Lernalgorithmen

SoSe 2008 in Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

Lernalgorithmen Gesamtübersicht

0. Einführung
1. Identifikation (aus positiven Beispielen)
2. Zur Identifikation regulärer Sprachen, mit XML-Anwendung
3. HMM — Hidden Markov Models
4. Lernen mittels Anfragen & zur Roboterorientierung
5. Lernen mit negativen Beispielen
6. PAC-Lernen

Informanten-Lernen

Beispielstrom für L kommt mit positiven oder negativen Markierungen, d.h.: $(w, +)$ ist im Beispielstrom gdw. $w \in L$, und $(w, -)$ ist im Beispielstrom gdw. $w \notin L$.
Nach "endlicher Zeit" haben wir daher zwei Sample-Mengen X_+ und X_- gesehen, die zusammen das Sample $X = (X_+, X_-)$ bilden, mit $X_+ \subseteq L$ und $X_- \subseteq \bar{L}$.
Im Folgenden betrachten wir wieder das Lernen von regulären Sprachen, genauer von DEAs.
Wir zeichnen jetzt bei DEAs eine Menge akzeptierender Zustände F_a und eine Menge *verwerfender* Zustände F_r (mit $F_a \cap F_r = \emptyset$ aus.
$A = (\Sigma, Q, q_0, F_a, F_r, \delta)$ heißt *schwach konsistent* mit $X = (X_+, X_-)$ gdw. $\forall w \in X_+ : \delta^*(q_0, w) \in F_a$ und $\forall w \in X_- : \delta^*(q_0, w) \notin F_a$.
A heißt *stark konsistent* mit $X = (X_+, X_-)$ gdw. $\forall w \in X_+ : \delta^*(q_0, w) \in F_a$ und $\forall w \in X_- : \delta^*(q_0, w) \in F_r$.

Präfixbäume mit positiven und negativen Beispielen

Algorithm 8.1: BUILD-PTA

Data: a sample $\langle X_+, X_- \rangle$
Result: A PTA : $\mathcal{A} = \text{PTA}(\langle X_+, \emptyset \rangle) = \langle \Sigma, Q, q_\lambda, \mathbb{F}_\mathtt{A}, \mathbb{F}_\mathtt{R}, \delta \rangle$
$Q \leftarrow \{q_u : u \in \text{PREF}(X_+ \cup X-)\}$;
for $q_{u \cdot a} \in Q$ **do**
  $\delta(q_u, a) \leftarrow q_{ua}$
end
for $q_u \in Q$ **do**
  if $u \in X_+$ **then**
    $\mathbb{F}_\mathtt{A} \leftarrow \mathbb{F}_\mathtt{A} \cup \{q_u\}$
  end
  if $u \in X_-$ **then**
    $\mathbb{F}_\mathtt{R} \leftarrow \mathbb{F}_\mathtt{R} \cup \{q_u\}$
  end
end

Präfixbäume mit positiven und negativen Beispielen

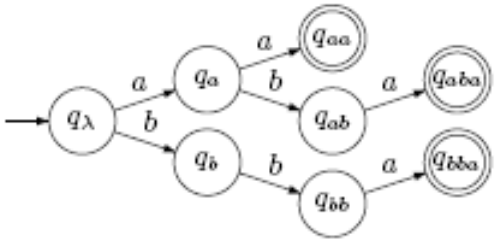


Fig. 8.3. PTA($\{(aa,+)\,(aba,+)\,(bba,+)\}$).

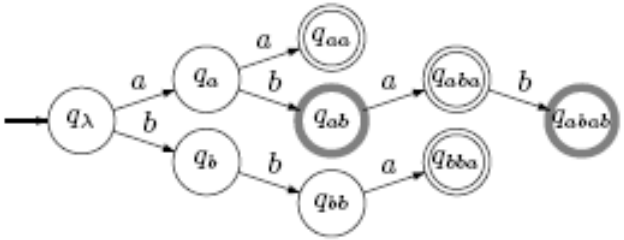


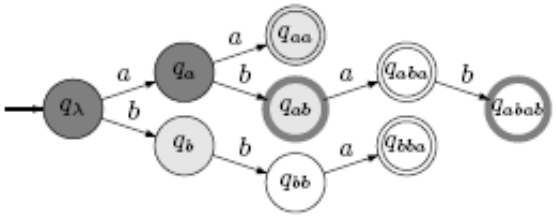
Fig. 8.2. PTA($\{(aa,+)\,(aba,+)\,(bba,+)\,(ab,-)\,(abab,-)\}$).

