Theorie des maschinellen Lernens

Hans U. Simon

16. Juni 2017

18 Entscheidungsbäume

Ein Entscheidungsbaum T stellt eine Voraussagefunktion $h_T : \mathcal{X} \to \mathcal{Y}$ dar. Stellen wir uns der Einfachheit halber einen binären Entscheidungsbaum T vor. Dann gilt folgendes:

- Jeder innere Knoten von T repräsentiert eine Eigenschaft, die eine Instanz $x \in \mathcal{X}$ entweder hat oder nicht hat.
- Jedes Blatt in T ist mit einem Label aus $\mathcal{Y}$ markiert.

Um das Label $y = h_T(x)$ zu finden, wird x „top-down“ durch T geroutet (beginnend an der Wurzel). Befindet sich x an einem inneren Knoten v, welcher die Eigenschaft E_v repräsentiert, dann wird x zum rechten Kind von v weitergeleitet, falls x die Eigenschaft E_v besitzt, und zum linken Kind von v, falls x die Eigenschaft E_v nicht besitzt. Irgendwann landet x bei diesem Routing-Prozess in einem Blatt von T. Dort lesen wir das betreffende Label y ab und setzen $h_T(x) = y$. Abbildung 1, dem Lehrbuch von Shalev-Shwartz und Ben-David entnommen, illustriert diese Vorgehensweise mit einem Entscheidungsbaum, welcher Papayas entweder als „lecker“ oder „nicht lecker“ klassifiziert.

Wenn Abfragen mit mehr als zwei möglichen Ausgängen verwendet werden, führt das auf völlig analoge Weise zu Entscheidungsbäumen, die i.A. nicht binär sind: ein innerer Knoten,

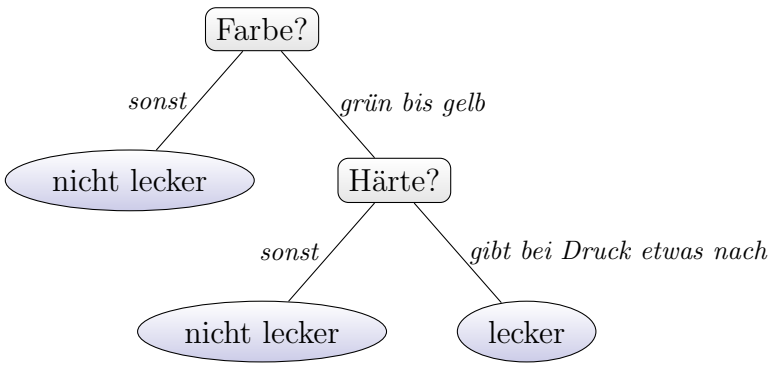

Abbildung 1: Entscheidungsbaum, der Papayas als „lecker“ bzw. „nicht lecker“ klassifiziert.

der eine Abfrage an x mit r möglichen Ausgängen repräsentiert, hat dann entsprechend r Kinder.

Abschnitt 18.1 ist den Booleschen Entscheidungsbäumen gewidmet. Die Informationskomplexität der Klasse der Booleschen Entscheidungsbäume lässt sich mit dem MDL-Ansatz[1] beherrschen (s. Abschnitt 18.1.1). Da das zugehörige Optimierungsproblem NP-hart ist, muss man sich beim Lernen von Booleschen Entscheidungsbäumen mit heuristischen Algorithmen (ohne mathematische Qualitätsgarantie) zufrieden geben. Wir besprechen hauptsächlich den ID3-Algorithmus von Quinlan (s. Abschnitt 18.1.2). In Abschnitt 18.2 besprechen wir einige Entscheidungsbaumvarianten, die über die einfachen Booleschen Entscheidungsbäume hinausgehen.

18.1 Boolesche Entscheidungsbäume

Definition 18.1 *Ein* Boolescher Entscheidungsbaum *über dem Grundbereich* $\{0,1\}^d$ *ist ein Binärbaum, bei welchem jeder innere Knoten mit einem Index aus* $[d]$ *und jedes Blatt mit einem Bit aus* $\{0,1\}$ *markiert ist.* $1-\mathrm{DT}_d$ *bezeichne die Menge aller Booleschen Entscheidungsbäume über* $\{0,1\}^d$.

Wir interpretieren das Label $i \in [d]$ eines inneren Knotens v als die Abfrage „$x_i = 1$?“. Auf diese Weise repräsentiert jeder Baum $T \in 1-\mathrm{DT}_d$ eine Boolesche Funktion $h_T : \{0,1\}^d \to \{0,1\}$.

