Theorie des maschinellen Lernens

Hans U. Simon

20. Juni 2017

10 Boosting

In diesem Kapitel gehen wir der Frage nach: „Kann man aus den Ratschlägen eines Narren Weisheit schöpfen?“ Die Rolle des „Narren“ übernimmt ein Algorithmus, den wir als „schwachen Lerner“ bezeichnen. Seine Voraussagen sind nur etwas besser als zufälliges Raten, d.h., er sagt ein binäres Label mit einer etwas unter 1/2 liegenden Fehlerrate voraus. Den Part der „Weisheit“ übernimmt eine Voraussagefunktion, die ein gewichtetes Majoritätsvotum über eine Kollektion von schwachen Hypothesen durchführt. Kern dieses Kapitels ist eine auf Freund und Schapire zurückgehende Methode namens AdaBoost, welche einen schwachen Lerner in einen (starken) PAC-Lerner (mit einer Fehlerrate nahe bei 0) transformiert.

In Abschnitt 10.1 beschäftigen wir uns mit dem Konzept eines „schwachen Lerners“. Abschnitt 10.2 liefert eine obere Schranke für die VC-Dimension der Klasse der Linearkombinationen von Basishypothesen. Dies wird uns später helfen, den Schätzfehler von AdaBoost zu kontrollieren. Der das Kapitel abschließende Abschnitt 10.3 ist AdaBoost und seiner Analyse gewidmet.

10.1 Schwache Lernbarkeit

Definition 10.1 *Es sei $0 < \gamma < 1/2$. $\mathcal{H}$ und $\mathcal{B}$ seien Hypothesenklassen mit Grundbereich $\mathcal{X}$.*

- *Wir sagen ein Lernalgorithmus A* lernt *$\mathcal{H}$ γ-schwach mit Hypothesen aus $\mathcal{B}$, wenn eine Funktion $m_{\mathcal{H}} : (0,1) \to \mathbb{N}$ existiert, so dass für jedes $\delta \in (0,1)$, jede Verteilung P über $\mathcal{X} \times \{0,1\}$, welche bezüglich $\mathcal{H}$ die Realisierbarkeitsannahme erfüllt, folgendes gilt. Wenn A auf einer Trainingsmenge $S \sim P^m$ mit $m \geq m_{\mathcal{H}}(\delta)$ gestartet wird, dann ist mit einer Erfolgswahrscheinlichkeit von mindestens $1 - \delta$ die Ausgabe von A eine Hypothese $h \in \mathcal{B}$ mit $L_P(h) \leq 1/2 - \gamma$.*

- *$\mathcal{H}$ heißt* γ-schwach lernbar *mit Hypothesen aus $\mathcal{B}$, wenn (unter der Realisierbarkeitsannahme) ein entsprechender γ-schwacher Lerner existiert.*

Beispiel 10.2 *Es sei $\mathcal{X} = \mathbb{R}$. Weiter sei $\mathcal{G}_T$ die Klasse aller Funktionen von $\mathbb{R}$ nach $\{-1,1\}$ mit höchstens T Vorzeichenwechseln, wenn wir die reelle Zahlengerade von links nach rechts*