Zur Analyse / Angabe von Zustandsverschmelzungsalgorithmen

—*Rote Zustände* (dunkelgrau) sind bereits betrachtet und werden nicht mehr später revidiert.

—*Blaue Zustände* (hellgrau) sind augenblickliche Kandidaten für die Verschmelzung mit roten Zuständen.

—*Weiße Zustände* werden momentan nicht betrachtet.



Hierarchie der Lernbarkeit versus Beschreibungsfähigkeit

Definition [k-KNF] Eine Formel $C_1 \wedge \ldots \wedge C_\ell$, bei der jedes C_i eine Disjunktion von höchstens k Literalen über Boolsche Variablen ist, ist in k-*konjunktiver Normalform* (k-KNF).

Definition [k-Term-DNF] Eine Formel $T_1 \vee \ldots \vee T_k$, bei der jeder Term T_i eine Konjunktion über höchstens n Boolsche Variablen ist, ist in k-*Term disjunktiver Normalform* (k-Term-DNF).

Natürlich ist k-KNF ausdrucksstärker als k-Term-DNF, denn man kann jeden Ausdruck in k-Term-DNF in k-KNF schreiben, aber nicht umgekehrt.
Die Repräsentationsklasse $C = k$-KNF ist lernbar durch $H = k$-KNF.
Die Repräsentationsklasse $C = k$-Term-DNF ist nicht lernbar durch $H = k$-Term-DNF, wohl aber lernbar durch $H = k$-KNF.

Nützliche Formeln

$$\forall x > 0 : (1 + x^{-1})^x < e$$

$$\forall x > 0 : (1 - x^{-1})^x < e^{-1}$$

Daraus folgt insbesondere:

$$\forall x > 0 : (1 - x) < e^{-x}$$

Definition [PAC-Lernen von Formeln] Es sei eine Familie von Klassen Boolscher Funktionen, die in Anhängigkeit von einem Parameter n die Eingaben $\{0,1\}^n$ nach $\{0,1\}$ abbilden. S heißt *PAC-lernbar*, wenn es einen Algorithmus M sowie Polynome p, q gibt, mit folgenden Eigenschaften:

- M lernt in Abhängigkeit des Parameters n, δ, ϵ jede Funktion $f \in S$.
- M erhält als Eingabe die Parameter n, δ, ϵ sowie $m = \lceil p(n, \frac{1}{\delta}, \frac{1}{\epsilon}) \rceil$ Beispiele $(x_1, f(x_1))$, $\ldots$, $(x_m, f(x_m))$, wobei die Vektoren $x_m \in \{0,1\}^n$ zufällig gemäß einer unbekannten Wahrscheinlichkeitsverteilung P gewählt werden.
- M berechnet aus den Daten in Zeit $q(n, \delta, \epsilon)$ eine Formel g.
- Mit Wahrscheinlichkeit $1 - \delta$ (“probably”) gilt, dass $P(\{x : f(x) \neq g(x)\}) \leq \epsilon$ ist (“approximately correct”).

Satz: Die Klasse $S_{mon} = \{f_a : (\forall x)[f_a(x) = 0 \Leftrightarrow x \leq_{lex} a]\}$ aller monotonen Funktionen ist PAC-lernbar.