18.1.1 MDL-basierte Analyse der Informationskomplexität

Bemerkung 18.2 *Jede Boolesche Funktion* $h : \{0,1\}^d \to \{0,1\}$ *ist durch einen Baum aus* $1-\mathrm{DT}_d$ *repräsentierbar, so dass* $\{0,1\}^d$ *durch* $1-\mathrm{DT}$ *aufspaltbar ist und es gilt*

$$\mathrm{VCdim}(1-\mathrm{DT}_d) = 2^d \ .$$

Beweis Der vollständige binäre Baum T_d der Tiefe d, bei welchem jeder innere Knoten einer Tiefe $i \in \{0,1,\ldots,d-1\}$ mit $i+1$ markiert ist, verteilt die 2^d Instanzen aus $\{0,1\}^d$ bijektiv auf seine 2^d Blätter. Da wir an den Blättern ein beliebiges Binärmuster aus $\{0,1\}^{2^d}$ anlegen können, kann jede Boolesche Funktion $h : \{0,1\}^d \to \{0,1\}$ mit Hilfe von T und einem entsprechenden Binärmuster an den 2^d Blättern dargestellt werden. **qed.**

Effiziente uniforme PAC-Lernbarkeit von $1-\mathrm{DT}$ ist damit ausgeschlossen. Wir weichen auf das Modell des nicht-uniformen Lernens aus. Die Klasse $1-\mathrm{DT} := \cup_{d\geq 1} 1-\mathrm{DT}_d$ ist zwar unendlich aber abzählbar. Aus dem Kapitel über nichtuniformes Lernen wissen wir folgendes. Wenn wir die Hypothesen h einer abzählbar unendlichen Klasse mit den Strings eines binären Präfixcodes darstellen und wenn $|h|$ die Länge des Codewortes für h bezeichnet, dann gilt (mit Wahrscheinlichkeit mindestens $1-\delta$ über $S \sim \mathcal{D}^m$) die folgende Fehlerschranke:

$$\forall h \in \mathcal{H} : L_{\mathcal{D}}(h) \leq L_S(h) + \sqrt{\frac{|h| + \ln(2/\delta)}{2m}} \ . \qquad (1)$$

[1] MDL = Minimum Description Length

Folgende Rekursion liefert einen Präfixcode über dem Alphabet $\{0, 1, (,)\}$ für die Binärbäume aus $1-$DT:

1. Der nur aus einer mit $b \in \{0, 1\}$ markierten Wurzel bestehende Baum wird kodiert mit dem String (b).

2. Wenn T einen Baum mit linkem Unterbaum T_0 und rechtem Unterbaum T_1 bezeichnet, dessen Wurzel mit i markiert ist, dann ergibt sich das Codewort von T aus der Rekursionsgleichung $K(T) = (\text{bin}(i)K(T_0)K(T_1))$. Hierbei steht $\text{bin}(\cdot)$ für die Binärdarstellung einer natürlichen Zahl.

Wir benötigen für jeden inneren Knoten $2 + \lceil \log(d) \rceil$ und für jedes Blatt 3 Zeichen. Durch Übergang zu einem binären Präfixcode verdoppelt sich die jeweilige Zeichenanzahl. Daher hat ein Baum aus $1-\text{DT}_d$ mit n Knoten (und daher $(n-1)/2$ inneren Knoten und $(n+1)/2$ Blättern) die Kodierungslänge

$$\frac{n-1}{2}(4+2\lceil \log(d) \rceil)+6\frac{n+1}{2} = 2(n-1)+(n-1)\lceil \log(d) \rceil+3(n+1) = 5n+1+(n-1)\lceil \log(d) \rceil \ .$$

Wenn wir dies für $|h|$ in (1) einsetzen, erhalten wir für alle Wahlen von d, n und alle Bäume T aus $1-\text{DT}_d$ mit n Knoten die Fehlerschranke

$$L_{\mathcal{D}}(T) \leq L_S(T) + \sqrt{\frac{5n + 1 + (n-1)\lceil \log(d) \rceil + \ln(2/\delta)}{2m}} \ . \tag{2}$$

Bei MDL-basiertem nichtuniformen Lernen von $1-$DT würden wir zu gegebenem S einen Baum aus $1-$DT auswählen, der diese Fehlerschranke minimiert. Wie gewohnt dient der Kostenterm $L_S(T)$ der Vermeidung von „underfitting“; der Term

$$\sqrt{\frac{5n + 1 + (n-1)\lceil \log(d) \rceil + \ln(2/\delta)}{2m}}$$

bestraft unnötig komplexe Bäume und dient der Vermeidung von „overfittimg“.

