsdatensatz $\mathcal{D}^{(\text{train})}$ und der aktuellen aggregierten Hypothese $\widetilde{h}^{(t-1)}$. Die gepunkteten Pfeile zeigen, dass der ursprüngliche Trainingsdatensatz und der vorhergehende Basislerner den nächsten modifizierten Trainingsdatensatz beeinflussen. Ein Basislerner lernt die nächste Korrektur aus diesem Trainingsdatensatz

Boosting-Methoden beginnen mit einer initialen Hypothese $\widetilde{h}^{(0)}$, die iterativ durch das Update verbessert wird

$$\widetilde{h}^{(t)} = \widetilde{h}^{(t-1)} + \eta^{(t)}\widehat{h}^{(t)}, \qquad t = 1, \ldots, T. \tag{1}$$

Hier ist $\eta^{(t)}$ eine Lernrate und $\widehat{h}^{(t)}$ die Hypothese des t-ten Basislerner (siehe Fig. 3). Die additiven Updates (1) bilden daher eine sequence von aggregierten Hypothesen.

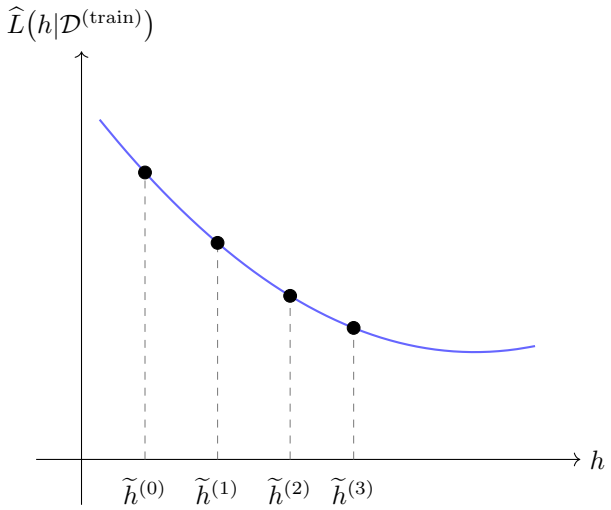


Abbildung 3: Boosting konstruiert eine sequence von Hypothese-Abbildungen. Jede neue Hypothese-Abbildung $\widetilde{h}^{(t+1)}$ wird von einem Basislerner $\widehat{h}^{(t+1)}$ erhalten, der darauf abzielt, die prediction-Fehler der aktuellen Hypothese $\widetilde{h}^{(t)}$ zu korrigieren

Verschiedene Boosting-Methoden konstruieren $\widetilde{\mathcal{D}^{(\text{train})}}^{(t)}$ unterschiedlich. Das gradient boosting ersetzt das Label y jedes Datenpunkt $(\mathbf{x}, y)$ durch die negative partial derivative seines Verlust bezüglich der aktuellen prediction,

$$- \left. \frac{\partial L\left((\mathbf{x}, y), z\right)}{\partial z} \right|_{z=\widetilde{h}^{(t-1)}(\mathbf{x})}.$$

Diese Werte sind Pseudo-Labels, auch negative Gradienten genannt. Diese Konstruktion gilt, wenn die loss function von der Hypothese über ihre prediction abhängt. Für den squared error loss sind die Pseudo-Labels proportional zu den prediction-Fehlern $y - \widetilde{h}^{(t-1)}(\mathbf{x})$. Algorithmus 1 fasst eine grundlegende Form des gradient boosting zusammen (Hastie et al., 2009, Algorithm 10.3).

Algorithm 1 Das gradient boosting (Friedman, 2001, Algorithm 1)

Input: Trainingsdatensatz $\mathcal{D}^{(\text{train})} = \left\{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(m)}, y^{(m)})\right\}$, Basislerner, loss function $L\left((\mathbf{x}, y), z\right)$, Lernraten $\eta^{(1)}, \ldots, \eta^{(T)}$, iteration count T