Beweis Der Lernalgorithmus bestimmt unter den Beispielen das lexikographisch größte b mit $f(b) = 0$ und wählt die Funktion f_b.
Falls kein solches Beispiel existiert, wird $b = 0^n$ gewählt.
Zu bestimmen bleibt die notwendige Anzahl m der Beispiele, damit der Algorithmus seinen Spezifikationen genügt.
Es sei c der lexikographisch erste Vektor mit $P(\{d : c <_{lex} x \leq_{lex} a\}) \leq \epsilon$.
Es gibt zwei Fälle:

- $c = 0^n$: Dann ist stets $c \leq_{lex} b \leq_{lex} a$ und mit Wahrscheinlichkeit 1 wird eine Formel gelernt, die der Bedingung $P(\{x : f(x) \neq g(x)\}) \leq \epsilon$ genügt.

- $c \neq 0^n$: Dann ist $P(\{x : c \leq_{lex} x \leq_{lex} a\}) > \epsilon$. $\rightsquigarrow$ Mit Wahrscheinlichkeit $1-(1-\epsilon)^m$ ist 1 aus m Beisp. in diesem Intervall.
  m muss so groß gewählt werden, dass $1-(1-\epsilon)^m \geq 1-\delta$ ist.

Also ist die Bedingung $1-(1-\epsilon)^m \geq 1-\delta$ hinreichend. Forme um:

$$\begin{aligned} 1-\delta &\leq 1-(1-\epsilon)^m \\ (1-\epsilon)^m &\leq \delta \\ m \cdot \log(1-\epsilon) &\leq \log(\delta) \\ m &\leq \frac{\log(\delta)}{\log(1-\epsilon)} \end{aligned}$$

Nun benötigt man am Ende nur ein hinreichend großes m, dieses muss aber nicht optimal sein.
Man kann den Term $|\log(1-\epsilon)| = \epsilon+\epsilon^2+\epsilon^3+\ldots$ nach unten durch ϵ abschätzen

und $|\log(\delta)|$ nach oben durch $\frac{1}{\delta}$ abschätzen.
Dadurch erhält man $m \geq \frac{1}{\delta\epsilon}$.
Dieses m ist ein Polynom in n, $\frac{1}{\delta}$ und $\frac{1}{\epsilon}$.
Die Rechenzeit ist ebenfalls polynomial in n und der Beispielanzahl m, also polynomial in den drei Parametern von p. Zusammenfassend gilt:

> Zu gegebenem n, δ und ϵ liest M dann $\lceil \frac{2}{\delta\epsilon} \rceil$ Beispiele $(x, f(x))$ ein und bestimmt unter ihnen das lexikographisch größte b mit $f(b) = 0$. Falls es ein solches nicht gibt, ist $b = 0^n$. Dann gilt mit Wahrscheinlichkeit $1 - \delta$, dass der Fehler von g bezüglich P durch ϵ begrenzt ist, d.h.: $P(\{x : g(x) \neq f(x)\}) \leq \epsilon$.

Um z.B. mit 99% Wahrscheinlichkeit einen Fehler von nur 1% zu haben, benötigt man 10000 Beispiele.

Satz: Die Klasse S_{k-KNF} aller Formeln in konjunktiver Normalform aus Termen mit höchstens k Variablen ist PAC-lernbar.

Beweis Man betrachte folgenden Lernalgorithmus:
g wird initialisiert als eine Konjunktion über alle Terme in bis zu k Variablen, d.h. $g = t_1 \wedge t_2 \wedge \ldots \wedge t_h$ mit $t_1 = (x_1 \vee x_2 \vee \ldots \vee x_k), \ldots$
(*) Lies die nächsten $m = \lceil \frac{1}{\delta\epsilon} \rceil$ Beispiele ein.
Falls g alle richtig klassifiziert, gib g aus und terminiere.
Sonst ist $g(x) = 0$ für einige Beispiele $(x, 1)$.

