5. Lernen von Konzepten

Literatur

Leslie G. Valiant, A Theory of Learnable, CACM 27 (1984), 1134 - 1142.

B. K. Natarajan, Machine Learning: A Theoretical Approach, Morgan Kaufmann 1991.

M. J. Kearns, U. V. Vazirani, An Introduction to Computational Learning Theory, MIT Press 1994.

T. M. Mitchell, Machine Learning, McGraw-Hill 1997.

Viele Lernprobleme bestehen darin, anhand von positiven und negativen Beispielen allgemeine Konzepte wie z.B. „Katze", „Auto" oder „Haus" zu lernen. Für das Konzept „Katze" könnten

- „Tier", „hat vier Beine", „kann auf Bäume klettern" positive Beispiele und
- „Auto", „kann bellen" negative Beispiele

sein.

Ziel:
Formale Definition derartiger Lernprobleme.

Sei S ein diskreter (z.B. $\mathbb{N}$) oder ein kontinuierlicher (z.B. $\mathbb{R}$) Ereignisraum. Die Elemente von S heißen Beispiele. Ein Konzept c ist eine Teilmenge von S. Eine Konzeptklasse C ist eine Menge von Konzepte. Sei $c \subseteq S$ ein Konzept. Dann heißt die Abbildung $f_c: S \to \{0,1\}$ mit

$$f_c(x) = \begin{cases} 1 & \text{falls } x \in c \\ 0 & \text{sonst} \end{cases}$$

charakteristische Funktion des Konzeptes c.

Ziel:
Entwicklung von Lernsysteme für Konzeptklassen, die polynomielle Lernzeit haben.

$\Rightarrow$

1) Es kann nur eine polynomiell große Datenmenge beobachtet werden.

2) Mit den beobachteten Daten kann nur das gemacht werden, was in Polynomzeit durchführbar ist.

$\Rightarrow$

Es ist häufig unmöglich, ein Konzept präzise zu lernen, so dass man nur hoffen kann, das zu lernende Konzept approximativ zu lernen.

$\rightsquigarrow$

Fragen:

1. Wie bestimmt man aus dem Ereignisraum diejenige Beispiele, die beobachtet werden?
2. Wie misst man die Approximationsgüte des gelernten Konzeptes im Vergleich zum zu lernenden Konzept?

Idee:
Wähle die zu beobachteten Beispiele gemäß einer Wahrscheinlichkeitsverteilung zufällig aus. Drücke die Approximationsgüte des gelernten Konzeptes im Vergleich zum zu lernenden Konzept durch die Wahrscheinlichkeit aus, dass sich beide Konzepte bezüglich zufällig gewählten Beispielen unterscheiden.

Durchführung:

Eine Konzeptklasse $C \in 2^S$ heißt genau dann PAC-lernbar (probably approximately correct-lernbar), wenn ein (probabilistischer) Lernalgorithmus OT existiert, für den folgendes erfüllt ist:

i) OT erhält als Eingabe einen Fehlerparameter $\varepsilon \in (0,1]$, einen Vertrauensparameter $\delta \in (0,1]$ und einen Längenparameter $n \in \mathbb{N}$.

ii) OT kann ein Orakel EXAMPLE bezüglich des zu lernenden Konzeptes $c \in C$ befragen.

EXAMPLE wählt dann bezüglich einer unbekannten Wahrscheinlichkeitsverteilung P auf $S^{\leq n} := \{v \in S \mid |v| \leq n\}$ ein Beispiel aus und teilt dieses sowie den Wert der charakteristischen Funktion des zu lernenden Konzeptes für dieses Beispiel mit.

iii) $\forall$ zu lernenden Konzepte $c \in C$ und alle Wahrscheinlichkeitsverteilungen P auf $S^{\leq n}$ hält der Algorithmus OT mit einer Ausgabe $c' \in C$, so dass

$$Pr\left(\sum_{v \in S^{\leq n} : f_c(v) \neq f_{c'}(v)} P(v) < \varepsilon \right) \geq 1 - \delta.$$

Eine Konzeptklasse C heißt <u>polynomiell PAC-lernbar</u>, wenn C PAC-lernbar ist und ein Polynom p existiert, so dass der Lernalgorithmus OT die Laufzeit $p(|c|, \frac{1}{\varepsilon}, \frac{1}{\delta}, n)$ hat, wobei $c \in C$ das zu lernende Konzept ist.