1: choose a constant Hypothese $\widetilde{h}^{(0)}$ minimizing the Trainingsdatensatz Verlust
2: **for** $t = 1, \ldots, T$ **do**
3: compute the pseudo-Labels $\widetilde{y}^{(r,t)} := -\left.\dfrac{\partial L\left((\mathbf{x}^{(r)}, y^{(r)}), z\right)}{\partial z}\right|_{z=\widetilde{h}^{(t-1)}(\mathbf{x}^{(r)})}$ for $r = 1, \ldots, m$
4: learn $\widehat{h}^{(t)}$ by applying the Basislerner to the Merkmalsvektoren $\mathbf{x}^{(r)}$ with $\widetilde{y}^{(r,t)}$ as Labels
5: update $\widetilde{h}^{(t)}$ according to the additive update (1)
6: **end for**

Output: $\widetilde{h}^{(T)}$

Das gradient boosting kann als Erweiterung des Gradientenabstieg von Updates eines endlichdimensionalen model parameter zu Updates in einem Hypothesenraum betrachtet werden. Für den squared error loss und einen Basislerner, der lineare Regression verwendet, ist diese Verbindung unter der unten angegebenen Orthogonalitäts- und Normalisierungsbedingung exakt. Bei iteration t lernt Algorithmus 1 eine linear map, die die Residuen $y^{(r)} - \left(\mathbf{x}^{(r)}\right)^\top \mathbf{w}^{(t-1)}$ für $r = 1, \ldots, m$ vorhersagt. Wenn die Merkmalsmatrix $\mathbf{X} = \left(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\right)^\top$ vollen Spalten-rank hat, ist der model parameter dieser Abbildung die eindeutige Kleinste-Quadrate-Lösung (Hastie et al., 2009, Sect. 3.2, Eq. (3.6))

$$\Delta\mathbf{w}^{(t)} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\left(\mathbf{y} - \mathbf{X}\mathbf{w}^{(t-1)}\right).$$

Wenn die Spalten von $\mathbf{X}$ auch orthogonal und so normiert sind, dass $\mathbf{X}^\top\mathbf{X} = m\mathbf{I}$, dann

$$\Delta\mathbf{w}^{(t)} = \frac{1}{m}\mathbf{X}^\top\left(\mathbf{y} - \mathbf{X}\mathbf{w}^{(t-1)}\right),$$

was der negative Gradient bei $\mathbf{w}^{(t-1)}$ des skalierten mittleren squared error loss $f(\mathbf{w}) := \frac{1}{2m}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$ ist. Der Faktor $1/2$ hebt den Faktor 2 auf, der beim Differenzieren des Quadrats entsteht. Bei passender Initialisierung und Lernraten erfüllen die Boosting-Iterationen $\widetilde{h}^{(t)} = h^{(\mathbf{w}^{(t)})}$ und stimmen mit den Gradientenabstieg-Iterationen überein.

Das adaptive boosting (AdaBoost) wendet Boosting auf die binäre Klassifikation an (Freund and Schapire, 1997). Wie das gradient boosting in Algorithmus 1 verwendet es das additive Update (1). Es verwendet jedoch eine andere Konstruktion für den Trainingsdatensatz $\widetilde{\mathcal{D}^{(\text{train})}}^{(t)}$: Die prediction-Fehler von $\widetilde{h}^{(t-1)}$ bestimmen weights, wobei falsch klassifizierten Datenpunkte größere weights zugewiesen werden.

Algorithmus 2 stellt AdaBoost.M1 vor (Hastie et al., 2009, Algorithm 10.1).

Algorithm 2 Das AdaBoost.M1 (Hastie et al., 2009, Algorithm 10.1)

Input: Trainingsdatensatz $\mathcal{D}^{(\text{train})} = \left\{(\mathbf{x}^{(r)}, y^{(r)})\right\}_{r=1}^{m}$ with $y^{(r)} \in \{-1, 1\}$, Hypothesenräume $\mathcal{H}^{(1)}, \ldots, \mathcal{H}^{(T)}$ of Hypothesen $h : \mathcal{X} \to \{-1, 1\}$, iteration count T