> Entferne alle Terme t von g mit $t(x) = 0$ für ein Beispiel $(x, 1)$.
> Durchlaufe erneut die Schleife (*).

g besteht zu Beginn aus $h = (2n)^k$ Termen.
Bei jedem Durchlauf der Schleife wird mindestens ein Term gelöscht oder die Schleife verlassen.
Daher kommt man mit $(2n)^k + 1$ Durchläufen der Schleife aus.
Beim Verlassen der Schleife stimmt g mit der zu lernenden Funktion f auf $m = \lceil \frac{1}{\delta\epsilon} \rceil$ neu gezogenen Beispielen überein, die von denjenigen stochastisch unabhängig sind, aus welchen g berechnet wurde.

g enthält dann alle Terme, die nicht den Daten widersprechen, d.h. g ist die Konjunktion aller Terme t, die für jedes der Beispiele $(x, 1)$ die Bedingung $t(x) = 1$ erfüllen.
f dagegen enthält nur einige dieser Terme, also gilt $g(x) \leq f(x)$ für alle Vektoren x.
Zu zeigen bleibt, dass die Menge der Unterschiede klein ist, d.h. $P(\{x : g(x) = 0 \wedge f(x) = 1\}) \leq \epsilon$.

Wie im Beweis des vorigen Satzes kann man auch hier Folgendes zeigen:
Wenn es genug Unterschiede gibt, d.h. wenn $P(\{x : f(x) \neq g(x)\}) > \epsilon$ ist, dann ist mit Wahrscheinlichkeit $1 - \delta$ mindestens eines der letzten m Beispiele in dieser Menge $\{x : f(x) \neq g(x)\}$.
Da aber g und f auf den zuletzt gezogenen m Beispielen übereinstimmen, gilt im Umkehrschluss, dass g und f mit Wahrscheinlichkeit $1 - \delta$ hinreichend ähnlich sind, d.h. mit Wahrscheinlichkeit $1 - \delta$ ist $P(\{x : f(x) \neq g(x)\}) \leq \epsilon$.
Also ist S_{k-KNF} PAC-lernbar mit der Schranke

$$p(n, \frac{1}{\delta}, \frac{1}{\epsilon}) = (2n)^{k+1} \cdot \frac{1}{\delta} \cdot \frac{1}{\epsilon}$$

auf die Gesamtzahl der Beispiele.

Vapnik-Chervonenkis-Dimension

Definition [Stichprobenkomplexität] Die Stichprobenkomplexität eines Algorithmus für gegebenes H, C, ϵ und δ ist die Anzahl von Beispielen $m(\epsilon, \delta, H)$, nach denen der Algorithmus spätestens ein beliebiges $c \in C$ mit Wahrscheinlichkeit δ bis auf einen Fehler von ϵ approximiert (PAC-gelernt) hat.

Zunächst benutzen wir einen Hypothesenraum mit endlicher Größe $|H|$.

Satz: [Stichprobenkomplexität] Die Stichprobenkomplexität m eines endlichen Hypothesenraums H mit $H = C$ ist begrenzt durch

$$m(\epsilon, \delta, H) \geq \frac{1}{\epsilon}(\ln |H| + \ln(1/\delta));$$

das kleinste m, für das die Ungleichung gilt, gibt die Anzahl von Beispielen an, nach denen spätestens jedes $c \in C$ PAC-gelernt werden kann.

Beweis (nach Haussler) Der Versionenraum zu H enthält zu jedem Zeitpunkt nur mit den bisher gesehenen Beispielen konsistente Hypothesen.
Der Hypothesenraum ist endlich.
Wir betrachten den schlimmsten Fall: bezüglich der bisher gesehenen Beispiele ist eine Hypothese im Versionenraum zwar korrekt, tatsächlich ist ihr Fehler aber größer als ϵ.
Die Wahrscheinlichkeit, daß eine beliebige Hypothese mit Fehler größer als ϵ ein zufällig gezogenes Beispiel richtig klassifiziert, ist also höchstens $(1-\epsilon)^m$.
Bei k Hypothesen ist die Wahrscheinlichkeit, dass eine von ihnen schlecht ist (d.h. einen Fehler größer als ϵ hat), höchstens $k(1-\epsilon)^m$.
Daher ist die Wahrscheinlichkeit, dass eine der Hypothesen im Versionenraum einen Fehler größer als ϵ hat, höchstens
$|H|(1-\epsilon)^m$.
Dies beschränkt die Wahrscheinlichkeit, dass m Beispiele nicht ausreichen, um die schlechten Hypothesen aus dem Versionenraum zu tilgen.
Da ϵ zwischen 0 und 1 liegt, können wir sie ganz grob abschätzen als $|H|e^{-\epsilon m}$. Damit ein Begriff PAC-gelernt wird, muss diese Wahrscheinlichkeit kleiner sein als δ:

$$|H|e^{-\epsilon m} \leq \delta.$$

Eine Umformung dieser Gleichung ergibt das Gewünschte.

