AGML, Sommersemester 2020 — Vorlesung 14 (5 Seiten)

Boosting

Thomas Kesselheim — Letzte Aktualisierung: 16. Juni 2020

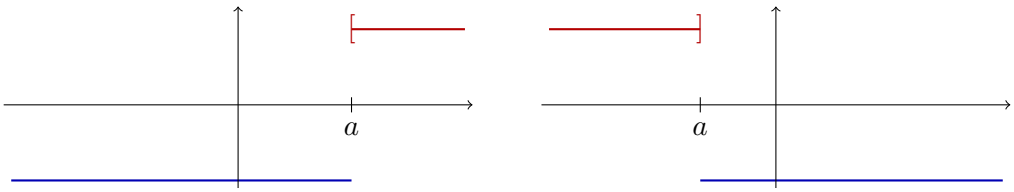
Beim maschinellen Lernen kommt es häufig vor, dass man in der Lage ist, Datenpunkte einigermaßen gut zu klassifizieren aber nicht genau genug. Ein Grund dafür kann sein, dass Hypothesen verwendet werden, die nicht ausdrucksstark genug sind, um Datenpunkte von einander zu unterscheiden. Heute werden wir eine sehr hilfreiche Technik kennenlernen, die die Genauigkeit auf der Trainingsmenge deutlich verbessert, das sogenannte *Boosting*.

1 Ein einführendes Beispiel

Schauen wir uns zunächst ein einfaches Beispiel an. Es mag sich zunächst etwas trivial anfühlen; es bedarf aber hoffentlich nicht zu viel Phantasie, um zu sehen, dass solche Probleme auch in komplexeren Szenarien auftreten.

Betrachten wir den Fall von binärer Klassifikation von Punkten aus den reellen Zahlen $\mathbb{R}$. Zu Beginn haben wir nur Schwellenwertfunktionen zur Verfügung. Dabei handelt es sich um Hypothesen der Form:

$$h(x) = \begin{cases} -1 & \text{falls } x < a \\ 1 & \text{falls } x \geq a \end{cases} \quad \text{oder} \quad h(x) = \begin{cases} -1 & \text{falls } x > a \\ 1 & \text{falls } x \leq a \end{cases} .$$



Wir können solche Hypothesen auch sehr knapp ausdrücken über zwei Parameter $w_1 \in \mathbb{R}$, $w_2 \in \{-1, 1\}$, sodass $h_{w_1,w_2}(x) = w_2 \cdot \text{sign}(x - w_1)$.[1]

Nehmen wir nun weiter an, dass die Grundwahrheit eigentlich ist, dass alle $x \in [a, b]$ positiv sind und alle $x \notin [a, b]$ negativ sind. Auch in diesem sehr einfachen Fall können wir im Allgemeinen keinen Trainingsfehler unter $\frac{1}{3}$ erreichen. Falls beispielsweise $S = \{(-1, -1), (0, +1), (1, -1)\}$ wird immer einer der Punkte inkorrekt klassifiziert werden.

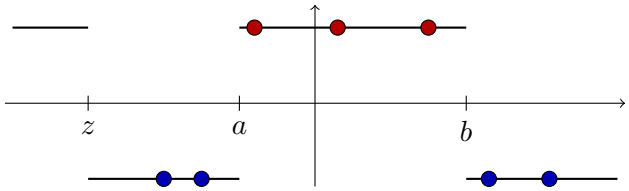
Hingegen wird eine *Linearkombination von Klassifikatoren* gute Ergebnisse liefern. Ist irgendeine Trainingsmenge $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ gegeben, definieren wir h^* mithilfe eines beliebigen $z < \min_i x_i$ über

$$h^*(x) = \text{sign}\left(h_{a,1}(x) + h_{b,-1}(x) + h_{z,-1}(x)\right) .$$

Nun ergibt sich für $x \in [a, b]$, dass $h_{a,1}(x) = h_{b,-1}(x) = 1$, $h_{z,-1}(x) = -1$. Also $h^*(x) = 1$. Für $x \in (z, a) \cup (b, \infty)$ ergibt sich $h^*(x) = -1$. Somit ist für alle i garantiert, dass $h^*(x_i) = y_i$ und damit $\text{err}_S(h^*) = 0$. Trotzdem handelt es sich nicht um die Grundwahrheit: Unterhalb von z ist die Klassifikation falsch.