1: $q^{(r)} := 1/m$ for $r = 1, \ldots, m$
2: $\widetilde{h}^{(0)}(\mathbf{x}) := 0$
3: **for** $t = 1, \ldots, T$ **do**
4: $\quad \widehat{h}^{(t)} \in \underset{h \in \mathcal{H}^{(t)}}{\arg\min} \sum_{r=1}^{m} q^{(r)} L^{(0/1)}\left((\mathbf{x}^{(r)}, y^{(r)}), h\right)$
5: $\quad \epsilon^{(t)} := \sum_{r=1}^{m} q^{(r)} L^{(0/1)}\left((\mathbf{x}^{(r)}, y^{(r)}), \widehat{h}^{(t)}\right)$
6: $\quad \eta^{(t)} := \frac{1}{2} \ln\left(\frac{1-\epsilon^{(t)}}{\epsilon^{(t)}}\right)$
7: $\quad$ update $\widetilde{h}^{(t)}$ according to the additive update (1)
8: $\quad q^{(r)} := q^{(r)} \exp\left(-\eta^{(t)} y^{(r)} \widehat{h}^{(t)}(\mathbf{x}^{(r)})\right)$ for $r = 1, \ldots, m$
9: $\quad$ normalize the weights so that $\sum_{r=1}^{m} q^{(r)} = 1$
10: **end for**

Output: $\widehat{h}(\mathbf{x}) = \text{sign}\left(\widetilde{h}^{(T)}(\mathbf{x})\right)$

Zu Beginn der iteration t erfüllen die normalisierten weights

$$q^{(r)} = \frac{\exp\left(-y^{(r)} \widetilde{h}^{(t-1)}(\mathbf{x}^{(r)})\right)}{\sum_{r'=1}^{m} \exp\left(-y^{(r')} \widetilde{h}^{(t-1)}(\mathbf{x}^{(r')})\right)}.$$

Da die normalisierten sample-weights $q^{(r)}$ in der vorhergehenden Darstellung zu 1 summieren, ist die in Algorithmus 2 definierte Größe $\epsilon^{(t)}$ der gewichtete mittlere 0/1 loss von $\widehat{h}^{(t)}$ auf dem bei iteration t verwendeten Trainingsdatensatz.

Eine einzelne iteration des AdaBoost reduziert den mittleren exponentiellen Verlust, welcher das empirical risk ist

$$\widehat{L}\left(\widetilde{h}^{(t)} | \mathcal{D}^{(\text{train})}\right) := \frac{1}{m} \sum_{r=1}^{m} \exp\left(-y^{(r)} \widetilde{h}^{(t)}(\mathbf{x}^{(r)})\right).$$

Das Einsetzen des additiven Updates (1) in dieses empirical risk ergibt

$$\widehat{L}\left(\widetilde{h}^{(t)} | \mathcal{D}^{(\text{train})}\right) = \kappa^{(t)} \widehat{L}\left(\widetilde{h}^{(t-1)} | \mathcal{D}^{(\text{train})}\right),$$

wobei

$$\kappa^{(t)} = (1 - \epsilon^{(t)}) e^{-\eta^{(t)}} + \epsilon^{(t)} e^{\eta^{(t)}} = 2\sqrt{\epsilon^{(t)}(1 - \epsilon^{(t)})}.$$

Wenn also $0 < \epsilon^{(t)} < 1/2$, dann ist $0 < \kappa^{(t)} < 1$. Für diese iteration ist die skalare Multiplikation mit $\kappa^{(t)}$ ein contractive operator auf dem mittleren exponentiellen Verlust. Da der 0/1 loss durch den exponentiellen Verlust begrenzt ist,

reduzieren wiederholte Kontraktionen auch eine obere Schranke für den Trainingsfehler (Schapire and Freund, 2012, Thm. 3.1). Dieses Argument betrifft den empirischen Verlust; es besagt nicht, dass die Hypothesen selbst eine kontraktive sequence bilden.

Siehe auch: Ensemble, AdaBoost, gradient boosting, Basislerner, Bagging, contractive operator.

Querverweise

Literatur

1. Freund and Schapire (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. J. Comput. Syst. Sci., 55(1), pp. 119–139.

2. Friedman (2001). Greedy Function Approximation: A Gradient Boosting Machine. Ann. Statist., 29(5), pp. 1189–1232.

3. Schapire and Freund (2012). Boosting: Foundations and Algorithms. MIT Press, Cambridge, MA, USA.

4. Hastie et al. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd ed., Springer Science+Business Media, New York, NY, USA.