durchlaufen (= Klasse der ± 1-wertigen $(T+1)$-stückweise konstanten Funktionen). Zum Beispiel ist jede Funktion aus $\mathcal{G}_2$ gegeben durch eine Zerlegung der reellen Zahlengeraden in drei (oder weniger) Intervalle I_1, I_2, I_3, geordnet von links nach rechts, auf denen entweder das Binärmuster $1, -1, 1$ oder das Binärmuster $-1, 1, -1$ angenommen wird. Eine Schwellenfunktion (mit Vorzeichen) *ist eine Funktion der Form $x \mapsto b \cdot \operatorname{sign}(x - \theta)$ mit $\theta \in \mathbb{R}$ und $b \in \{-1, 1\}$. Sie ordnet x den Wert b zu, falls $x \geq \theta$ und den Wert $-b$ andernfalls. Es sei DS_1 die Klasse aller Funktionen dieser Form. Wir zeigen, dass ein ERM-Algorithmus für die Klasse DS_1 zu einem $1/12$-schwachen Lerner für die Klasse $\mathcal{G}_2$ aufgebaut werden kann. Es sei S eine Trainingsmenge, die oBdA so gelabeled ist, dass Punkte auf den Intervallen I_1, I_2, I_3 jeweils die Werte $-1, 1, -1$ annehmen. (Der Fall mit vertauschten ± 1-Labels oder mit weniger als drei Intervallen ist analog zu behandeln.) Es ist leicht zu sehen, dass es zu jedem Intervall $I \in \{I_1, I_2, I_3\}$ eine Hypothese aus DS_1 gibt, die nur die Trainingspunkte in I falsch labeled. Daher gilt für eine ERM-Hypothese $h_S \in \mathrm{DS}_1$: $L_S(h_S) \leq 1/3 = 1/2 - 1/6$. Die VC-Dimension von DS_1 ist gleich 2 (einfach zu sehen). Daher hat DS_1 die „Uniform Convergence Property (UCP)". Es sei P die auf $\mathbb{R}$ zugrunde liegende Verteilung. Wir können die Größe der Trainingsmenge S so bemessen, dass mit einer Wahrscheinlichkeit von mindestens $1 - \delta$ die absolute Differenz von $L_P(h)$ und $L_S(h)$ für alle $h \in \mathrm{DS}_1$ durch $1/12$ beschränkt ist. Dann gilt: $L_P(h_S) \leq 1/2 - 1/6 + 1/12 = 1/2 - 1/12$.*

Die Funktionen der Klasse DS_1 werden auch *Entscheidungsstümpfe (decision stumps)* über $\mathbb{R}$ genannt (hauptsächlich weil sie als rudimentäre Entscheidungsbäume gesehen werden können).
Es sei

$$S = [(x_1, y_1), \ldots, (x_m, y_m)] \in (\mathbb{R} \times (\mathbb{R} \setminus \{0\}))^m \quad \text{mit} \quad x_1 < x_2 < \ldots < x_m \tag{1}$$

eine „sortierte gewichtete Trainingssequenz". Die Gewichte können wahlweise Mehrfachvorkommen einer Instanz oder auch Wahrscheinlichkeiten repräsentieren. Zum Beispiel könnte $(x_i, -3)$ das 3-fache Vorkommen einer mit -1 markierten Instanz x_i repräsentieren.

Wir wollen im Folgenden überlegen, welcher Entscheidungsstumpf auf S die kleinste (gewichtete) Anzahl von Fehlern macht. Es ist leicht zu sehen, dass wir nur Hypothesen aus DS_1 mit $\theta \in \{x_1, \ldots, x_m\}$ in Betracht ziehen müssen. Für $j = 1, \ldots, m$ sei

$$W^+(j) = \sum_{i \in [j]: y_i > 0} y_i \quad \text{und} \quad W^-(j) = \sum_{i \in [j]: y_i < 0} |y_j| \ .$$

Weiter sei $W^+ = W^+(m)$ bzw. $W^- = W^-(m)$ die Summe aller positiven Gewichte bzw. aller Absolutbeträge von negativen Gewichten. Die folgende Funktion wird sich als nützlich erweisen:

$$F(j) = W^+(j) - W^-(j) = \sum_{i \in [j]} y_i \ ,$$

d.h., $F(j)$ ist die Summe der ersten j Gewichte, wobei $F(0) = 0$. Es bezeichne DS_1^+ (bzw. DS_1^-) die Menge der Hypothesen aus DS_1 mit $b = 1$ (bzw. $b = -1$).

Lemma 10.3 *Es sei $j \in [m]$. Dann gilt folgendes. Die Hypothese $h_j^+(x) = \text{sign}(x - x_j)$ macht auf S insgesamt $W^- + F(j-1)$ Fehler und die Hypothese $h_j^-(x) = -\text{sign}(x - x_j)$ macht auf S insgesamt $W^+ - F(j-1)$ Fehler.*

Beweis Wir beweisen die erste Aussage. (Der Beweis für die zweite Aussage ist analog zu führen.) Die Hypothese h_j^+ macht $W^+(j-1)$ Fehler auf $(x_1, y_1), \ldots, (x_{j-1}, y_{j-1})$ und $W^- - W^-(j-1)$ Fehler auf $(x_j, y_j), \ldots, (x_m, y_m)$. Die gesamte Fehlerzahl von h_j^+ beträgt demnach $W^+(j-1) + (W^- - W^-(j-1)) = W^- + F(j-1)$, wie behauptet. **qed.**