Unser heutiges Ziel wird sein, diese Idee auf allgemeine Hypothesenklassen zu verallgemeinern.

[1] Wir nehmen an, dass die Signum-Funktion +1 an der Stelle 0 ist.



2 Problemstellung

Uns ist eine Trainingsmenge $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, $x_i \in X$, $y_i \in \{-1, +1\}$ für alle i, gegeben und wir möchten eine Hypothese h berechnen, sodass $\mathrm{err}_S(h) = \frac{1}{m}|\{i \mid h(x_i) \neq y_i\}|$ nah an 0 liegt.

Wir haben allerdings nur einen *schwachen Lernalgorithmus* zur Verfügung. Als Eingabe erhält er beliebige Gewichte $p_1, \ldots, p_m \geq 0$ mit $\sum_i p_i = 1$ (Die Gewichte definieren also eine Wahrscheinlichkeitsverteilung über S). Daraufhin berechnet er eine Hypothese h_p, sodass

$$\mathrm{err}_p(h_p) := \sum_{i:h_p(x_i)\neq y_i} p_i \leq \frac{1}{2} - \gamma$$

für ein festes $\gamma > 0$.[2] Für $p_i = \frac{1}{m}$ ist err_p genau der Trainingsfehler. Das heißt, die Hypothese h_p stellt sicher, dass eine gewichtete Version des Trainingsfehlers klein ist.

Beispiel 14.1. *Betrachten wir wieder das Eingangsbeispiel mit Schwellenwertfunktionen, aber die Grundwahrheit ist definiert über ein Intervall $[a, b]$. Wir werden nun zeigen, dass die Schwellenwertfunktion h, die $\mathrm{err}_p(h)$ minimiert, die obige Garantie mit $\gamma = \frac{1}{6}$ erfüllt. Das heißt, ein Algorithmus, der den gewichteten Trainingsfehler auf Schwellenwertfunktionen minimiert ist ein schwacher Lernalgorithmus, wenn die Grundwahrheit durch ein Intervall definiert ist.*

Um $\gamma = \frac{1}{6}$ zu zeigen, stellen wir fest, dass für jeden Vektor p und alle $a < b$ gilt

$$\sum_{i:x_i<a} p_i \leq \frac{1}{3} \quad \textit{oder} \quad \sum_{i:a\leq x_i\leq b} p_i \leq \frac{1}{3} \quad \textit{oder} \quad \sum_{i:x_i>b} p_i \leq \frac{1}{3} \ .$$

Zwei der drei Arten von Datenpunkten (unter a, zwischen a und b, über b) können wir leicht durch eine Schwellenwertfunktion richtig klassifizieren. Der kleinste Fehler wird also höchstens $\frac{1}{3} = \frac{1}{2} - \frac{1}{6}$ sein.

Wir werden zeigen, dass ein solcher schwacher Lernalgorithmus ausreicht, um einen *starken Lernalgorithmus* zu entwerfen: Gegeben ein $\epsilon > 0$ und eine Trainingsmenge S wird er den schwachen Lernalgorithmus nutzen, um Hypothesen $h_1, \ldots, h_T$ und $\alpha_1, \ldots, \alpha_T$ zu berechnen, sodass h^* mit $h^*(x) = \mathrm{sign}(\alpha_1 h_1(x) + \ldots + \alpha_T h_T(x))$ die Bedingungen $\mathrm{err}_S(h^*) \leq \epsilon$ erfüllt.