5.1 Lernen von Boolesschen Ausdrücken

- Ereignisraum $\{0,1\}^n$.
- $a \in \{0,1\}^n \hat{=}$ Belegung von n Variablen $x_1, x_2, \ldots, x_n$.
- C_n bezeichnet die Menge aller Monome über die Variablenmenge $\{x_1, x_2, \ldots, x_n\}$.
- Für $m \in C_n$ bezeichnet $|m|$ die Länge des Monoms m; d.h., die Anzahl der Literale in m

Beispiel:

Monom $m = x_1 \wedge \bar{x}_3 \wedge x_4 \wedge x_6 \wedge \bar{x}_8$

$\{a \in \{0,1\}^4 \mid a_1 = 1, a_3 = 0, a_4 = 1, a_6 = 1, a_8 = 0\}$
ist die Menge der Belegungen, die m erfüllen.

Es gilt: $|m| = 5$.

◊

Ziel:
Konstruktion eines in n, $\frac{1}{\varepsilon}$ und $\frac{1}{\delta}$ polynomiellen Lernalgorithmus für Boolesche Monome.

Idee:

- Starte mit dem Monom h, das alle Literale enthält. D.h., $h = x_1 \bar{x}_1 x_2 \bar{x}_2 \dots x_n \bar{x}_n$.
- Befrage EXAMPLE und ignoriere dabei negative Beispiele $\langle a, 0 \rangle$.
- Sei $\langle a, 1 \rangle$ ein positives Beispiel, das EXAMPLE nach einer Befragung präsentiert.

  Modifiziere das aktuelle Monom h durch

$$\begin{cases} \text{Entfernen des Literals } x_i & \text{falls } a_i = 0 \\ \text{Entfernen des Literals } \bar{x}_i & \text{falls } a_i = 1 \end{cases}$$

für $1 \le i \le n$.

$\Rightarrow$

i) Nach dem ersten positiven Beispiel $\langle a, 1 \rangle$ ist das resultierende Monom h erfüllbar.

ii) h enthält stets alle Literale aus c. Dies bedeutet, dass negative Beispiele für c auch stets negative Beispiele für h sind.

Ziel:
Analyse der Fehlerwahrscheinlichkeit.

Sei z ein Literal in h, das kein Literal in c ist. Sei

$$Q(z) := \sum_{a \in S^{sn}:\ c(a)=1,\ z(a)=0} P(a)$$

Dann gilt

$$\sum_{a \in S^{sn}:\ c(a) \neq h(a)} P(a) \leq \sum_{z \in h} Q(z).$$

Annahme:

$$Q(z) < \frac{\varepsilon}{2n} \quad \forall\, z \in h.$$

Dann gilt

$$\sum_{a \in S^{sn}:\ c(a) \neq h(a)} P(a) < 2n \cdot \frac{\varepsilon}{2n} = \varepsilon.$$

Somit verbleibt noch der Fall, dass $z \in h$ existiert mit $Q(z) \geq \frac{\varepsilon}{2n}$.

Ein Literal $z \in h$ heißt <u>schlecht</u>, falls

$$Q(z) \geq \frac{\varepsilon}{2n}.$$

<u>Ziel</u>:
Beschränkung der Wahrscheinlichkeit, dass ein schlechtes Literal in h enthalten ist.

<u>Beobachtung</u>
Für jedes schlechte Literal z ist die Wahrscheinlichkeit $Q(z)$, dass nach einer Befragung von EXAMPLE z aus h eliminiert wird, mindestens $\frac{\varepsilon}{2n}$.

$\Rightarrow$

Die Wahrscheinlichkeit, dass nach m Befragungen z nach wie vor in h enthalten ist, ist

$$\leq \left(1 - \frac{\varepsilon}{2n}\right)^m.$$

$\Rightarrow$

Die Wahrscheinlichkeit, dass irgendein schlechtes Literal in h nach m Befragungen von EXAMPLE enthalten ist, ist

$$\leq 2n \left(1 - \frac{\varepsilon}{2n}\right)^m.$$

Um einen PAC-Lernalgorithmus zu erhalten wählen wir m derart, dass

$$2n\left(1-\frac{\varepsilon}{2n}\right)^m \leq \delta.$$

Dies ist äquivalent zu

$$m \geq \frac{2n}{\varepsilon}\left(\ln 2n + \ln\frac{1}{\delta}\right).$$

Jede Befragung von EXAMPLE benötigt $O(n)$ Zeit. Also beträgt die Gesamtlaufzeit $O(m \cdot n)$, was für $m := \lceil \frac{2n}{\varepsilon}(\ln 2n + \ln\frac{1}{\delta}) \rceil$ polynomiell in $\frac{1}{\varepsilon}$, $\frac{1}{\delta}$ und n ist.