Dies führt zu dem folgenden Algorithmus A_{DS}, der die ERM-Lernregel für die Klasse DS_1 implementiert:

1. Transformiere eine Trainingssequenz in die zugehörige sortierte gewichtete Trainingssequenz S der Form (1).
2. Berechne die Größen W^+, W^- und $F(j)$ für $j = 0, 1, \ldots, m$.
3. Bestimme $j^+ = \text{argmin}_{j \in [m]} F(j-1)$ und $j^- = \text{argmax}_{j \in [m]} F(j-1)$.
4. Falls $W^- + F(j^+ - 1) \leq W^+ - F(j^- - 1)$, dann gib die Hypothese $\text{sign}(x - x_{j^+})$ aus, andernfalls die Hypothese $-\text{sign}(x - x_{j^-})$.

Die Laufzeit für den ersten Schritt wird vom Sortieren dominiert und beträgt somit $O(|S| \log |S|)$. Die weiteren Schritte lassen sich in Linearzeit ausführen. Wir fassen diese Diskussionen in folgendem Resultat zusammen:

Lemma 10.4 *Die ERM-Lernregel für die Hypothesenklasse DS_1 ist mit Laufzeitschranke $O(|S| \log |S|)$ implementierbar (bzw. sogar Linearzeit, falls die Trainingssequenz bereits sortiert gegeben ist).*

Wir kommen nun zu einer d-dimensionalen Verallgemeinerung der Klasse DS_1. Die *Klasse der Entscheidungsstümpfe* über $\mathbb{R}^d$ ist gegeben durch

$$\text{DS}_d = \{x \mapsto b \cdot \text{sign}(x_i - \theta) : b \in \{-1, 1\}, i \in [d], \theta \in \mathbb{R}\} \ .$$

Wir können also eine Hypothese aus DS_d wählen, indem wir uns für eine Dimension $i \in [d]$ entscheiden und dann einen 1-dimensionalen Entscheidungsstumpf auswählen, welcher auf x_i angewendet wird.

Der obige Algorithmus zur Implementierung der ERM-Lernregel für die Klasse DS_1 lässt sich auf die offensichtliche Weise benutzen, um die ERM-Lernregel auf DS_d zu implementieren:

- Wende A_{DS} d-mal an (eine Anwendung pro Dimension) und erhalte die jeweils besten Hypothesen, sagen wir $h_1, \ldots, h_d$ sowie die Anzahl der Fehler, die sie jeweils auf der Trainingssequenz machen.

- Wähle unter $h_1, \ldots, h_d$ die Hypothese mit der kleinsten Fehleranzahl aus.

Diese Vorgehensweise führt zu folgendem Ergebnis:

Theorem 10.5 *Die ERM-Lernregel für die Hypothesenklasse* DS_d *ist mit Laufzeitschranke* $O(d \cdot |S| \log |S|)$ *implementierbar.*

Wie man sieht wird die Laufzeit dominiert durch das d-malige Sortieren der Trainingssequenz (eine Sortierung für jede der d Dimensionen).

10.2 Linearkombination von Basishypothesen

Wir werden sehen, dass geeignete Linearkombinationen von Hypothesen, welche von einem schwachen Lerner produziert wurden, eine sehr kleine empirische Fehlerrate auf den Trainingsdaten haben. Mit anderen Worten: der Approximationsfehler wird sich als kontrollierbar erweisen. Dies wirft die Frage nach der Kontrollierbarkeit des Schätzfehlers und somit die Frage nach der VC-Dimension auf.

Es sei nun $\mathcal{B}$ eine Klasse von reellwertigen Basishypothesen über einem Grundbereich $\mathcal{X}$ und $T \in \mathbb{N}$. Wir definieren

$$\mathrm{LIN}(\mathcal{B}, T) = \left\{ x \mapsto \operatorname{sign}\left(\sum_{t=1}^{T} w_t h_t(x) \right) : \; w \in \mathbb{R}^T, h_1, \ldots, h_T \in \mathcal{B} \right\} \; .$$

Der Einfachheit halber nennen wir die Elemente von $\mathrm{LIN}(\mathcal{B}, T)$ „Linearkombinationen“ von T Basisfunktionen aus $\mathcal{B}$ (obwohl es sich, streng genommen, um die sign-Funktion angewendet auf eine solche Linearkombination handelt).