3 Idee für einen Algorithmus

Wir nehmen nun an, dass uns ein schwacher Lernalgorithmus gegeben ist, und wollen auf dieser Basis einen starken Lernalgorithmus entwickeln. Die Eingabe für diesen ist eine Trainingsmenge $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$.

Unser Algorithmus ruft also den schwachen Lernalgorithmus wiederholt auf der Trainingsmenge mit unterschiedlicher Gewichtung auf. Anfänglich setzen wir $p_i^{(1)} = \frac{1}{m}$ für alle i. Das

[2] Das Boosting-Framework kann erweitert werden um die Annahme, dass diese Schranke nur mit gewisser Wahrscheinlichkeit eingehalten wird.

heißt, der schwache Lernalgorithmus minimiert den Trainingsfehler so gut es ihm möglich ist. Wir erhalten eine Hypothese h_1. Wir wissen nun, dass h_1 falsch liegt auf höchstens $(\frac{1}{2} - \gamma)m$ Punkten in S. Anders formuliert ist h_1 auf etwas mehr als der Hälfte der Punkte in S korrekt.

Nun rufen wir den schwachen Lernalgorithmus erneut auf. Die Idee ist nun $p^{(2)}$ so zu setzen, dass $p_i^{(2)} > \frac{1}{m}$, falls $h_1(x_i) \neq y_i$, und $p_i^{(2)} < \frac{1}{m}$, falls $h_1(x_i) = y_i$. Das heißt, in dieser Ausführung soll der schwache Lernalgorithmus sich mehr auf diejenigen Punkte konzentrieren, die bislang falsch klassifiziert wurden.

Wir müssen nun angeben, wie sich die Gewichte $p_i^{(t)}$ ergeben. Wir nutzen dazu Gewichte $w_i^{(t)}$, die sich nicht zwangsläufig zu 1 aufsummieren. Es ist immer $p_i^{(t)} = w_i^{(t)}/W^{(t)}$, wobei $W^{(t)}$ die Summe aller $w_i^{(t)}$ ist.

Das Gewicht $w_i^{(t)}$ drückt aus, wie oft Datenpunkt i von den Hypothesen $h_1, \ldots, h_{t-1}$ falsch klassifiziert worden ist. Je größer es ist, desto mehr dieser Hypothesen lagen falsch. Entsprechend wichtiger ist es, dass der schwache Lernalgorithmus diesen Punkt richtig klassifiziert, wenn er h_t berechnet.

Konkret ist $w_i^{(1)} = 1$ für alle i. In Schritt t werden die Gewichte über die Regel

$$w_i^{(t+1)} = \begin{cases} \mathrm{e}^{-\eta_t} w_i^{(t)} & \text{falls } h_t(x_i) = y_i \\ \mathrm{e}^{\eta_t} w_i^{(t)} & \text{sonst} \end{cases}$$

angepasst. Dabei ist $\eta_t > 0$ für alle Punkte gleich und abhängig vom aktuellen Fehler. Für die spätere Rechnung können wir dies etwas knapper schreiben. Wegen

$$y_i h_t(x_i) = \begin{cases} +1 & \text{falls } h_t(x_i) = y_i \\ -1 & \text{sonst} \end{cases}$$

ergibt sich

$$w_i^{(t+1)} = w_i^{(t)} \mathrm{e}^{-\eta_t y_i h_t(x_i)} \quad .$$

Diese Art, Gewichte mittels multiplikativer Veränderung anzupassen, ist beim Entwurf von Algorithmen relativ verbreitet. Sie begegnet uns auch bei Algorithmen für ganz andere Probleme. Die Zusammenhänge können wir an dieser Stelle leider nicht diskutieren.

4 AdaBoost

Der Algorithmus *AdaBoost* (für Adaptive Boosting) benutzt genau diese Ideen. Er lautet wie folgt.