Sei H der Hypothesenraum über X und S eine m-elementige Teilmenge von X. S wird von H *zerschmettert* (shattered), falls es für alle $S' \subseteq S$ eine Hypothese $h_{S'} \in H$ gibt, die S' abdeckt, d.h. $S \cap h_{S'} = S'$.
M.a.W.: $2^S = \{ h \cap S \mid h \in H \}$.
Alle Teilmengen von S werden also durch Hypothesen in H erkannt.

Die *Vapnik-Chervonenkis-Dimension* von H, $VCdim(H)$, ist die Anzahl der Elemente von der größten Menge S, wobei S von H zerschmettert wird.

$$VCdim(H) = \max\{m : \exists S \subseteq X, |S| = m, H \text{ zerschmettert } S\}.$$

Sie gibt also an, wieviele Unterschiede H machen kann. Wenn es kein Maximum der Kardinalität von S gibt, ist $VCdim$ unendlich.

Mitteilung[Blumer] C ist PAC-lernbar gdw. $VCdim(C) < \infty$.

Satz: [VC-Dimension endlicher Hypothesenräume] Wenn der Hypothesenraum H endlich ist, dann ist $VCdim(H) \leq \log_2(|H|)$.

Beweis Um eine Menge der Größe m zu zerschmettern, sind mindestens 2^m verschiedene Hypothesen nötig, weil es ja 2^m verschiedene Teilmengen gibt. □

Umgekehrt kann man für $H \subseteq 2^X$ per Induktion zeigen:

$$|H| \leq (|X| + 1)^{VDdim(H)}.$$

Mitteilung Die Ermittlung der genauen Vapnik-Chervonenkis-Dimension ist NP-hart.

VC Dimension und PAC-Lernen

Begriffserweiterung für $H \subseteq \Sigma^*$:
$VCdim(H) : \mathbb{N} \to \mathbb{N}$, $VCdim(H)(n) = VCdim(\{h \cap \Sigma^{\leq n} \mid h \in H\})$.

Mitteilung: Dann gibt es einen Lernalgorithmus für H mit Stichprobenkomplexität

$$m(\epsilon, \delta, n) = \frac{1}{\epsilon}\left((n+1)VCdim(H)(n)\log 2 - \log \delta\right).$$

Dabei wird einfach irgendeine mit der gezogenen Beispielmenge konsistente Hypothese ausgegeben.

Folgerung: Eine Konzeptmenge ist mit polynomieller Stichprobenkomplexität lernbar gdw. ihre VC-Dimension ist polynomiell.

PAC-artiges Lernen regulärer Sprachen

Es gebe eine beliebige aber fixierte Wahrscheinlichkeitsverteilung $P(\cdot)$ auf der Menge aller Wörter über dem Alphabet Σ. Dem Lerner sei diese Verteilung unbekannt.

$R \subseteq \Sigma^*$ sei die zu lernende Sprache. Der Lerner kann Informationen durch Aufruf zweier Orakel gewinnen:

- IN(x) liefert 1 gdw. $x \in R$ (sonst 0);
- EX liefert gemäß der Wahrscheinlichkeitsverteilung P ein Paar $(x, d) \in \Sigma^* \times \{0, 1\}$ mit $x \in R$ gdw. $d = 1$. Aufeinanderfolgende Aufrufe von EX seien statistisch unabhängig.