Wie mächtig kann die Klasse $\mathrm{LIN}(\mathcal{B}, T)$ im Vergleich zu $\mathcal{B}$ werden? Wir betrachten zunächst ein

Beispiel 10.6 *Es sei* $\mathcal{G}_T$ *die bereits in Beispiel 10.2 eingeführte Hypothesenklasse. Offensichtlich gilt* $\mathrm{VCdim}(\mathcal{G}_T) = T + 1$. *Es ist nicht schwer zu zeigen, dass* $\mathcal{G}_T$ *eine Teilklasse von* $\mathrm{LIN}(\mathrm{DS}_1, T)$ *ist. S. Übung. Es folgt, dass* $\mathrm{VCdim}(\mathrm{LIN}(\mathrm{DS}_1, T)) \geq \mathrm{VCdim}(\mathcal{G}_T) = T + 1$, *wohingegen* $\mathrm{VCdim}(\mathrm{DS}_1) = 2$.

Allgemeinere Einsichten ergeben sich aus folgendem

Theorem 10.7 *Es sei* $\mathcal{B}$ *eine Klasse von Basishypothesen der VC-Dimension* $d \geq 3$ *und* $3 \leq T \in \mathbb{N}$. *Dann gilt:*

$$\mathrm{VCdim}(\mathrm{LIN}(\mathcal{B}, T)) \leq \underbrace{T(d+1)(2 + 3\log(d+1))}_{=D(T,d)} \; .$$

Beweis Es seien $x_1, \ldots, x_m \in \mathcal{X}$. Wir sagen eine Matrix $B \in \{-1,1\}^{T \times m}$ ist *realisierbar* durch $\mathcal{B}$, falls $h_1, \ldots, h_T \in \mathcal{B}$ existieren mit $h_t(x_i) = B[t,i]$ für alle $t \in [T]$ und für alle $i \in [m]$. Mit Sauer's Lemma folgt, dass in jeder Zeile $[h_t(x_1), \ldots, h_t(x_m)]$ von B höchstens $(em/d)^d$ Binärmuster durch $\mathcal{B}$ realisierbar sind. Somit sind höchstens $(em/d)^{dT}$ verschiedene Matrizen $B \in \{-1,1\}^{T \times m}$ durch $\mathcal{B}$ realisierbar. Die Realisierung eines Binärmusters auf $x_1, \ldots, x_m$ durch die Klasse $\mathrm{LIN}(\mathcal{B}, T)$ können wir uns zweistufig vorstellen wie folgt:

1. Wir wählen $h_1, \ldots, h_T$ und bilden x_i auf den Spaltenvektor $B_i = (h_1(x_i), \ldots, h_T(x_i))^\top$ ab (für alle $i \in [m]$).

2. Zu jeder festen (durch $\mathcal{B}$ realisierbaren) Wahl von $B = [B_1 \ldots B_m]$ wählen wir einen Gewichtsvektor $w \in \mathbb{R}^T$ und bilden B_i auf $\mathrm{sign}(\langle w, B_i \rangle)$ ab (für alle $i \in [m]$).

Wie bereits oben festgestellt gibt es höchstens $(em/d)^{dT}$ mögliche Wahlen von B. Da die VC-Dimension von $\mathrm{sign} \circ \mathrm{LIN}_T^0$ gleich T ist, gibt es für jede feste Wahl von B höchstens $(em/T)^T$ Binärmuster, die auf $B_1, \ldots, B_m$ durch Wahl eines Gewichtsvektors w realisierbar sind. Insgesamt sind in diesem zweistufigen Prozess also maximal

$$(em/d)^{dT} \cdot (em/T)^T < m^{(d+1)T}$$

Binärmuster realisierbar. Falls $x_1, \ldots, x_m$ durch $\mathrm{LIN}(\mathcal{B}, T)$ aufspaltbar wären, dann müsste

$$2^m \leq m^{(d+1)T} \tag{2}$$

gelten. Dies ist für „große“ Werte von m sicherlich falsch, da die linke Seite von (2) exponentiell, die rechte jedoch nur polynomiell, in Abhängigkeit von m wächst. Eine genauere Rechnung (s. Lemma A.1 im Appendix A des Lehrbuches) zeigt, dass (2) für alle $m > T(d+1)(2+3\log(d+1))$ falsch ist. Daher muss die VC-Dimension von $\mathrm{LIN}(\mathcal{B}, T)$ durch $T(d+1)(2+3\log(d+1))$ nach oben beschränkt sein. **qed.**

Wir merken kurz an, dass wir die Voraussetzung $T, d \geq 3$ in Theorem 10.7 eliminieren können, wenn wir in der oberen Schranke des Theorems T bzw. d durch $\max\{3, T\}$ bzw. $\max\{3, d\}$ ersetzen.