- Initialisiere $w_i^{(1)} = 1$ für alle i
- In Schritt $t = 1, \ldots, T$
  - Berechne $W^{(t)} = \sum_{i=1}^m w_i^{(t)}$, $p_i^{(t)} = w_i^{(t)}/W^{(t)}$
  - Sei h_t das vom schwachen Lernalgorithmus auf $p^{(t)}$ berechnete Ergebnis
  - Berechne $\epsilon_t = \sum_{i:h_t(x_i)\neq y_i} p_i^{(t)}$ (Fehler von h_t auf $p^{(t)}$)
  - Sei $\eta_t = \frac{1}{2} \ln\left(\frac{1}{\epsilon_t} - 1\right)$
  - Aktualisiere Gewichte $w_i^{(t+1)} = w_i^{(t)} \mathrm{e}^{-\eta_t y_i h_t(x_i)}$
- Gib h^* aus, definiert über $h^*(x) = \mathrm{sign}\left(\sum_{t=1}^T \eta_t h_t(x)\right)$

Satz 14.2. *Der Algorithmus AdaBoost garantiert* $\mathrm{err}_S(h^*) \leq \exp(-2\gamma^2 T)$.

Zur Erinnerung: γ stammt aus der Garantie des schwachen Lernalgorithmus. Interessanterweise braucht der Algorithmus es nicht zu kennen.

Beweis. Sei $g_t(x) = \sum_{t'=1}^{t} \eta_{t'} h_{t'}(x)$. Durch diese Definition sind $w_i^{(t)} = \mathrm{e}^{-y_i g_{t-1}(x_i)}$ and $h^*(x) = \mathrm{sign}(g_T(x))$.

Wir betrachten, wie sich die Summe der Gewichte, $W^{(t)} = \sum_{i=1}^{m} w_i^{(t)}$, über die Zeit verändert. Wir werden zeigen, dass gilt

$$W^{(t+1)} \leq \mathrm{e}^{-2\gamma^2} W^{(t)} \quad \text{für alle } t \in \{1, \ldots, T\} \ . \tag{1}$$

Betrachten wir zunächst, wie diese aus dieser Ungleichung die Aussage des Satzes folgt. Der Algorithmus läuft für T Schritte. Aufgrund von Ungleichung (1) gilt im Anschluss $W^{(T+1)} \leq \mathrm{e}^{-2\gamma^2 T} W^{(1)} = \mathrm{e}^{-2\gamma^2 T} m$.

Darüber hinaus gilt für alle i mit $h^*(x_i) \neq y_i$, dass $y_i g_T(x_i) \leq 0$, denn das Produkt zweier reeller Zahlen mit unterschiedlichem Vorzeichen ist immer nicht-positiv. Das bedeutet, dass für diese i auch $w_i^{(T+1)} = \mathrm{e}^{-y_i g_T(x_i)} \geq 1$. Für alle andere i nutzen wir, dass $w_i^{(T+1)} \geq 0$. So erhalten wir insgesamt $W^{(T+1)} \geq |\{i \mid h^*(x_i) \neq y_i\}|$ und damit

$$\mathrm{err}_S(h^*) \leq \frac{1}{m} W^{(T+1)} \leq \mathrm{e}^{-2\gamma^2 T} \ .$$

Die Aussage des Satzes ist damit bewiesen.