Der (zu konstruierende) Lernalgorithmus L_a^* bekommt bei seinem Aufruf zwei Parameter übergeben, die *Genauigkeit* ϵ und die *Konfidenz* δ (beide in $(0,1)$ gelegen).
Ziel ist es, eine *ϵ-Näherung* von R zu inferieren, d.i., einen Automaten A zu finden, so dass die Wahrscheinlichkeit für das Ereignis "symmetrische Differenz von R und $L(A)$" höchstens gleich ϵ ist, also

$$\sum_{x \in R \triangle L(A)} P(x) \leq \epsilon$$

gilt.
Ist A eine ϵ-Näherung von R, so ist die Wahrscheinlichkeit dafür, durch einen Aufruf von EX einen Unterschied von R und $L(A)$ zu finden, höchstens gleich ϵ.

L_a^* ist eine Modifikation von L^*.
Eine Elementanfrage wird durch einen Aufruf von IN simuliert.
Jede Hypothese wird durch eine Anzahl von Aufrufen von EX getestet, was in diesem stochastischen Modell die Äquivalenzanfragen an den Lehrer ersetzt.
Liefert nämlich irgendein Aufruf von EX ein Paar (x, d), so dass $d = 1$, aber $M(S, E, T)$ lehnt x ab (oder umgekehrt), so fungiert x als ein Gegenbeispiel für die Hypothese $M(S, E, T)$ von L_a^*, und L_a^* modifiziert seine Hypothese wie vordem L^*.
Liefert keiner der Aufrufe von EX ein Gegenbeispiel, so hält L_a^* mit der Ausgabe $M(S, E, T)$.

Frage: Wieviele Aufrufe –in Abhängigkeit von δ und ϵ– muss L_a^* machen, um eine Hypothese “genügend” zu testen?

Wir lassen eine Abhängigkeit von der Zahl der bisher getesteten Hypothesen zu. Sei

$$r_i = \frac{1}{\epsilon}\left(\log\frac{1}{\delta} + (\log 2)(i+1)\right).$$

Sind bislang i Hypothesen getestet worden, so macht $L_a^*\ \lceil r_i \rceil$ Aufrufe von EX.

Satz: Es sei n die Anzahl der Zustände des minimalen DEAs für die unbekannte, zu lernende reguläre Sprache $R \subseteq \Sigma^*$.
Dann terminiert L_a^* nach

$$\mathcal{O}\left(n + \frac{1}{\epsilon}\left(n \log \frac{1}{\delta} + n^2\right)\right)$$

vielen Aufrufen des EX Orakels. Die Wahrscheinlichkeit dafür, dass der dabei von L_a^* ausgegebene Automat eine ϵ-Näherung von R ist, beträgt mindestens $1 - \delta$.

Die Laufzeit von L_a^* ist polynomiell in n und der Länge der durch die EX-Aufrufe erhaltenen Beispiele.

Beweis Wie wir schon bei der Analyse des Algorithmus L^* gesehen hatten, werden höchstens $n-1$ Gegenbeispiele verarbeitet, bis der Algorithmus terminiert. Daher beträgt die Zahl der Aufrufe von EX bis zur Terminierung höchstens:

$$\begin{aligned}\sum_{i=1}^{n-2}(r_i+1) &= (n-1)+\frac{1}{\epsilon}\left((n-1)\log\frac{1}{\delta}+(\log 2)\sum_{i=0}^{n-2}(i+1)\right)\\ &\in O\left(n+\frac{1}{\epsilon}\left(n\log\frac{1}{\delta}+n^2\right)\right).\end{aligned}$$

Wie groß ist die Wahrscheinlichkeit dafür, dass L_a^* mit einer Hypothese terminiert, die keine ϵ-Näherung von R darstellt, nachdem bislang i Hypothesen getestet wurden?
Diese beträgt (sozusagen in "Runde i") höchstens $(1-\epsilon)^{r_i}$.
Daher ist die Wahrscheinlichkeit dafür, dass der dabei von L_a^* ausgegebene Au-

tomat keine ϵ-Näherung von R ist, höchstens:

$$\begin{aligned}\sum_{i=0}^{n-2}(1-\epsilon)^{r_i} &\leq \sum_{i=0}^{n-2} e^{-\epsilon r_i} \\ &\leq \sum_{i=0}^{n-2} \frac{\delta}{2^{i+1}} \leq \delta.\end{aligned}$$