10.3 AdaBoost

„AdaBoost“ steht für „Adaptive Boosting“ und bezeichnet ein von Freund und Schapire entworfenes Verfahren, einen schwachen Lerner WL für eine Hypothesenklasse $\mathcal{H}$ in einen (starken) PAC-Lerner für $\mathcal{H}$ zu transformieren. Falls WL Hypothesen aus einer Basisklasse $\mathcal{B}$ verwendet, dann verwendet AdaBoost Hypothesen aus $\cup_{T \geq 1} \mathrm{LIN}(\mathcal{B}, T)$. AdaBoost wird WL als Hilfsprozedur einsetzen. Ein Aufruf von WL hat die allgemeine Form $\mathrm{WL}(P, S, \delta')$. Hierbei bezeichnet $S \in (\mathcal{X} \times \{-1,1\})^m$ eine Trainingssequenz, P eine Verteilung auf $[m]$ (und damit auch auf $S_\mathcal{X}$) und $0 < \delta' < 1$ den Zuverlässigkeitsparameter. Die Parameter P, S repräsentieren eine auf $S_\mathcal{X}$ konzentrierte Verteilung auf $\mathcal{X}$ und WL wird mit einer Erfolgswahrscheinlichkeit von mindestens $1 - \delta'$ eine Hypothese $h \in \mathcal{B}$ mit $L_P(h) \leq 1/2 - \gamma$ abliefern.

Bemerkung 10.8 *Da wir* WL *über die Parameter P, S die Verteilung auf $\mathcal{X}$ mitteilen, kann (eine entsprechende Erweiterung von)* WL *sich die von ihm benötigte Trainigssequenz selber generieren. Die markierten Beipiele werden dabei P-zufällig aus S gezogen.*
Falls WL *die ERM-Lernregel für $\mathcal{B}$ auf* gewichteten *Trainingssequenzen implementiert (wie es zum Beispiel für $\mathcal{B} = \mathrm{DS}_d$ auf effiziente Weise möglich ist) so vereinfacht sich die Vorgehensweise: die Prozedur* WL *kann dann unmittelbar auf die mit P gewichtete Trainingssequenz S angewendet werden.*

AdaBoost wird rundenweise vorgehen und in jeder Runde t folgende Objekte generieren:

- eine Basishypothese h_t (mit Hilfe von WL)
- die Fehlerrate ε_t von h_t bezüglich der in dieser Runde gültigen Verteilung $D^{(t)}$ auf S
- einen Gewichtsparameter w_t
- die für die nächste Runde gültige Verteilung $D^{(t+1)}$

Am Ende wird eine Art gewichtetes Majoritätvotum der Basishypothesen h_t ausgegeben.

Der Schlüssel zum Verständnis von AdaBoost liegt in der Wahl der Verteilungen $D^{(t)}$. Sie legen ein vergleichsweise hohes Gewicht auf Beispiele, die bei einem gewichteten Majoritätsvotum (zum gegenwärtigen Zeitpunkt) falsch behandelt würden. Dies zwingt den schwachen Lerner, den „problematischen“ Beispielen verstärkte Aufmerksamkeit zu schenken.
Es folgt eine kompakte Beschreibung des Algorithmus' AdaBoost.

Eingabedaten: eine zufällige und zu einer Hypothese aus $\mathcal{H}$ konsistente Trainingssequenz $S = [(x_1, y_1), \ldots, (x_m, y_m)] \in (\mathcal{X} \times \{-1, 1\})^m$, ein Parameter $T \in \mathbb{N}$ für die Anzahl der Runden sowie ein Zuverlässigkeitsparameter δ

Hilfsprozedur: ein γ-schwacher Lerner WL für $\mathcal{H}$, welcher Hypothesen aus der Basisklasse $\mathcal{B}$ produziert

Initialisierung: $D^{(1)} = (1/m, \ldots, 1/m)$.