18.1.2 Quinlan's ID3-Algorithmus

Das Problem, einen Baum aus $1-$DT zu bestimmen, der die obere Schranke für $L_{\mathcal{D}}(T)$ aus (2) minimiert, ist NP-hart. Es gibt aber eine Reihe von heuristischen Verfahren (ohne mathematische Qualitätsgarantie). Wir besprechen im Folgenden den ID3-Algorithmus von Quinlan.[2] Er benutzt eine Funktion $\text{Gain}(S, i)$, die misst, welchen „Fortschritt“ wir machen, wenn wir die Trainingsmenge S gemäß der Abfrage „$x_i = 1$?“ zerlegen in $\{(x, y) \in S |\ x_i = 0\}$ und $\{(x, y) \in S |\ x_i = 1\}$. ID3 ist ein gieriger Algorithmus zur Konstruktion von T, der den aktuell betrachteten inneren Knoten immer so labelt, dass der betreffende Gain-Wert maximiert wird. Wenn am aktuell betrachteten Knoten v keine Aufteilung von S in zwei echte

[2] „ID3“ steht für „Iterative Dichotomizer 3“.

Untermengen möglich ist, weil alle in S befindlichen Instanzen das selbe Label besitzen, dann erklärt ID3 den Knoten v zu einem Blatt von T und markiert dieses mit dem betreffenden Label aus $\{0,1\}$. Es folgt der Pseudocode für ID3 in Form einer rekursiven Prozedur:
Eingabe: Trainingsmenge S und $A \subseteq [d]$
Mehode:

Fall 1: Alle Instanzen in S besitzen das Label 0.
Dann gib ein mit 0 markiertes Blatt aus.

Fall 2: Alle Instanzen in S besitzen das Label 1.
Dann gib ein mit 1 markiertes Blatt aus.

Fall 3: S besitzt Instanzen mit unterschiedlichen Labels.

1. Setze $j := \text{argmax}_{i \in A} \text{Gain}(S, i)$.
2. Für $b = 0, 1$ setze $T_b := \text{ID3}(\{(x, y) \in S |\ x_j = b\}, A \setminus \{j\})$.
3. Gib den Baum aus, der aus der mit j markierten Wurzel, dem linken Unterbaum T_0 und dem rechten Unterbaum T_1 besteht.

Im Hauptprogramm erfolgt der Aufruf $\text{ID3}(S, [d])$. Beachte, dass eine Menge S der Größe m höchstens $(m-1)$-mal echt aufgespalten werden kann. Die rekursive Prozedur ID3, die bei jedem Erreichen von Fall 3 einen inneren Knoten anlegt und eine Aufspaltung durchführt, wird also den Fall 3 höchstens $(m-1)$-mal erreichen. Daher wird die Prozedur Gain insgesamt höchstens $(m-1)d$-mal aufgerufen. Diese Aufrufe dominieren (asymptotisch gesehen) die gesamte Laufzeit.

Es sei $p(S)$ (bzw. $1 - p(S)$) der Bruchteil, der mit 1 (bzw. mit 0) gelabelten Instanzen in S. Je näher $p(S)$ dem Wert $1/2$ kommt, desto „unreiner“ ist S. Der Definition von $\text{Gain}(S, i)$ liegt meist ein Kostenmaß $C(p(S))$ für die Unreinheit von S zugrunde. Die folgenden Funktionen in der Rolle von $C : (0, 1) \to \mathbb{R}^+$ sind populär:

$$C_1(p) = \min\{p, 1-p\}\ ,\ C_2(p) = -p\log(p) - (1-p)\log(1-p) \ \text{ und } C_3(p) = 2p(1-p)\ .$$

$C_2(p)$ wird auch „binäre Entropie“ genannt. $C_2(p)$ und $C_3(p)$ sind konkave und differenzierbare obere Schranken von $C_1(p)$.