Damit müssen wir also nur noch Ungleichung (1) zeigen. Die Summe der Gewichte nach Schritt t ist

$$W^{(t+1)} = \sum_{i=1}^{m} w_i^{(t+1)} = \sum_{i=1}^{m} w_i^{(t)} \mathrm{e}^{-y_i \eta_t h_t(x_i)} \ .$$

Das heißt, die Änderung ist

$$\frac{W^{(t+1)}}{W^{(t)}} = \sum_{i=1}^{m} \frac{w_i^{(t)}}{W^{(t)}} \mathrm{e}^{-y_i \eta_t h_t(x_i)} = \sum_{i=1}^{m} p_i^{(t)} \mathrm{e}^{-y_i \eta_t h_t(x_i)} = \sum_{i: h_t(x_i) = y_i} p_i^{(t)} \mathrm{e}^{-\eta_t} + \sum_{i: h_t(x_i) \neq y_i} p_i^{(t)} \mathrm{e}^{\eta_t} \ .$$

Gemäß Definition $\sum_{i: h_t(x_i) \neq y_i} p_i^{(t)} = \epsilon_t$ und $\mathrm{e}^{\eta_t} = \sqrt{1/\epsilon_t - 1}$. Also

$$\begin{aligned} \frac{W^{(t+1)}}{W^{(t)}} &= (1-\epsilon_t)\mathrm{e}^{-\eta_t} + \epsilon_t \mathrm{e}^{\eta_t} = (1-\epsilon_t)\frac{1}{\sqrt{1/\epsilon_t - 1}} + \epsilon_t \sqrt{1/\epsilon_t - 1} \\ &= (1-\epsilon_t)\sqrt{\frac{\epsilon_t}{1-\epsilon_t}} + \epsilon_t \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} = 2\sqrt{\epsilon_t(1-\epsilon_t)} \ . \end{aligned}$$

Der schwache Lernalgorithmus garantiert uns, dass $\epsilon_t \leq \frac{1}{2} - \gamma$ und somit

$$\frac{W^{(t+1)}}{W^{(t)}} = 2\sqrt{\epsilon_t(1-\epsilon_t)} \leq 2\sqrt{\left(\frac{1}{2} - \gamma\right)\left(\frac{1}{2} + \gamma\right)} = \sqrt{1 - 4\gamma^2} \leq \sqrt{\mathrm{e}^{-4\gamma^2}} = \mathrm{e}^{-2\gamma^2} \ .$$

Dies zeigt Ungleichung (1) und damit ist die Aussage bewiesen. □

5 Die Schattenseiten

Wir haben nun hergeleitet, dass wir auf jeder Trainingsmenge S einen Trainingsfehler von höchstens $\exp(-2\gamma^2 T)$ erhalten werden, wenn wir T Iteration von AdaBoost verwenden. Es liegt nun nahe, T so groß wie möglich zu wählen, weil dadurch der Fehler kleiner und kleiner wird. Dieser kleinere Fehler kann jedoch durch Overfitting zustande kommen.

Formaler gesprochen: Die VC-Dimension der Klasse von Hypothesen, die AdaBoost in T Iterationen berechnen kann, wächst in T. Betrachten wir dazu $X = \mathbb{R}$ und ein beliebiges $m \in \mathbb{N}$. Wenn T im Verhältnis groß genug ist, dann kann Boosting von Schwellenwertfunktionen m Punkte mit beliebigen Labels versehen. Das heißt, die VC-Dimension ist mindestens m in diesem Fall. Hierzu stellen wir fest, dass $\gamma = \frac{1}{2m}$ gilt, wenn wir das Label eines Punktes richtig setzen und bei mindestens der Hälfte der übrigen Punkte. Mit $T \geq \frac{1}{2\gamma^2} \ln(2m)$ erhalten wir $\mathrm{err}_S(h^*) \leq \frac{1}{2m}$. Also muss $\mathrm{err}_S(h^*) = 0$ sein.

Insgesamt heißt dies, dass man vorsichtig sein muss, wenn man Boosting anwendet: Es ist ein hilfreiches Werkzeug, um besser klassifizieren zu können, aber es gibt den bekannte Abwägung zwischen Trainingsfehler und Overfitting.

Referenzen

- Understanding Machine Learning, Kapitel 10
- Foundations of Machine Learning, Kapitel 7
- Freund, Yoav; Schapire, Robert E (1997). A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences. 55: 119. (Original AdaBoost-Paper)