Insgesamt haben wir folgenden Satz bewiesen:

Satz 5.1
Die Konzeptklasse der Booleschen Monomen ist polynomiell PAC-lernbar.

Die Disjunktion von Monomen ist ein Boolescher Ausdruck in disjunktiver Normalform. Die Disjunktion von drei Monomen ist ein Boolescher Ausdruck in 3-Term disjunktiver Normalform (3-Term DNF). Falls die Monome Elemente von C_n sind, dann ist die Disjunktion von drei Monomen in 3-Term DNF(n). Bezeichne $C_{n,3}$ die Konzeptklasse aller Booleschen Ausdrücke in 3-Term DNF(n).

Ziel:
Finden von Eigenschaften, die zeigen, dass es schwer ist, einen in n, $\frac{1}{\varepsilon}$ und $\frac{1}{\delta}$ polynomiellen PAC-Lernalgorithmus für $C_{n,3}$ zu konstruieren.

Bezeichne RP die Klasse derjenigen Sprachen L, die randomisiert in polynomieller Zeit akzeptiert werden können. D.h., es gibt einen probabilistischen Algorithmus O_7, der folgende Eigenschaften besitzt:

i) O_7 erhält als Eingabe einen String $x \in \Gamma^*$, wobei $L \subseteq \Gamma^*$ und einen Parameter $\delta \in (0,1]$.

ii) O_7 entscheidet mit Wahrscheinlichkeit $\geq 1-\delta$ korrekt, ob $x \in L$ oder nicht. Falls $x \notin L$, dann ist die Ausgabe von O_7 immer korrekt.

iii) O_7 hat in $|x|$ und $\frac{1}{\delta}$ polynomielle Laufzeit.

Bemerkung:
Ob RP = NP ist eine der großen offenen Fragen in der Theoretischen Informatik. Vielfach wird vermutet, dass RP ≠ NP.

Satz 5.2
Falls ein polynomieller PAC-Lernalgorithmus für die Konzeptklasse $C_{n,3}$ existiert, dann gilt RP = NP.

Beweis:

Idee:
Reduktion eines NP-vollständigen Problems $L \subseteq \Gamma^*$ auf PAC-Lernen eines Konzeptes in $C_{n,3}$.

allgemeine Technik:
gegeben: Eingabe $x \in \Gamma^*$, für die $x \in L$ entschieden werden soll.

Ziel:
Konstruktion einer Beispielmenge

$$S_x := \{ \langle y_1, b_1 \rangle, \langle y_2, b_2 \rangle, \ldots, \langle y_m, b_m \rangle \},$$

so dass folgendes erfüllt ist:

1) Die Konstruktion erfolgt in polynomieller Zeit. Dies impliziert insbesondere, dass $m \leq p(n)$ für ein Polynom p, wobei $n = |x|$.

2) $x \in L \Leftrightarrow S_x$ ist konsistent mit einem Konzept $c \in C$, wobei C die zu lernende Konzeptklasse ist.

D.h.,
$x \in L \Leftrightarrow f_c(y_i) = b_i$ für $1 \leq i \leq m$.

Gegeben S_x und ein polynomieller PAC-Lernalgorithmus $\mathcal{A}$ für C kann mit Wahrscheinlichkeit

$\geq 1-\delta$ wie folgt entschieden werden, ob in C ein mit S_x konsistentes Konzept existiert.

- Wähle für O_1 den Fehlerparameter $\varepsilon := \frac{1}{2m}$.
- Beantworte jede Befragung von EXAMPLE durch O_1 mit einem zufällig gewählten Paar $\langle y_i, b_i \rangle \in S_x$. Dabei wird jedes Paar in S_x mit derselben Wahrscheinlichkeit $\frac{1}{m}$ gewählt.
- ...
- Überprüfe die Ausgabe von O_1 auf Konsistenz mit S_x und akzeptiere x genau dann, wenn dies der Fall ist.

Obiger Algorithmus besitzt folgende Eigenschaften:

1) Falls ein mit S_x konsistentes Konzept $c \in C$ existiert, dann nimmt der Algorithmus den Lernalgorithmus O_1 mit der Wahrscheinlichkeitsverteilung

$$P(y_i) = \begin{cases} \frac{1}{m} & \text{falls } \langle y_i, b_i \rangle \in S_x \\ 0 & \text{sonst} \end{cases}$$

nach.