Die Funktion $\text{Gain}(S, i)$ misst, wie stark die Unreinheit von S im Mittel reduziert wird, wenn wir die Aufteilung in $S_{i,0} := \{(x, y) \in S | x_i = 0\}$ und $S_{i,1} := \{(x, y) \in S | x_i = 1\}$ vornehmen:

$$\text{Gain}(S, i) := C(p(S)) - (p \cdot C(p(S_{i,1})) + (1-p) \cdot C(p(S_{i,0})))\ .$$

Pruning. Einen Baum einer mit $m = |S|$ linear wachsenden Größe für eine Trainingsmenge S anzulegen ist so etwas wie der Garantieschein für „overfitting“. Es hat sich experimentell gezeigt, dass es besser ist, einen Baum T mit maximal $m-1$ inneren Knoten wieder auf ein kleineres Maß zurückzuschneiden (englisch: to prune) als ihn von vorneherein nur auf eine

kleinere Größe anwachsen zu lassen. Die betreffende Pruning-Prozedur bearbeitet T „bottom-up“ (in Richtung von den Blättern zur Wurzel). Bei jedem aktuell erreichten inneren Knoten v mit den Kindern v_0 und v_1 wird, der von v aufgespannte Unterbaum, notiert als $T(v)$, entweder unverändert gelassen oder ersetzt durch ein Blatt mit Label 0 oder 1, oder ersetzt durch einen der Bäume $T(v_0)$ und $T(v_1)$. Welche Option zum Zuge kommt hängt von einer Funktion $f(T, m)$ ab, die eine Schätzung bzw. eine obere Schranke für $L_{\mathcal{D}}(T)$ repräsentiert. Es folgt der Pseudocode für eine solche Pruning-Prozedur:

Eingabe: Baum T, Bewertungsfunktion $f(T, m)$

Methode: Durchlaufe T „bottom-up“. Bei jedem inneren Knoten v von T mit Kindern v_0 und v_1 wähle unter den folgenden Modifikationen von T eine aus, die zur besten Bewertung $f(T, m)$ führt:

- Ersetzung von $T(v)$ durch ein mit 0 markiertes Blatt
- Ersetzung von $T(v)$ durch ein mit 1 markiertes Blatt
- Ersetzung von $T(v)$ durch $T(v_0)$
- Ersetzung von $T(v)$ durch $T(v_1)$
- Übernahme eines unveränderten Baumes T

Ausgabe: Nach Abschluss des „bottom-up“ Durchlaufes gib die aktuelle Version von T aus.

18.2 Weitere Entscheidungsbaumvarianten

18.2.1 Die Klasse k-DT

Wir bezeichnen mit $k{-}\mathrm{DT}_d$ die Klasse, die aus $1{-}\mathrm{DT}_d$ dadurch hervorgeht, dass wir an jedem inneren Knoten eine Abfrage der Form „$M(x) = 1$?“ durchführen. Dabei ist M ein Boolesches Monom bestehend aus maximal k Literalen, d.h., $M = \ell_1 \wedge \ldots \wedge \ell_{k'}$ mit $k' \in [k]$, und $\ell_1, \ldots, \ell_{k'} \in \{x_1, \bar{x}_1, \ldots, x_d, \bar{x}_d\}$.

Da eine Abfrage der Form „$\bar{x}_i = 1$?“ die selbe Information liefert wie die Abfrage „$x_i = 1$?“, ist die Definition von $k{-}\mathrm{DT}$ im Spezialfall $k = 1$ deckungsgleich zu der Definition 18.1 von $1{-}\mathrm{DT}$.

Eine polynomielle L-Reduktion von $k{-}\mathrm{DT}$ nach $1{-}\mathrm{DT}$ ist leicht zu konstruieren (mit neuen Variablen, die 1–zu–1 den Booleschen Monomen der Maximallänge k entsprechen). Kombiniert man diese L-Reduktion mit ID3, erhält man einen (heuristischen) Lernalgorithmus für die Klasse $k{-}\mathrm{DT}$.

18.2.2 Binäre Abfragen zu reellwertigen Merkmalen

Wenn die Beispielinstanzen x aus $\mathbb{R}^d$ stammen, dann sind lineare Abfragen der Form „$\langle w, x\rangle \geq \theta$?“ gebräuchlich. Der Algorithmus von Quinlan kann an Anfragen dieser Art angepasst werden. In diesem Abschnitt diskutieren wir kurz spezielle lineare Abfragen von der Form „$x[i] \geq \theta$?“. Ein innerer Knoten kann dann einfach mit (i, θ) markiert werden. Es bezeichne

$1{-}\mathrm{DT}_d^{\mathbb{R}}$ die Klasse aller solcher Entscheidungsbäume über dem Grundbereich $\mathbb{R}^d$. Wir wollen zeigen, dass sich $(1{-}\mathrm{DT}_d^{\mathbb{R}})_{d\geq 1}$ polynomiell auf $(1{-}\mathrm{DT}_d)_{d\geq 1}$ reduzieren lässt. Die Reduktion, die wir verwenden werden, ist im strengen Sinn keine L-Reduktion, weil wir die Instanzen- und die Hypothesentransformationen von der zugrunde liegenden Trainingsmenge abhängig machen. Die Reduktion erfüllt aber den gleichen Zweck wie eine „anständige“ L-Reduktion. Insbesondere kann sie mit dem ID3-Algorithmus zu einem Lernalgorithmus für $(1{-}\mathrm{DT}_d^{\mathbb{R}})_{d\geq 1}$ kombiniert werden.