Hauptschleife: Für $t = 1, \ldots, T$ mache folgendes:

$$h_t = \mathrm{WL}\left(D^{(t)}, S, \frac{\delta}{2T}\right) \ , \tag{3}$$

$$\varepsilon_t = \sum_{i=1}^{m} D_i^{(t)} \mathbb{1}_{[h_t(x_i) \neq y_i]} \ , \tag{4}$$

$$w_t = \frac{1}{2} \ln\left(\frac{1}{\varepsilon_t} - 1\right) \ , \tag{5}$$

$$D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-w_t y_i h_t(x_i))}{\sum_{j=1}^{m} D_j^{(t)} \exp(-w_t y_j h_t(x_j))} \text{ für } i = 1, \ldots, m \tag{6}$$

Ausgabe: $h_S(x) = \text{sign}\left(\sum_{t=1}^{T} w_t h_t(x)\right)$.

Der folgende Satz beschreibt, wie der Approximationsfehler von AdaBoost kontrolliert werden kann.

Theorem 10.9 *Mit den im Algorithmus AdaBoost verwendeten Notationen gilt folgendes. Falls* WL *in jeder Runde* $t \in [T]$ *eine Hypothese* h_t *mit*

$$\varepsilon_t \leq 1/2 - \gamma \tag{7}$$

abliefert, dann gilt für die von AdaBoost ausgegebene Hypothese

$$L_S(h_S) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}_{[h_S(x_i) \neq y_i]} \leq e^{-2\gamma^2 T} \ .$$

Beweis Für $t = 0, 1 \ldots, T$ definieren wir

$$f_t = \sum_{p=1}^{t} w_p h_p \ \text{ und } Z_t = \frac{1}{m} \sum_{i=1}^{m} \exp(-y_i f_t(x_i)) \ . \tag{8}$$

Es gilt dann $f_0 = 0$ und $Z_0 = 1$. Expandieren der rekursiven Gleichung (6) für $D_i^{(t+1)}$ liefert

$$D_i^{(t+1)} = \frac{e^{-y_i f_t(x_i)}}{\sum_{j=1}^{m} e^{-y_j f_t(x_j)}} \ . \tag{9}$$

Die von AdaBoost ausgegebene Hypothese h_S hat die Form $H_S = \text{sign} \circ f_T$. Für eine beliebige reellwertige Funktion f und $(x, y) \in \mathcal{X} \times \{-1, 1\}$ gilt $\mathbb{1}_{[\text{sign}(f(x)) \neq y]} \leq e^{-yf(x)}$. Folglich ist Z_T eine obere Schranke für $L_S(h_S) = L_S(\text{sign} \circ f_T)$. Es genügt daher $Z_T \leq e^{-2\gamma^2 T}$ nachzuweisen. Wegen $Z_0 = 1$ kann Z_T als Teleskopprodukt geschrieben werden:

$$Z_T = \frac{Z_T}{Z_0} = \prod_{t=0}^{T-1} \frac{Z_{t+1}}{Z_t} \ .$$

Also brauchen wir für jedes t lediglich die Ungleichung $Z_{t+1}/Z_t \leq e^{-2\gamma^2}$ nachzuweisen. Dies

leistet die folgende Rechnung:

$$\begin{aligned}
\frac{Z_{t+1}}{Z_t} &\overset{(8)}{=} \frac{\sum_{i=1}^m \exp(-y_i f_{t+1}(x_i))}{\sum_{j=1}^m \exp(-y_j f_t(x_j))} \\
&\overset{(8)}{=} \frac{\sum_{i=1}^m \exp(-y_i f_t(x_i)) \exp(-y_i w_{t+1} h_{t+1}(x_i))}{\sum_{j=1}^m \exp(-y_j f_t(x_j))} \\
&\overset{(9)}{=} \sum_{i=1}^m D_i^{(t+1)} \exp(-y_i w_{t+1} h_{t+1}(x_i)) \\
&= e^{-w_{t+1}} \sum_{i:y_i h_{t+1}(x_i)=1} D_i^{(t+1)} + e^{w_{t+1}} \sum_{i:y_i h_{t+1}(x_i)=-1} D_i^{(t+1)} \\
&= e^{-w_{t+1}}(1-\varepsilon_{t+1}) + e^{w_{t+1}}\varepsilon_{t+1} \\
&\overset{(5)}{=} \frac{1}{\sqrt{1/\varepsilon_{t+1}-1}}(1-\varepsilon_{t+1}) + \sqrt{1/\varepsilon_{t+1}-1}\,\varepsilon_{t+1} \\
&= \sqrt{\frac{\varepsilon_{t+1}}{1-\varepsilon_{t+1}}}(1-\varepsilon_{t+1}) + \sqrt{\frac{1-\varepsilon_{t+1}}{\varepsilon_{t+1}}}\varepsilon_{t+1} \\
&= 2\sqrt{\varepsilon_{t+1}(1-\varepsilon_{t+1})} \\
&\overset{(7)}{\leq} 2\sqrt{\left(\frac{1}{2}-\gamma\right)\left(\frac{1}{2}+\gamma\right)} = (1-4\gamma^2)^{1/2} \leq e^{-2\gamma^2}
\end{aligned}$$

Im letzten Schritt haben wir die bekannte Ungleichung $1 + a \leq e^a$ verwendet. **qed.**

Wir merken an, dass die Voraussetzung (7) in Theorem 10.9 mit einer Wahrscheinlichkeit von mindestens $1 - \delta/2$ für alle $t \in [T]$ erfüllt ist, da AdaBoost die Prozedur WL mit dem Parameter $\delta/(2T)$ aufruft.

Zusammensetzen der Puzzleteile. Es bezeichne

$$m_D(\varepsilon, \delta) = O\left(\frac{1}{\varepsilon^2}\left(D + \log\frac{1}{\delta}\right)\right)$$

eine Anzahl von Trainingsbeispielen, die den Schätzfehler für eine Hypothesenklasse der VC-Dimension D mit einer Wahrscheinlichkeit von mindestens $1 - \delta$ auf ε begrenzt. Die Basisklasse $\mathcal{B}$ habe die VC-Dimension d. Es sei $D(T, d)$ die aus Theorem 10.7 bekannte obere Schranke von $\mathrm{VCdim}(\mathrm{LIN}(\mathcal{B}, T))$.

Wir wollen klären, wie die Parameter T und m gewählt werden müssen, damit AdaBoost mit einer Wahrscheinlichkeit von mindestens $1 - \delta$ eine Hypothese mit einer Fehlerrate von höchstens ε abliefert. Es genügt zu diesem Zweck folgende Ziele durchzusetzen:

- Die empirische Fehlerrate von AdaBoost soll (unter der Voraussetzung (7)) durch $\varepsilon/2$ beschränkt sein. Hierzu genügt es,

$$T = \left\lceil \frac{\ln(2/\varepsilon)}{2\gamma^2} \right\rceil$$

zu setzen, da dann $e^{-2\gamma^2 T} \leq \varepsilon/2$.

- Mit einer Wahrscheinlichkeit von mindestens $1 - \delta/2$ soll $|L_S(h) - L_{\mathcal{D}}(h)| \leq \varepsilon/2$ für alle $h \in \mathrm{LIN}(\mathcal{B}, T)$ gelten. Hierzu genügt es,

$$m = m^{UC}_{\mathrm{LIN}(\mathcal{B},T)}(\varepsilon/2, \delta/2)$$

zu setzen.

Theorem 10.10 *Unter der Realisierbarkeitsannahme für $\mathcal{D}$ und $\mathcal{H}$ gilt folgendes. Es sei* WL *ein Algorithmus, der $\mathcal{H}$ mit Hypothesen aus $\mathcal{B}$ γ-schwach lernt. Dann ist AdaBoost (mit der Hilfsprozedur* WL *und mit den obigen Wahlen der Parameter T, m) ein PAC-Lerner für $\mathcal{H}$ mit Hypothesen aus $\cup_{T\geq 1}\mathrm{LIN}(\mathcal{B}, T)$.*

Beachte dabei folgendes: wenn $\mathcal{D}$ bezüglich $\mathcal{H}$ die Realisierbarkeitsannahme erfüllt, so gilt dies auch für alle von AdaBoost verwendeten Verteilungen $D^{(t)}$, da diese auf einer Trainingsmenge $S \sim \mathcal{D}^m$ konzentriert sind.