2) Für jedes nicht mit S_x konsistente Konzept h gilt

$$\sum_{y:\, f_c(y) \neq f_h(y)} P(y) \geq \frac{1}{m} = 2\varepsilon.$$

Sei $h \in C$ die Ausgabe von O_1 bei obiger

Simulation. Dann impliziert

$$\sum_{y:\, f_c(y) \neq f_h(y)} P(y) < \varepsilon,$$ dass h mit S_x konsistent ist.

3) Falls C kein mit S_x konsistentes Konzept enthält, dann kann OT auch kein solches finden und die Überprüfung fällt dementsprechend aus.

OT PAC-Lernalgorithmus $\Rightarrow$

OT gibt mit Wahrscheinlichkeit $\geq 1-\delta$ ein Konzept mit Fehlerrate $< \varepsilon$ aus, falls ein mit S_x konsistentes Konzept existiert.

Falls OT solches ausgibt, dann ist diese Ausgabe mit S_x konsistent, so dass dann x als Element der Sprache akzeptiert wird. Falls kein mit S_x konsistentes Konzept existiert, dann wird x bei jeder Ausgabe von OT nicht als Element der Sprache L akzeptiert.

$\Rightarrow$

Es wird mit Wahrscheinlichkeit $\geq 1-\delta$ korrekt entschieden, ob $x \in L$ oder nicht. Falls $x \notin L$, dann akzeptiert OT die Eingabe x nicht.

Zum Beweis des Satzes wählen wir folgendes NP-vollständige Problem:

3 - Färbbarkeit von Graphen.

Eingabe:
ungerichteter Graph $G = (V, E)$ mit Knotenmenge $V = \{1, 2, \ldots, n\}$ und Kantenmenge $E \subseteq V \times V$.

Frage:
Kann V mit drei Farben legal gefärbt werden? (D.h., benachbarte Knoten erhalten verschiedene Farben).

Konstruktion von S_G:

Seien S_G^+ die positiven und S_G^- die negativen Beispiele in S_G. D.h.,

$$S_G = S_G^+ \cup S_G^-.$$

Für $1 \leq i, j \leq n$ seien

$$v(i) := (a_1, a_2, \ldots, a_n), \text{ wobei für } 1 \leq k \leq n$$

$$a_k := \begin{cases} 0 & \text{falls } k = i \\ 1 & \text{sonst} \end{cases}$$

und $e(i,j) := (e_1, e_2, \ldots, e_n)$, wobei für $1 \leq k \leq$

$$e_k := \begin{cases} 0 & \text{falls } k \in \{i, j\} \\ 1 & \text{sonst} \end{cases}$$

Wir definieren dann

$$S_G^+ := \{ \langle v(i), 1 \rangle \mid 1 \leq i \leq n \} \text{ und}$$

$$S_G^- := \{ \langle e(i,j), 0 \rangle \mid (i,j) \in E \}.$$

Beh.:

G dreifärbbar $\Leftrightarrow$ S_G ist konsistent mit einem 3-Term DNF(n) - Ausdruck.

Bew. d. Beh.:

„$\Rightarrow$"

Annahme: G ist dreifärbbar

Sei $f: V \to \{R, B, Y\}$ eine legale Färbung von G. Seien

$$R := \{ i \in V \mid f(i) = R \},$$
$$B := \{ i \in V \mid f(i) = B \} \text{ und}$$
$$Y := \{ i \in V \mid f(i) = Y \}.$$

Definiere

$$T_R := \bigwedge_{i \notin B \cup Y} x_i ,$$

$$T_B := \bigwedge_{i \in R \cup Y} x_i ,$$

$$T_Y := \bigwedge_{i \in R \cup B} x_i . \text{ und}$$

$$T := T_R \vee T_B \vee T_Y .$$

Konstruktion $\Rightarrow$

i) $\forall i \in R$ gilt: $v(i)$ erfüllt T_R
$\forall i \in B$ gilt: $v(i)$ erfüllt T_B
$\forall i \in Y$ gilt: $v(i)$ erfüllt T_Y

$\Rightarrow$

Jedes $\langle v(i), 1\rangle \in S_G^+$ erfüllt T.

ii) Da für jede Kante $(i,j) \in E$ die beiden Endknoten zu unterschiedlichen Mengen der Partition $R \cup B \cup Y$ von V gehören, erfüllt für jedes $\langle e(i,j), 0\rangle \in S_G^-$ die Belegung $e(i,j)$ den Ausdruck T nicht

$\Rightarrow$

$T = T_R \vee T_B \vee T_Y$ ist konsistent mit der Beispielmenge $S_G = S_G^+ \cup S_G^-$.