Trainingsmenge: $S = [(x_1, y_1), \ldots, (x_m, y_m)] \in (\mathbb{R}^d \times \{0,1\})^m$.

verwendete Schwellwerte: Für $i = 1, \ldots, m$ sei $x_1^{(i)}, \ldots, x_m^{(i)}$ eine Umordnung von $x_1, \ldots, x_m$, so dass $x_1^{(i)}[i] \leq \ldots \leq x_m^{(i)}[i]$. Wir unterstellen im Folgenden, dass die i-ten Koordinaten von $x_1, \ldots, x_m$ paarweise verschieden sind (aber unsere Ausführungen sind leicht auf den allgemeinen Fall erweiterbar). Für $i = 1, \ldots, m$ sei $\Theta_i := \{\theta_{i,j} |\ j = 0, \ldots, m\}$ mit

$$\theta_{i,0} < x_1^{(i)}[i] < \theta_{i,1} < \ldots < x_m^{(i)}[i] < \theta_{i,m} \ .$$

Kommentar: Zur Herstellung von ERM-Hypothesen auf S genügt es innere Knotenmarkierungen der Form (i, θ) mit $\theta \in \Theta_i$ zu verwenden.

Instanzentransformation: Ein Vektor $x \in \mathbb{R}^d$ wird abgebildet auf einen Vektor $x' \in \{0,1\}^{d'}$ für $d' = d(m+1)$. Für jedes Paar $(i,j) \in \{1, \ldots, d\} \times \{0, 1, \ldots, m\}$ setze $x'_{i,j} = \mathbb{1}_{[x_i \geq \theta_{i,j}]}$.

1. Hypothesentransformation: Ersetze einen Baum T aus der Klasse $1{-}\mathrm{DT}_d^{\mathbb{R}}$ durch einen Baum der selben Form aber mit veränderten Markierungen: die Markierung $(i, \theta_{i,j})$, welche die Abfrage „$x[i] \geq \theta_{i,j}$?“ repräsentiert, wird ersetzt durch die Markierung (i,j), welche die Abfrage „$x'_{i,j} = 1$?“ repräsentiert.

2. Hypothesentransformation: Wir benutzen einfach die umgekehrte Markierungssubstitution.

Beachte: wegen $x'_{i,j} = \mathbb{1}_{[x_i \geq \theta_{i,j}]}$ sind die Abfragen „$x[i] \geq \theta_{i,j}$?“ und „$x'_{i,j} = 1$?“ äquivalent zueinander!

Da $x \in \mathbb{R}^d$ auf $x' \in \{0,1\}^{d(m+1)}$ abgebildet wird, muss bei Anwendung von ID3 auf eine transformierte Trainingsmenge S' die Abbildung Gain insgesamt $(m-1)(m+1)d$-mal ausgewertet werden. Eine cleverere Implementierung kommt jedoch mit $O(dm\log(m))$ Auswertungen von Gain aus.

18.2.3 Breiman's Konzept der zufälligen Wälder

Ein Entscheidungswald ist eine Kollektion $W = (T_1, \ldots, T_r)$ von Entscheidungsbäumen. W repräsentiert eine Voraussagefunktion h_W, welche ein Majoritätsvotum über $T_1, \ldots, T_r$ durchführt, d.h., $h_W(x) = 1$ gilt genau dann, wenn mindestens $k/2$ der Werte $h_{T_1}(x), \ldots, h_{T_r}(x)$ gleich 1 sind. Breiman's Grundidee besteht darin, „overfitting“ zu vermeiden, indem ein

Baum T_i seine inneren Knoten nur mit Variablen einer Menge $A_i \subset [d]$ labeln darf. Dabei ist A_i eine zufällige Auswahl von $k << d$ Indizes aus der Grundmenge $[d]$ (und die Zufallsmengen $A_1, \ldots, A_r$ werden unabhängig voneinander ermittelt). Der Baum T_i könnte durch einen Aufruf ID3(S, A_i) bestimmt werden. Jeder einzelne Baum hat i.A. eine hohe Fehlerrate. Die Hoffnung ist, dass durch das Majoritätsvotum dennoch eine kleine Fehlerrate erzielt wird.