„$\Leftarrow$"

Annahme:

$\exists$ Ausdruck $T' := T_1 \vee T_2 \vee T_3 \in C_{n,3}$, der mit S_G konsistent ist.

2.2. $G = (V, E)$ ist dreifärbbar.

Hierzu definieren wir zunächst eine Färbung von G und zeigen dann, dass diese legal ist.

Für $1 \leq i \leq n$ sei

$$c(i) := \min \{ j \in \{1,2,3\} \mid v(i) \text{ erfüllt } T_j \}.$$

Die Färbung $f: V \to \{R, B, Y\}$ von G ist definiert durch

$$f(i) := \begin{cases} R & \text{falls } c(i) = 1 \\ B & \text{falls } c(i) = 2 \\ Y & \text{falls } c(i) = 3 \end{cases}$$

Da der Ausdruck $T' = T_1 \vee T_2 \vee T_3$ konsistent mit S_G ist, erfüllt $v(i)$ mindestens eines der Monome. Also ist $f(i)$ definiert $\forall i$.

2.2. f ist legal.

Annahme: f ist nicht legal

$\Rightarrow$

$\exists$ Kante $(i,j) \in E$ mit $f(i) = f(j)$.

$\Rightarrow$

$$c(i) = c(j)$$

$\Rightarrow$

$v(i)$ und $v(j)$ erfüllen dasselbe Monom $T_{c(i)}$.

Da die i-te Komponente in $v(i)$ den Wert ① und in $v(j)$ den Wert ⓪ hat, kommen weder x_i noch $\bar{x}_i$ im Monom $T_{c(i)}$ vor.

Genauso sieht man ein, dass weder x_j noch $\bar{x}_j$ in $T_{c(i)}$ vorkommen.

Da in $e(i,j)$ alle Variablen bis auf x_i und x_j mit 1, also genauso wie in $v(i)$ belegt sind und $v(i)$ das Monom $T_{c(i)}$ erfüllt, erfüllt auch die Belegung $e(i,j)$ das Monom $T_{c(i)}$ und somit auch T'.

Dies ist ein Widerspruch dazu, dass der Ausdruck T' mit S_G konsistent ist. Also war unsere Annahme falsch. D.h., f ist legal. ∎

Bemerkung:

Falls wir für die Repräsentation der Ausgabe eine andere als einen Ausdruck in 3-Term DNF erlauben, dann existiert möglicherweise ein polynomieller PAC-Lernalgorithmus für die Konzeptklasse 3-Term-DNF.

Ziel:

Beweis der Existenz eines polynomiellen PAC-Lernalgorithmus für eine konkrete andere Repräsentation der Ausgabe.

Hierzu unten wir aus, dass in der Booleschen

Algebra für Boolesche Variablen x, y und z das Distributivgesetz $x \wedge (y \vee z) = xy \vee xz$ gilt.

Idee:

- Unter Verwendung des Distributivgesetzes schreibe einen Ausdruck $T = T_1 \vee T_2 \vee T_3$ in 3-Term-DNF als Ausdruck in konjunktiver Normalform mit drei Literalen pro Klausel.

  $\rightsquigarrow$

  $$T_1 \vee T_2 \vee T_3 \equiv \bigwedge_{u \in T_1, v \in T_2, w \in T_3} (u \vee v \vee w)$$

  Derartige Ausdrücke sind in 3-konjunktiver Normalform (3-KNF).

- Reduziere das Problem "Lernen von Ausdrücken in 3-KNF" auf das Problem "Lernen von Monomen".

Durchführung:

Wir interpretieren einen Ausdruck in 3-KNF als ein Monom über eine größere neue Variablenmenge. D.h., für jede mögliche Disjunktion dreier Literalen $u \in T_1$, $v \in T_2$ und $w \in T_3$ definieren wir eine neue Variable x_{uvw} und ersetzen in dem Ausdruck in 3-KNF die Klausel $u \vee v \vee w$ durch diese Variable.

Dann lernen wir das so konstruierte Monom

unter Verwendung des polynomiellen PAC-Lernalgorithmus zum Lernen von Monomen.

Die Ausgabe des Lernalgorithmus kann durch Ersetzen jeder Variablen x_{uvw} durch seine korrespondierende Klausel $(u \vee v \vee w)$ wieder als Ausdruck in 3-KNF geschrieben werden.

$\Rightarrow$

Wenn wir erlauben, dass die Ausgabe des Lernalgorithmus ein Ausdruck in 3-KNF ist, dann gibt es einen polynomiellen PAC-Lernalgorithmus für Ausdrücke in 3-Term DNF.

Übung:
Zeigen Sie, dass die Ausgabe des obigen Lernalgorithmus nicht notwendigerweise äquivalent zu einem Ausdruck in 3-Term-DNF ist.

5.2 Ockhams Rasiermesserprinzip

Ockhams Rasiermesserprinzip besagt, dass das einfachste mit den Beispielen konsistente Konzept als Ausgabe des Lernalgorithmus gewählt werden soll.

Problem
Einfachste oder auch kürzeste Konzepte können

häufig überhaupt nicht oder nur mit zu großem Zeitaufwand berechnet werden.

Beispiel:
Die Berechnung eines kürzesten Ausdrucks in DNF mit maximal k Literalen pro Monom, der mit einer gegebenen Menge von Beispielen konsistent ist, ist NP-hart.

◆

⇝

Betrachte polynomiell berechenbare Approximationen von kürzeste Konzepten.

Seien C eine Konzeptklasse, $d \geq 1$ und $0 \leq \alpha < 1$ Konstanten. Ein (d, α)-Ockham-Algorithmus für C ist ein Lernalgorithmus OT, der folgende Eigenschaften besitzt:

i) Gegeben m Beispiele für ein beliebiges zu lernendes Konzept $c \in C$ der Länge $\leq s$ Bits gibt der Algorithmus OT ein zu den Beispielen konsistentes Konzept $c' \in C$ der Länge $\leq s^d m^\alpha$ aus.

ii) OT hat in m polynomielle Laufzeit.

Satz 5.3
Seien C eine Konzeptklasse, $d \geq 1$ und $0 \leq \alpha < 1$ Konstanten. Falls C einen (d, α)-Ockham-

Algorithmus O_T besitzt, dann ist C polynomiell PAC-lernbar.

Beweis:

Sei P die Wahrscheinlichkeitsverteilung, mittels der der Lernalgorithmus O_T die Beispiele unabhängig voneinander aus S wählt.

Betrachten wir Fehlerparameter $0 < \varepsilon < 1$ und Vertrauensparameter $0 < \delta < 1$ beliebig aber fest.

Idee:

Wähle m groß genug, so dass gezeigt werden kann, dass der (d, α)-Ockham-Algorithmus auch ein PAC-Lernalgorithmus ist.

Sei

$$m \geq \max\left\{ \left(\frac{2(s^d+1)}{-\log(1-\varepsilon)} \right)^{\frac{1}{1-\alpha}}, \frac{2 \cdot \log \frac{1}{\delta}}{-\log(1-\varepsilon)} \right\}$$

Die Wahl von m ergibt sich aus den nachfolgenden Berechnungen.

Konstruktion $\Rightarrow$

m ist polynomiell in s, $\frac{1}{\varepsilon}$ und $\frac{1}{\delta}$.

O_T (d, α)-Ockham-Algorithmus $\Rightarrow$

O_T gibt ein Konzept c' der Länge $\leq s^d m^\alpha$ aus.

z.z.

$$Pr\left(\sum_{v:\, f_c(v) \neq f_{c'}(v)} P(v) \geq \varepsilon \right) < \delta.$$

Betrachten wir hierzu folgende Teilmenge C' von C:

$$C' := \{ \bar{c} \in C \mid |\bar{c}| \leq s^d m^\alpha \}.$$

$\Rightarrow$

$$c' \in C'.$$

Sei $r := |C'|$.

Lemma 5.1

Sei $\bar{c} \in C'$ ein beliebiges Konzept in C', das sich bei m unabhängig gewählten Beispielen bezüglich der Werte der charakteristischen Funktion genauso wie das zu lernende Konzept c verhält. Dann gilt:

$$Pr\left(\sum_{v:\, f_c(v) \neq f_{\bar{c}}(v)} P(v) \geq \varepsilon \right) < (1-\varepsilon)^m \cdot r$$

Bevor wir das Lemma beweisen, führen wir den Beweis des Satzes zu Ende.

Wegen

$$m \geq \left(\frac{2(s^d+1)}{-\log(1-\varepsilon)} \right)^{\frac{1}{1-\alpha}}$$ gilt: