

- (a) Das **Online-Spiel**: In jeder Runde präsentiert ein Lehrer ein Beispiel, das ein Schüler klassifiziert. Nach wie vielen Runden hat der Schüler das unbekannte Zielkonzept gelernt?
- ▶ Charakterisierung der minimalen Rundenzahl.
 - ▶ Zusammenhang zur VC-Dimension.
 - ▶ Der Zusammenhang zwischen Online- und PAC-Algorithmen.
- (b) Wichtige algorithmische Ideen:
- ▶ Der **Halbierungs-Algorithmus**,
 - ▶ **Weighted-Majority**: Auswahl von Experten,
 - ▶ **Winnow**: Relevante Eigenschaften schnell bestimmen!
 - ▶ der **Perzeptron-Algorithmus**: das Online-Lernen von Halbräumen.

Das Online-Spiel

1. Das Spiel beginnt in Runde $t = 1$.
2. Solange das Spiel noch nicht beendet ist:
 - (a) Der Lehrer präsentiert ein Beispiel x_t .
 - (b) Der Schüler gibt eine Klassifizierung an.
/* Der Schüler ist an keine Hypothesenklasse gebunden. */
 - (c) Ist die Klassifizierung richtig, ist das Spiel beendet.
Bei einer falschen Antwort (x_t ist ein **Gegenbeispiel**),
erhält der Schüler die richtige Klassifikation.
 - (d) Setze $t := t + 1$.

- Der Schüler möchte möchte so schnell wie möglich gewinnen.
- Der Lehrer ist böseartig.

- (a) Sei A ein Algorithmus, mit dem der Schüler Beispiele klassifizieren kann. Dann ist

$$\text{Gegenbeispiel}_A(\mathcal{C}) := m,$$

falls bei Klassifikation der Beispiele durch A

- (*) jedes Konzept in \mathcal{C} nach höchstens m Gegenbeispielen für jeden Lehrer gelernt wird
- (*) und mindestens m Gegenbeispiele für irgendeinen Lehrer und irgendein Konzept benötigt werden.

- (b) Es ist

$$\text{Gegenbeispiel}(\mathcal{C}) := m,$$

falls es einen Algorithmus A mit $\text{Gegenbeispiel}_A(\mathcal{C}) = m$ gibt und falls $\text{Gegenbeispiel}_B(\mathcal{C}) \geq m$ für jeden Algorithmus B gilt.

Gegenbeispiel-Komplexität: Beispiele

(a) Sei $X = \{1, \dots, n\}$. Die Konzeptklasse $\mathbf{1}_n$ bestehe aus allen

„Einermengen“ $\{i\}$ für $1 \leq i \leq n$.

- ▶ Es ist $\text{Gegenbeispiel}(\mathbf{1}_n) = 1$:

Der Schüler behauptet, dass x_1 ein negatives Beispiel ist :-))

(b) Ein Online-Algorithmus für **MONOM** $_n$:

- ▶ Der Schüler klassifiziert anfänglich mit der Hypothese

$$x_1 \wedge \neg x_1 \wedge x_2 \wedge \neg x_2 \wedge \dots \wedge x_n \wedge \neg x_n.$$

- ▶ Der Schüler erhält ein Gegenbeispiel und arbeitet mit der maximal-konsistenten Hypothese weiter.
- ▶ Der Lehrer kann nur positive Gegenbeispiele geben. Es ist

$$\text{Gegenbeispiel}(\mathbf{MONOM}_n) \leq n + 1.$$

Für jede Konzeptklasse \mathcal{C} gilt

$$\text{VC}(\mathcal{C}) \leq \text{Gegenbeispiel}(\mathcal{C}).$$

1. Sei $s = \text{VC}(\mathcal{C})$ und \mathcal{C} zertrümmere die Menge $S = \{x_1, \dots, x_s\}$.
2. \implies **jede** Teilmenge von S ist ein Konzept.
3. $\implies \text{Gegenbeispiel}(\mathcal{C}) \geq s$. □

Eine erste Konsequenz:

$$n \leq \text{Gegenbeispiel}(\mathbf{MONOM}_n) \leq n + 1.$$

Endliche Konzeptklassen

Die Konzeptklasse \mathcal{C} bestehe aus endlich vielen Konzepten. Dann folgt

$$\text{Gegenbeispiel}(\mathcal{C}) \leq \log_2 |\mathcal{C}|.$$

1. Der Schüler führt den **Halbierungs-Algorithmus** aus
 - ▶ und klassifiziert mit der Mehrheitshypothese

$$x \in M \iff \left| \{c \in \mathcal{C} : x \in c\} \right| \geq \left| \{c \in \mathcal{C} : x \notin c\} \right|.$$

- ▶ Jedes falsch klassifizierende Konzept $c \in \mathcal{C}$ wird aus \mathcal{C} entfernt.
2. Ein Gegenbeispiel eliminiert mindestens die Hälfte aller Konzepte.

Insbesondere: $\text{Gegenbeispiel}_{\text{Halbierung}}(\mathcal{C}) \leq \log_2 |\mathcal{C}|.$

Wie viele Gegenbeispiele? Eine exakte Charakterisierung

Wie arbeitet ein Lehrer?

- (a) Der Lehrer beginnt mit einem ersten Beispiel $b_\epsilon := x_1$.
 - (b) Wenn der Lehrer gerade Beispiel b_w präsentiert, dann
 - ▶ zeigt er Beispiel b_{w0} bei einer negativen Klassifizierung von b_w durch den Schüler
 - ▶ bzw b_{w1} bei einer positiven Klassifizierung.
 - (c) Die Strategie des Lehrers L wird durch einen vollständigen binären, geordneten Baum B_L modelliert:
 - ▶ Innere Knoten w sind mit den Beispielen b_w markiert
 - ▶ und Kanten mit der Antwort des Schülers.
- ! Es muss zu jedem Weg W ein Konzept $c \in \mathcal{C}$ geben, so dass c **konsistent** mit den Klassifizierungen von W ist.

- 1 Ein solcher Baum heißt **konsistent** mit \mathcal{C} .
- 2 **Tiefe**(\mathcal{C}) ist die **maximale** Tiefe eines mit \mathcal{C} konsistenten Baums.
- 3 **Tiefe**(\mathcal{C}) \leq **Gegenbeispiel**(\mathcal{C}) ✓

Tiefe(\mathcal{C}) \geq Gegenbeispiel(\mathcal{C})

Ein pfiffiger Schüler:

1. Setze $i = 1$ und $\mathcal{C}_i := \mathcal{C}$.
2. Wiederhole, solange der Schüler noch nicht gewonnen hat:
 - (a) Wenn der Lehrer das Beispiel x vorlegt, dann
 - (b) bestimme $\mathcal{C}_i^{(0)} := \{c \in \mathcal{C}_i : x \notin c\}$ und $\mathcal{C}_i^{(1)} := \{c \in \mathcal{C}_i : x \in c\}$.
 - (c) Wenn $\text{Tiefe}(\mathcal{C}_i^{(0)}) \geq \text{Tiefe}(\mathcal{C}_i^{(1)})$, dann antworte mit Klassifizierung $a = 0$ und sonst mit Klassifizierung $a = 1$.
 - (d) Ist die Antwort falsch, dann setze $\mathcal{C}_{i+1} := \mathcal{C}_i^{(1-a)}$ und $i = i + 1$.

- Wenn $\text{Tiefe}(\mathcal{C}_i^{(0)}) < \text{Tiefe}(\mathcal{C}_i)$ oder $\text{Tiefe}(\mathcal{C}_i^{(1)}) < \text{Tiefe}(\mathcal{C}_i)$, dann sind wir fertig: Ein Gegenbeispiel senkt die Tiefe um mindestens 1.
- Also ist $\text{Tiefe}(\mathcal{C}_i^{(0)}) = \text{Tiefe}(\mathcal{C}_i^{(1)}) = \text{Tiefe}(\mathcal{C}_i)$.
Seien $B^{(0)}, B^{(1)}$ tiefste mit $\mathcal{C}_i^{(0)}$ bzw $\mathcal{C}_i^{(1)}$ konsistente Bäume \implies

$$\text{Tiefe}(\mathcal{C}) = t + 1 \quad \text{⚡}$$

(a) Für alle Konzeptklassen gilt

$$\text{VC}(\mathcal{C}) \leq \text{Tiefe}(\mathcal{C}) = \text{Gegenbeispiel}(\mathcal{C}).$$

(b) Für Konzeptklassen mit endlich vielen Konzepten folgt zusätzlich

$$\begin{aligned} \text{VC}(\mathcal{C}) \leq \text{Tiefe}(\mathcal{C}) &= \text{Gegenbeispiel}(\mathcal{C}) \\ &\leq \text{Gegenbeispiel}_{\text{Halbierung}}(\mathcal{C}) \leq \log_2 |\mathcal{C}| \end{aligned}$$

Die Unterschiede zwischen

- „VC-Dimension und Gegenbeispielzahl“ sowie zwischen
- „Gegenbeispielzahl und der logarithmierten Konzeptzahl“

können unbeschränkt groß werden.

Online- und PAC-Algorithmen

Angenommen, der **Online-Algorithmus** A lernt die Konzeptklasse \mathcal{C} mit Hilfe der Hypothesenklasse \mathcal{H} .

? Können wir A als PAC-Algorithmus auffassen?

? Und wenn ja, wieviele Beispiele werden benötigt?

- Fordere $s = \Theta\left(\frac{1}{\epsilon} \cdot (\text{VC}(\mathcal{H}) \cdot \ln\left(\frac{1}{\epsilon}\right) + \ln\left(\frac{1}{\delta}\right))\right)$ klassifizierte Beispiele an.
 - ▶ Es genügt eine konsistente Hypothese zu bestimmen. Aber wie?
 - ▶ Wende A solange auf die Beispiele an, bis eine konsistente Hypothese gefunden wird!
- Und wenn A (wie später etwa Weighted-Majority) eine Hypothesenklasse \mathcal{H} mit sehr großer VC-Dimension benutzt?
 - ▶ Die notwendige Beispielzahl explodiert: Wir haben ein Problem!

Wir sagen, dass ein Online-Algorithmus A **konservativ** ist, wenn A seine Hypothese nur nach einem Gegenbeispiel wechselt.

- Angenommen, irgendein Online-Algorithmus lernt die Konzeptklasse \mathcal{C} nach höchstens G Gegenbeispielen.
- Dann gibt es einen konservativen Online-Algorithmus, der \mathcal{C} ebenfalls nach höchstens G Gegenbeispielen lernt!

Wir transformieren einen konservativen Online-Algorithmus in einen PAC Algorithmus, der wenige Beispiele benötigt.

A sei ein konservativer Online-Algorithmus, der die Konzeptklasse \mathcal{C} nach weniger als G Gegenbeispielen lernt.

- (1) A klassifiziert mit einer Anfangshypothese h_1 .
- (2) A fordert

$$s = G \cdot \Delta \text{ mit } \Delta = \frac{1}{\epsilon} \ln\left(\frac{1}{\delta}\right)$$

klassifizierte Beispiele x_1, \dots, x_s an.

- ▶ Ist h_1 mit x_1, \dots, x_Δ konsistent, dann heißt h_1 **gut**: A gibt h_1 aus.
 - ★ Sonst gibt A eine neue Hypothese h_2 für das (erste) Gegenbeispiel x_i mit $i \leq \Delta$ aus.
- ▶ Ist h_2 mit $x_{i+1}, \dots, x_{i+\Delta}$ konsistent, dann heißt h_2 gut: A gibt h_2 aus.
 - ★ Sonst gibt A eine neue Hypothese h_3 für das (erste) Gegenbeispiel x_j mit $j \leq i + \Delta$
- ▶ ...

Wird stets eine gute Hypothese gefunden?

- (1) A lernt nach weniger als G Gegenbeispielen:
- ▶ A stellt höchstens G Hypothesen für die $s = G \cdot \Delta$ Beispiele auf.
 - ▶ Mindestens eine Hypothese h_i muss auf den „zugeordneten“ Δ Beispielen konsistent sein und wird ausgegeben.
- (2) Wie groß ist die Wahrscheinlichkeit p , dass h einen Fehler größer als ε besitzt?
- ▶ Wenn h einen großen Fehler hat, dann ist es überraschend, dass h mit sehr vielen (Δ vielen) Beispielen konsistent ist.

$$p \leq (1 - \varepsilon)^\Delta = (1 - \varepsilon)^{\frac{1}{\varepsilon} \cdot \log \frac{1}{\delta}} \leq e^{-\varepsilon \cdot \frac{1}{\varepsilon} \cdot \log \frac{1}{\delta}} = e^{-\log \frac{1}{\delta}} = \delta.$$

Wir haben einen **effizienten** PAC-Algorithmus für

$$s = G \cdot \left(\frac{1}{\varepsilon} \cdot \ln\left(\frac{1}{\delta}\right) \right)$$

Beispiele gefunden, wenn A **effizient** ist.

Verbesserte Analyse: Eine Skizze

- Für eine Hypothese h_i mit Fehler $> \varepsilon$ sollte ein Gegenbeispiel **im Mittel** nach spätestens $\frac{1}{\varepsilon}$ Beispielen auftreten.
 - ▶ A benötigt weniger als G Gegenbeispiele \implies
 - ▶ $O(\frac{G}{\varepsilon} + \Delta)$ Beispiele „sollten“ reichen!

- (a) Wenn ein Online-Algorithmus A für eine Konzeptklasse \mathcal{C} weniger als G Gegenbeispiele benötigt, dann kann A in einen PAC Algorithmus mit

$$O\left(\frac{G}{\varepsilon} + \frac{1}{\varepsilon} \cdot \ln\left(\frac{1}{\delta}\right)\right)$$

Beispielen umgewandelt werden.

- (b) Wenn A effizient ist, dann ist auch der PAC-Algorithmus effizient.

Auswahl von Experten

Die Auswahl von Experten

- ! n Experten geben in mehreren Runden Ja/Nein Empfehlungen ab
wie etwa zum Kauf oder Verkauf einer Aktie
- ! Nach jeder Runde wird mitgeteilt, welche Empfehlung richtig ist.

- ? Gibt es eine Strategie, die fast mit dem besten Experten mithält?
- ? Warum nicht in jeder Runde den bisher besten Experten wählen?

Eine erste Anwendung: Lerne eine unbekannte Threshold-Funktion

$$\text{sign}\left(\sum_{i=1}^n w_i x_i\right)$$

Interpretiere die Variablen x_i als Experten.

Weighted-Majority, die deterministische Variante

Weighted-Majority: Die deterministische Variante

Es gelte $0 < \beta < 1$ für den „Bestrafungsparameter“ β .

1. Setze $w_i = 1$ für alle Experten i .
2. Wenn Experte i die Empfehlung $x_i \in \{\text{Ja}, \text{Nein}\}$ abgibt, dann treffe die Entscheidung „**Ja**“, falls

$$\sum_{i, x_i = \text{Ja}} w_i \geq \sum_{i, x_i = \text{Nein}} w_i$$

und ansonsten treffe die Entscheidung „**Nein**“.

3. Nach Erhalt der richtigen Entscheidung: Hat Experte i eine falsche Empfehlung abgegeben, dann bestrafe ihn mit der Setzung

$$w_i = \beta \cdot w_i.$$

Ansonsten lasse das Gewicht w_i unverändert.

Analyse

- (1) Sei W_t das Gesamtgewicht aller Experten zu Beginn von Runde t . Dann ist anfänglich $W_1 = n$.
- (2) Wenn die Entscheidung in Runde t falsch ist, dann haben sich Experten mit einem Gesamtgewicht von mindestens $W_t/2$ geirrt:

$$W_{t+1} \leq \frac{W_t}{2} + \beta \cdot \frac{W_t}{2} = \frac{1 + \beta}{2} W_t.$$

- (3) Bei f falschen Entscheidungen bis zum Zeitpunkt t , ist

$$W_t \leq n \cdot \left(\frac{1 + \beta}{2}\right)^f.$$

- (4) Wenn der beste Experte i bis zum Zeitpunkt t genau f_{opt} Fehler gemacht hat, dann ist $w_i = \beta^{f_{\text{opt}}}$ und als Konsequenz

$$\beta^{f_{\text{opt}}} \leq W_t \leq n \cdot \left(\frac{1 + \beta}{2}\right)^f.$$

Das Resultat

Es ist $\beta^{f_{\text{opt}}} \leq W_t \leq n \cdot \left(\frac{1+\beta}{2}\right)^f$ und nach Logarithmierung:

$$f_{\text{opt}} \log_2 \beta \leq \log_2 n + f \log_2 \frac{1+\beta}{2}.$$

- (a) Sei f_{opt} die Anzahl der Fehlentscheidungen des besten Experten und f die Anzahl der Fehlentscheidungen von Weighted-Majority.
- (b) Für n Experten und $0 < \beta < 1$ gilt

$$f \leq \frac{f_{\text{opt}} \cdot \log_2 \frac{1}{\beta} + \log_2 n}{\log_2 \frac{2}{1+\beta}}.$$

Nach logarithmischer „Aufwärmzeit“ wird die Prognosekraft des besten Experten erreicht bis auf den Faktor $\log_2\left(\frac{1}{\beta}\right) / \log_2\left(\frac{2}{1+\beta}\right) \geq 2$.

Wir haben die **Potentialmethode** angewandt: Wir haben beobachtet wie schnell sich

das Potential W_t

im Vergleich zum Gewicht des besten Experten verringert.

Weighted-Majority, die randomisierte Variante

Weighted-Majority: Wir würfeln

Bewerte die **komplexe** Empfehlung eines Experten i zum Zeitpunkt t mit der „Note“ c_i^t auf einer Skala von 0 (sehr gut) bis 1 (sehr schlecht).

Unser Algorithmus:

1. Setze $w_i := 1$ für alle Experten i . Setze $t := 1$ und wähle $\varepsilon \in]0, 1/2[$.
2. Wähle den Experten i mit Wahrscheinlichkeit

$$p_i = \frac{w_i}{\sum_{k=1}^n w_k}$$

und übernahm seine Entscheidung.

3. Berechne neue Gewichte: Setze

$$w_i = w_i(1 - \varepsilon \cdot c_i^t).$$

Setze $t := t + 1$.

Kommentar: Beachte $\varepsilon \cdot c_i^t \in [0, 1/2[$.

- (1) w_i^t sei das Gewicht von Experte i zu Beginn von Runde t . Die Gewichtssumme aller Experten vor Runde t ist $W_t = \sum_{i=1}^n w_i^t$.
- (2) $K_t = \sum_{i=1}^n \frac{w_i^t}{W_t} \cdot c_i^t$ ist die erwartete Benotung unserer Entscheidung zum Zeitpunkt t .
- (3) Wie berechnet sich W_{t+1} aus W_t ?

$$\begin{aligned} W_{t+1} &= \sum_{i=1}^n w_i^t \cdot (1 - \varepsilon \cdot c_i^t) = \sum_{i=1}^n w_i^t - \varepsilon \cdot \sum_{i=1}^n w_i^t \cdot c_i^t \\ &= W_t - \varepsilon \cdot K_t \cdot W_t = W_t \cdot (1 - \varepsilon \cdot K_t) \end{aligned}$$

Wir expandieren und erhalten $W_{T+1} = W_1 \cdot \prod_{t=1}^T (1 - \varepsilon \cdot K_t)$.

Es ist $W_{T+1} = W_1 \cdot \prod_{t \leq T} (1 - \varepsilon \cdot K_t)$.

(4) Beachte $\log(1 - x) \leq -x$ und

$$\ln W_{T+1} = \ln W_1 + \sum_{t \leq T} \ln(1 - \varepsilon \cdot K_t) \leq \ln n - \sum_{t \leq T} \varepsilon K_t$$

Erhalten wir schlechte Noten, sinkt das Gesamtgewicht.

(5) Ein Experte mit guter Benotung hingegen stärkt das Gesamtgewicht: Für den Experten i gilt

$$W_{T+1} \geq w_i^{T+1} = \prod_{t \leq T} (1 - \varepsilon \cdot c_i^t).$$

Es ist $W_{T+1} \geq w_i^{T+1} = \prod_{t \leq T} (1 - \varepsilon \cdot c_i^t)$ und
 $\ln(1 - x) \geq -x - x^2$ für $x \in [0, 1/2]$.

(6) Nach Logarithmierung folgt wegen $0 \leq c_i^t \leq 1$:

$$\begin{aligned} \ln W_{T+1} &\geq \sum_{t \leq T} \ln(1 - \varepsilon \cdot c_i^t) \geq - \sum_{t \leq T} (\varepsilon \cdot c_i^t + (\varepsilon \cdot c_i^t)^2) \\ &\geq - \sum_{t \leq T} (\varepsilon \cdot c_i^t + \varepsilon^2 \cdot c_i^t) = -(\varepsilon + \varepsilon^2) \cdot \sum_{t \leq T} c_i^t. \end{aligned}$$

(7) Wir vergleichen die untere mit der oberen Gewichtsschranke:

$$-(\varepsilon + \varepsilon^2) \cdot \sum_{t \leq T} c_i^t \leq \ln W_{T+1} \leq \ln n - \sum_{t \leq T} \varepsilon K_t.$$

Das Ergebnis

- Es ist $-(\varepsilon + \varepsilon^2) \cdot \sum_{t \leq T} c_j^t \leq \ln n - \sum_{t \leq T} \varepsilon K_t$.
- Nach Division durch ε folgt:

Wenn K_{opt} die Gesamtbenotung des besten Experten ist und K_t die erwartete Benotung unserer Entscheidung in Runde t ist, dann folgt

$$\sum_{t \leq T} K_t \leq (1 + \varepsilon) \cdot K_{\text{opt}} + \frac{\ln n}{\varepsilon}.$$

- Wir erreichen die Benotung des besten Experten bis auf den Faktor $1 + \varepsilon$, wenn wir die „Aufwärmzeit“ $\frac{\ln n}{\varepsilon}$ erlauben.

- 1 Im Lernen monotoner Threshold-Funktionen $\text{sign}(\langle w, x \rangle)$ entsprechen die Variablen x_i den Experten.
 - ▶ Siehe später den **Winnow-Algorithmus**.
- 2 „**AdaBoost**“ implementiert Weighted-Majority.
 - ▶ Die Ensemble-Methode baut eine bessere Hypothese aus guten Hypothesen.
- 3 Der **Universal Portfolio Algorithmus** von Cover ist eine weitere Implementierung von Weighted-Majority:
 - ▶ Portfolios, bestehend aus Finanzanlagen, stehen im Wettbewerb miteinander.
 - ▶ Der Markt legt die Gewichte der Portfolios fest.

Constant Rebalanced Portfolios, Der Universal Portfolio Algorithmus

Constant Rebalanced Portfolio: Die CRP-Methode

Ein Vermögen wird über mehrere Anlagezeitpunkte auf m Anlagemöglichkeiten verteilt: Anlage i erhält stets den Anteil p_i .

Zwei Aktien mit $p_1 = p_2 = 0.5$: Die erste Aktie bewegt sich nicht, während sich die zweite Aktie stets erst halbiert und dann verdoppelt.

- Das Vermögen nach einem Anlagezeitpunkt sinkt auf $V \cdot (1/2 + (1/2) \cdot (1/2)) = 3 \cdot V/4$
- und steigt nach dem zweiten Anlagezeitpunkt auf $V \cdot (3/8 + 2 \cdot 3/8) = 9 \cdot V/8$.
- Vor dem Anlagetermin $2n + 1$ ist das Vermögen damit exponentiell auf $(\frac{9}{8})^n \cdot V$ angewachsen!

Der Universal Portfolio Algorithmus

- Welche Verteilung p sollte gewählt werden?
- **Risiko-Minimierung:**
Der Universal Portfolio Algorithmus versucht die **erwartete Vermögensentwicklung** des CRP-Ansatzes zu erreichen.
 - Arbeiten mit N „charakteristischen“ Verteilungen.
 - Zu jedem Anlagezeitpunkt: Rebalanciere ein Portfolio nach seiner Verteilung.
 - Gelder dürfen nicht zwischen Portfolios transferiert werden.

Der Universal Portfolio Algorithmus und Weighted-Majority

- Anfänglich erhält jedes der N Portfolios das Gewicht $\frac{V}{N}$.
- Der Markt aktualisiert die Gewichte.
- Die Gewichte geben den Erfolg wieder.

Wie gut ist der Universal Portfolio Algorithmus?

Es möge m Anlagemöglichkeiten und n Anlageperioden geben.

- opt_n sei das Vermögen des **besten** Portfolios und
- e_n das **erwartete** Vermögen nach n Anlageperioden \implies

$$e_n \geq \frac{opt_n}{(n+1)^{m-1}}.$$

- Für $n \gg m$ ist der durchschnittliche relative Verlustfaktor pro Anlageperiode (gegenüber der optimalen Strategie) höchstens

$$((n+1)^{m-1})^{1/n} \approx (n^{m-1})^{1/n} = n^{(m-1)/n} = 2^{\frac{(m-1) \cdot \log_2 n}{n}} \approx 1.$$

- Wächst die optimale CRP-Strategie um den Faktor a_n^n für $a_n > 1$, dann wächst das erwartete Vermögen um den Faktor b_n^n mit $b_n > 1$ und $\lim_{n \rightarrow \infty} a_n = \lim_{n \rightarrow \infty} b_n$.

Was passiert nach Steuern und Gebühren?

Der Winnow-Algorithmus für Disjunktionen

Der Winnow Algorithmus für monotone Disjunktionen

- (1) Setze $t = \frac{n}{2}$ und $w_1 = \dots = w_n = 1$. Die Mehrheitsfunktion

$$m(x_1, \dots, x_n) = 1 \Leftrightarrow \sum x_i = \sum w_i x_i \geq t = \frac{n}{2}$$

wird als Anfangshypothese verwendet.

- (2) Wiederhole, solange es Gegenbeispiele gibt.

- ▶ Wenn x ein positives Gegenbeispiel ist: Verdopple w_i , falls $x_i = 1$.
/* x_i könnte ein „relevantes Attribut“ sein. */
- ▶ Wenn x ein negatives Gegenbeispiel ist: Setze $w_i = 0$, falls $x_i = 1$.
/* Das nicht relevante Attribut x_i wird eliminiert. */

b_+ und b_- sind die Anzahlen positiver und negativer Gegenbeispiele. Dann ist

$$n + b_+ \cdot t - b_- \cdot t \geq 0.$$

- (1) Bei einem positiven Gegenbeispiel wird das Gesamtgewicht der zu positiven Literalen gehörenden Gewichte verdoppelt.
Aber deren Gesamtgewicht hat t nicht erreicht: Ein positives Gegenbeispiel erhöht das Gesamtgewicht um höchstens t .
- (2) Ein Bestrafungsschritt erniedrigt das Gesamtgewicht um mindestens t .
- (3) Das anfängliche Gesamtgewicht beträgt n und $n + b_+ \cdot t - b_- \cdot t$ ist eine obere Schranke für das aktuelle Gesamtgewicht.
- (4) Die Behauptung folgt: Das Gesamtgewicht ist stets nicht-negativ.

Das Zielkonzept besitze genau k Literale.

Es kann nicht zu viele negative Gegenbeispiele geben.
Aber warum ist auch die Anzahl positiver Gegenbeispiele klein?

- (1) Für jedes Gewicht w_i gilt stets $w_i \leq 2t$: Ein Gewicht w_i mit $w_i \geq t$ nimmt nie an einem Belohnungsschritt teil.
- (2) Nach b_+ positiven Gegenbeispielen gibt es i mit $\log_2 w_i \geq \frac{b_+}{k}$.
 - ▶ Jedes positive Gegenbeispiel verdoppelt das Gewicht von mindestens einem der k Literale des Zielkonzepts.
 - ▶ Kein Literal des Zielkonzepts wird je bestraft:
Es gilt $w_i \geq 2^{b_+/k}$ für mindestens ein Literal x_i des Zielkonzepts.
- (3) Es gibt ein Literal x_i mit $\frac{b_+}{k} \leq \log_2 w_i \leq \log_2 t + 1$.

Was wissen wir?

- ✓ Es ist $n + b_+ \cdot t - b_- \cdot t \geq 0$.
- ✓ Es gibt ein Literal x_i mit $\frac{b_+}{k} \leq \log_2 w_i \leq \log_2 t + 1$.

(1) Also ist

$$b_+ \leq k \cdot (\log_2 t + 1)$$

und es gibt $\leq k \cdot (\log_2 t + 1) = k \cdot \log_2 n$ positive Gegenbeispiele.

(2) Wieviele negative Gegenbeispiele kann es geben? Es ist

$$b_- \leq \frac{n}{t} + b_+ \leq \frac{n}{t} + k \cdot (\log_2 t + 1) = 2 + k \cdot \log_2 n$$

Die Anzahl der Gegenbeispiele ist durch $2 + 2k \cdot \log_2 n$ beschränkt.

Annahme: Alle „ursprünglichen“ Beispiele liegen in $\{0, 1\}^n$.

- (1) Allgemeine Disjunktionen und Monome mit k Literalen können nach höchstens $O(k \cdot \log_2 n)$ Gegenbeispielen gelernt werden.
 - ▶ Durch das Hinzufügen der n neuen Literale x'_1, \dots, x'_n (mit $x'_i = \neg x_i$) werden nicht-monotone Disjunktionen zu monotonen Disjunktionen.
 - ▶ Das Lernen von Monomen ist äquivalent zum Lernen ihrer Komplemente, nämlich der Disjunktionen.

- (2) DNFs mit
 - höchstens k Literalen pro Monom und
 - höchstens s Monomenwerden nach höchstens $O(s \cdot k \cdot \log_2 n)$ Gegenbeispielen gelernt.
 - ▶ Benutze statt Eingabe $x = (x_1, \dots, x_n)$ das Ergebnis von x auf allen $N = O((2n)^k)$ möglichen Monomen mit k Literalen.
 - ▶ Ist das Zielkonzept eine Disjunktion von s Monomen, benötigt Winnow nur $O(s \cdot \log_2((2n)^k)) = O(s \cdot k \cdot \log_2 n)$ Gegenbeispiele.

- Für die Konzeptklasse **k-DNF**_n werden höchstens $O(s \cdot k \log_2 n)$ Gegenbeispielen benötigt, falls das Zielkonzept nur aus s Monomen besteht.
- Allerdings muss die Laufzeit $n^{O(k)}$ investiert werdenen, da alle Monome mit höchstens k Literalen als neue Eingaben auftreten.

Der Versuch einer weitreichenden Verallgemeinerung:

Lerne eine unbekannte, monotone Threshold-Funktion

$$\text{sign}\left(\sum_{i=1}^n w_i x_i\right)$$

mit nicht-negativen Koeffizienten.

Threshold-Funktionen

$$\text{sign}\left(\sum_{i=1}^n w_i x_i + t\right)$$

1. Vektornotation: Für $w = (w_1, \dots, w_n)$, $x = (x_1, \dots, x_n) \in \mathbb{R}^n$ gilt

$$\langle w, x \rangle := \sum_{i=1}^n w_i x_i.$$

2. $H_{w,t} := \{x : \langle w, x \rangle + t = 0\}$ ist eine Hyperebene.

3. Die Vektoren $x, y \in \mathbb{R}^n$ mögen die Länge Eins haben. Dann ist

$$\text{Kosinus}(x, y) = \langle x, y \rangle.$$

4. Eine Konsequenz: Die Ungleichung von Cauchy und Schwartz

$$\langle u, v \rangle \leq \|u\| \cdot \|v\|.$$

5. Der Abstand zwischen $x \in \mathbb{R}^n$ und der Hyperebene $H_{w,t}$

$$= \left| \frac{\langle w, x \rangle + t}{\|w\|} \right|$$

Für $\text{Def} \subseteq \mathbb{R}^n$ ist $f : \text{Def} \rightarrow \{-1, 1\}$ die unbekannte Zielfunktion.

Eine Menge $S \subseteq \mathbb{R}^n$ sei gegeben ebenso wie die Hyperebene

$$H_{w,t} = \{y : \langle w, y \rangle + t = 0\}.$$

(a) Der Margin eines mit b klassifizierten Beispiels $x \in \mathbb{R}^n$ ist

$$\text{Margin}_f(x, w, t) := f(x) \cdot (\langle w, x \rangle + t)$$

► $\text{Margin}_f(x, w, t) \leq 0 \iff \text{sign}(\langle w, x \rangle + t) \neq f(x).$

($H_{w,t}$ klassifiziert Beispiel x richtig $\iff \text{Margin}_f(x, w, t) > 0.$)

(b) Der Abstand zwischen dem Vektor x und der Hyperebene $H_{w,t}$ ist

$$\left| \frac{\text{Margin}_f(x, w, t)}{\|w\|} \right|.$$

(c) Sei $S \subseteq \mathbb{R}^n$ eine Menge von Beispielen. Dann ist

$$\text{Margin}_f(S, w, t) := \inf_{x \in S} \text{Margin}_f(x, w, t).$$

Hyperebene $H_{w,t}$ trennt die positiven von den negativen Beispielen in S , wenn $\text{Margin}_f(S, w, t) > 0$.

(d) Der Margin von S wird definiert durch

$$\text{Margin}_f(S) := \sup_{w,t} \left\{ \frac{\text{Margin}_f(S, w, t)}{\|w\|} : \text{Margin}_f(S, w, t) > 0 \right\}.$$

$\text{Margin}_f(S)$ ist der

größtmögliche minimale Abstand

eines Beispiels $x \in S$ zu einer Hyperebene, die die positiven von den negativen Beispielen trennt.

Der Margin von UND

$$f(x) = 1 \Leftrightarrow x_1 \wedge \dots \wedge x_n \text{ ist wahr.}$$

(1) Setze $w = (1, \dots, 1)$ und $S = \text{Def} = \{0, 1\}^n$:

- ▶ Die Hyperbene

$$\langle w, x \rangle - n + \frac{1}{2}$$

trennt $x = (1, \dots, 1)$ von allen anderen Vektoren in $\{0, 1\}^n$.

- ▶ Für alle $x \in S$ folgt

$$\text{Margin}_{\text{UND}}(x, w, t) \geq \frac{1}{2}$$

- ▶ und als Konsequenz

$$\text{Margin}_{\text{UND}}(S) \geq \frac{1}{2 \cdot \|w\|} = \frac{1}{2 \cdot \sqrt{n}}.$$

(2) Die Und-Funktion hat also einen Margin von mindestens $\frac{1}{2\sqrt{n}}$.

Statt eine allgemeine Threshold-Funktion

$$\text{sign}(\langle x, w \rangle + t)$$

zu lernen, genügt es

- (a) Threshold-Funktionen der Form $\text{sign}(\langle x, u \rangle)$ zu lernen
(Übersetze Beispiele $x \in \mathbb{R}^n$ in geeignete Beispiele $x' \in \mathbb{R}^{n+1}$.)
- (b) **monotone Threshold-Funktionen** $\text{sign}(\langle x, u \rangle)$ zu lernen.
– $\text{sign}(\langle x, u \rangle)$ ist monoton, wenn $u \geq 0$.–
(Übersetze Beispiele $x \in \mathbb{R}^n$ in geeignete Beispiele $x' \in \mathbb{R}^{2n}$.)

Winnow lernt monotone Threshold-Funktionen

- (a) Für monotone Disjunktionen bestraft Winnow irrelevante Attribute ($x_i = 1$ für ein negatives Gegenbeispiel) drakonisch ($w_i = 0$).
 - ▶ Katastrophales Vorgehen, wenn die Zielfunktion sich ein wenig von der Zielfunktion unterscheidet.
- (b) Soll eine monotone Threshold-Funktion gelernt werden, dann gibt es keine Unterscheidung in
 - relevant und irrelevant.
 - ▶ Behandle positive und negative Beispiele auf symmetrische Art und Weise.

Winnow für monotone Threshold-Funktionen

Die unbekannte Threshold-Funktion $f : \text{Def} \rightarrow \{-1, 1\}$ ist zu lernen.

1. Für $w^{(0)} = (\frac{1}{n}, \dots, \frac{1}{n})$ verwende die Hypothese

$$h(x) = \text{sign}(\langle w^{(0)}, x \rangle).$$

η ist eine positive reelle Zahl und $t := 0$.

2. Wiederhole, solange es ein Gegenbeispiel $x^{(t)}$ gibt:

(a) Setze $Z_t = \sum_{i=1}^n w_i^{(t)} \cdot \exp^{\eta \cdot f(x^{(t)}) \cdot x_i^{(t)}}$ und für $i = 1, \dots, n$

$$w_i^{(t+1)} := \frac{w_i^{(t)} \cdot \exp^{\eta \cdot f(x^{(t)}) \cdot x_i^{(t)}}}{Z_t}.$$

/* Gegenbeispiele werden „im Exponenten addiert bzw subtrahiert“. */
/* $w^{(t+1)}$ ist eine Wahrscheinlichkeitsverteilung. */

(b) $h(x) = \text{sign}(\langle w^{(t+1)}, x \rangle)$ ist die neue Hypothese. Setze $t := t + 1$.

Wenn $\text{Def} = \{-1, 1\}^n$.

- 1 Winnow's Hypothesen führen einen (gewichteten) Mehrheitsentscheid durch.
- 2 Winnow multipliziert das Gewicht eines „inkorrekten Bits“ x_i mit \exp^η und das Gewicht eines „korrekten Bits“ mit $\exp^{-\eta}$.
 - ▶ Alternativ: Bestrafe alle inkorrekten Bits mit $\exp^{-2\eta}$.

Winnow = Weighted-Majority für den Beispierraum $\text{Def} = \{-1, 1\}^n$.

Einige Vorbereitungen:

1. Winnow benutzt die **1-Norm**

$$\|x\|_1 = \sum_{i=1}^n |x_i|.$$

Wenn $x \geq 0$, dann ist $\frac{x}{\|x\|_1}$ eine Wahrscheinlichkeitsverteilung.

2. Fordere für alle Beispiele x : $\|x\|_1 \leq R$ mit der **Maximum-Norm**

$$\|x\|_\infty := \max_{i=1}^n |x_i|.$$

3. Wähle einen Gewichtsvektor $v \in \mathbb{R}_{\geq 0}^n$, so dass der „**L₁-Margin**“

$$\min_t \frac{f(x^{(t)}) \cdot \langle v, x^{(t)} \rangle}{\|v\|_1} =: \rho$$

auf den Beispielen $x^{(t)}$ möglichst groß wird.

1. Ein „Potential-Argument“: Das **Potential** zum Zeitpunkt t ist

$$\Phi_t := \sum_{i=1}^N \frac{v_i}{\|v\|_1} \log_2 \frac{v_i / \|v\|_1}{w_i^{(t)}}.$$

- ▶ Φ_t misst die relative Entropie der Verteilung $v / \|v\|_1$ bezüglich $w^{(t)}$.
- ▶ Φ_t fällt, wenn sich $w^{(t)}$ der Verteilung v annähert.

2. Wie stark fällt das Potential, d.h. wie groß ist $\Phi_{t+1} - \Phi_t$?

$$\begin{aligned} \Phi_{t+1} - \Phi_t &= \sum_{i=1}^n \frac{v_i}{\|v\|_1} \log_2 \frac{w_i^t}{w_i^{t+1}} = \sum_{i=1}^n \frac{v_i}{\|v\|_1} \log_2 \frac{Z_t}{\exp^{\eta \cdot f(x^{(t)}) \cdot x_i^{(t)}}} \\ &= \log_2 Z_t - \eta \cdot \sum_{i=1}^n \frac{v_i}{\|v\|_i} \cdot f(x^{(t)}) \cdot x_i^{(t)} \\ &\leq \log_2 \left(\sum_{i=1}^n w_i^{(t)} \cdot \exp^{\eta \cdot f(x^{(t)}) \cdot x_i^{(t)}} \right) - \eta \cdot \rho \end{aligned}$$

$$\begin{aligned}
\Phi_{t+1} - \Phi_t &\leq \log_2 \left(\sum_{i=1}^n w_i^{(t)} \cdot \exp^{\eta \cdot f(x^{(t)}) \cdot x_i^{(t)}} \right) - \eta \cdot \rho \\
&= \log_2 \mathbb{E}_{w^{(t)}} \left[\exp^{\eta \cdot f(x^{(t)}) \cdot x_i^{(t)}} \right] - \eta \cdot \rho \\
&\leq \log_2 \left[\exp^{\eta^2 \cdot (2R)^2 / 8} \right] - \eta \cdot \rho = \eta^2 R^2 / 2 - \eta \cdot \rho.
\end{aligned}$$

3. Die letzte Ungleichung folgt aus Hoeffding's Lemma:

Für $t \geq 0$ ist $\mathbb{E}[\exp^{tX}] \leq \exp^{\frac{t^2(b-a)^2}{8}}$, falls $\mathbb{E}[X] \leq 0$ und $a \leq X \leq b$.

4. Als Konsequenz

$$\Phi_{T-1} - \Phi_0 = \sum_{t=0}^{T-2} \Phi_{t+1} - \Phi_t \leq (T-1) \cdot (\eta^2 R^2 / 2 - \eta \cdot \rho).$$

- Die relative Entropie ist stets nicht negativ $\implies \Phi_{T-1} \geq 0$.
- Wie groß ist das anfängliche Potential?

$$\Phi_0 = \sum_{i=1}^n \frac{v_i}{\|v\|_1} \cdot \log_2 \frac{v_1/\|v\|_1}{1/n} = \log_2 n + \sum_{i=1}^n \frac{v_i}{\|v\|_1} \cdot \log_2 \frac{v_i}{\|v\|_1} \leq \log_2 n.$$

- Also folgt

$$-\log_2 n \leq -\Phi_0 \leq \Phi_{T-1} - \Phi_0 \leq (T-1) \cdot (\eta^2 R^2 / 2 - \eta \cdot \rho).$$

- Wähle $\eta := \rho / R^2 \implies$

$$T^2 \leq \frac{2R^2}{\rho^2} \cdot \log_2 n.$$

Die Threshold-Funktion f sei zu lernen.

- (a) $x^{(0)}, \dots, x^{(T-1)} \in \mathbb{R}^n$ seien Gegenbeispielen für Winnow.
- (b) Es gelte $\|x^{(t)}\|_\infty \leq R$ für $0 \leq t < T$.
- (c) $v \in \mathbb{R}_{\geq 0}^n$ sei ein beliebiger Vektor mit

$$0 < \rho \leq \frac{f(x^{(t)}) \cdot \langle v, x^{(t)} \rangle}{\|v\|_1}.$$

\implies Für $\eta := \frac{\rho}{R^2}$ erhält Winnow höchstens

$$T \leq \frac{2R^2}{\rho^2} \cdot \log_2 n$$

Gegenbeispiele.

Für die Funktion $f : \text{Def} \rightarrow \{-1, 1\}$ ist

$$\text{Margin}_{f,1}(\text{Def}) := \sup_{v \in \mathbb{R}^n} \inf_{x \in \text{Def}} \frac{f(x) \cdot \langle v, x \rangle}{\|v\|_1}.$$

der **L₁-Margin** (von f auf Def).

Es gelte $\|x\|_\infty \leq R$ für alle Vektoren $x \in \text{Def} \implies$

Winnow lernt eine äquivalente Klassifizierung nach höchstens

$$2 \cdot \left(\frac{R}{\text{Margin}_{f,1}(\text{Def})} \right)^2 \cdot \log_2 n$$

Gegenbeispielen.

Winnow: Eine Anwendung

Beispiele müssen aus der Menge $\{0, 1\}^n$ gewählt werden.

Die monotone Disjunktion $\alpha = x_{i_1} \vee \dots \vee x_{i_k}$ sei zu lernen.

1. Für alle $x \in \{0, 1\}^n$ ist

$$\alpha(x) \text{ ist wahr} \iff \langle v, x \rangle + v_{n+1} \geq 0$$

mit $v_{n+1} = 0.5$, ($v_j = 1$ gdw. $j \in \{i_1, \dots, i_k\}$) und $v_j = 0$ sonst.

2. Für $x \in \{0, 1\}^n$ ist $\|x_t\|_\infty \leq 1$.

3. $\|v\|_1 = k + 1/2$ und $\langle v, x \rangle \geq 1/2$ gilt für alle 0-1 Vektoren \implies

▶ $\rho = \frac{1}{2} \cdot \frac{1}{(k+1/2)} = \frac{1}{2k+1}$ und $R = 1$.

$T \leq 2(2k + 1)^2 \cdot \log_2 n$ Gegenbeispiele genügen für das Lernen monotoner Disjunktionen mit k Literalen.

Der Perzeptron-Algorithmus

Der Perzeptron-Algorithmus

Lerne die unbekannte Threshold-Funktion $f : \text{Def} \rightarrow \{-1, 1\}$ mit

$$f(x) = \text{sign}(\langle v, x \rangle).$$

1. $h(x) = \text{sign}(\langle w^{(0)}, x \rangle)$ mit $w^{(0)} = 0$ ist die Anfangshypothese.
Setze $t = 0$.

2. Wiederhole, solange es ein Gegenbeispiel $x^{(t)}$ gibt:

(a) Setze

$$w^{(t+1)} = w^{(t)} + f(x^{(t)}) \cdot x^{(t)}.$$

/* Gegenbeispiele werden addiert bzw subtrahiert. hfill */

(b) Setze $t = t + 1$.

Große Ähnlichkeit zwischen Winnow und Perzeptron:

- Winnow ist ein multiplikatives,
- Perzeptron ein additives Lernverfahren.

Hält der Perzeptron-Algorithmus immer? **Nein!**

(a) Die Threshold-Funktion $\text{sign}(\langle v, x \rangle)$ mit $\|v\| = 1$ und

$$0 < \rho \leq f(x) \cdot \langle v, x \rangle$$

(für alle $x \in \text{Def}$) ist zu lernen.

(b) $x^{(0)}, \dots, x^{(T-1)} \in \mathbb{R}^n$ seien Gegenbeispiele mit $\|x^{(t)}\| \leq R$.

\implies der Perzeptron-Algorithmus benötigt höchstens

$$T \leq \frac{R^2}{\rho^2}$$

Gegenbeispiele.

Analyse: Vollständige Trennbarkeit

Aus $T \cdot \rho \leq \sqrt{T \cdot R^2}$ folgt $T^2 \cdot \rho^2 \leq T \cdot R^2$ und damit die Behauptung.

$$\begin{aligned}
 T \cdot \rho &\leq \left\langle v, \sum_{t=0}^{T-1} f(x^{(t)}) \cdot x^{(t)} \right\rangle \\
 &\leq \left\| \sum_{t=0}^{T-1} f(x^{(t)}) \cdot x^{(t)} \right\| && \|v\| = 1, \langle a, b \rangle \leq \|a\| \cdot \|b\| \\
 &= \left\| \sum_{t=0}^{T-1} (w^{(t+1)} - w^{(t)}) \right\| = \|w^{(T)}\| \\
 &= \sqrt{\sum_{t=0}^{T-1} \left(\|w^{(t+1)}\|^2 - \|w^{(t)}\|^2 \right)} && \|w^{(0)}\| = 0
 \end{aligned}$$

$$\begin{aligned}
T \cdot \rho &\leq \sqrt{\sum_{t=0}^{T-1} \left(\|w^{(t+1)}\|^2 - \|w^{(t)}\|^2 \right)} \\
&= \sqrt{\sum_{t=0}^{T-1} \left(\|w^{(t)} + f(x^{(t)}) \cdot x^{(t)}\|^2 - \|w^{(t)}\|^2 \right)} \\
&= \sqrt{\sum_{t=0}^{T-1} \left(\underbrace{2f(x^{(t)}) \cdot \langle x^{(t)}, w^{(t)} \rangle}_{\leq 0} + \|x^{(t)}\|^2 \right)} \\
&\leq \sqrt{\sum_{t=0}^{T-1} \|x^{(t)}\|^2} = \sqrt{T \cdot R^2} \quad \checkmark
\end{aligned}$$

Sei $f : \text{Def} \rightarrow \{-1, 1\}$ eine unbekannte Threshold-Funktion.

Es gelte $\|x\| \leq R$ für alle Vektoren $x \in \text{Def} \implies$
der Perzeptron-Algorithmus lernt eine äquivalente Klassifizierung nach
höchstens

$$\left(\frac{R}{\text{Margin}_f(\text{Def})} \right)^2$$

Gegenbeispielen.

Die Anzahl benötigter Gegenbeispiele hängt nicht von der Dimension, sondern nur vom Margin und der maximalen Beispielnorm ab.

Analyse: Partielle Trennbarkeit

Positive und negative Beispiele sind nicht mehr linear trennbar!

- Die Funktion $f : \text{Def} \rightarrow \{-1, 1\}$ ist keine Threshold-Funktion:
Lerne f mit Threshold-Funktionen.
- Holt Perzeptron das Mögliche heraus?
 - ▶ Vergleiche Perzeptron mit einer Threshold-Funktion $\text{sign}(\langle v, x \rangle)$ für eine Folge $x^{(0)}, \dots, x^{(T-1)}$ von **Beispielen**.
 - ★ Die Folge besteht nicht nur aus Gegenbeispielen.
 - ★ $\text{sign}(\langle v, x \rangle)$ ist optimal auf die Folge eingestellt.
- Wieviele **Aktualisierungen** benötigt Perzeptron?

Partielle Trennbarkeit: Das Resultat

Die Funktion $f : \text{Def} \rightarrow \{-1, 1\}$ ist gegeben.

- (a) $x^{(0)}, \dots, x^{(T-1)} \in \mathbb{R}^n$ seien Beispiele mit $\|x^{(t)}\| \leq R$.
- (b) Sei $v \in \mathbb{R}^n$ ein beliebiger Vektor mit $\|v\| = 1$ und $\rho \in \mathbb{R}_{\geq 0}$ sei beliebig. Es gelte

$$d_t := \max\{0, \rho - f(x_t) \cdot \langle v, x^{(t)} \rangle\} \text{ sowie } d = (d_t \mid 0 \leq t \leq T-1).$$

\implies der Perzeptron-Algorithmus führt höchstens

$$\left(\frac{R + \|d\|}{\rho} \right)^2$$

Aktualisierungen für die Beispielfolge durch.

Ist das gut?

Perzeptron führt höchstens $\left(\frac{R + \|d\|}{\rho}\right)^2$ Aktualisierungen durch, wobei

$$d_t := \max\{0, \rho - f(x_t) \cdot \langle v, x^{(t)} \rangle\} \text{ sowie } d = (d_t \mid 0 \leq t \leq T - 1).$$

Wenn $(\|d\|/\rho)^2 = O(\text{Klassifikationsfehler von } \text{sign}(\langle v, x \rangle))$, dann ist

$$\left(\frac{R + \|d\|}{\rho}\right)^2 \approx \frac{R^2}{\rho^2} + \frac{\|d\|^2}{\rho^2} \approx \frac{R^2}{\rho^2} + \text{Klassifikationsfehler von } v$$

Perzeptron erreicht $\text{sign}(\langle v, x \rangle)$ nach „Aufwärmphase“ der Länge $\frac{R^2}{\rho^2}$.

Die Bestimmung von Hyperebenen, die nur eine (fast-)minimale Anzahl von Beispielen falsch klassifizieren, führt auf ein NP-hartes Problem.

- Keine Garantie für eine gute Approximation.

Stelle lineare Trennbarkeit her.

Wir transformieren die Beispiele:

$$\bar{x}^{(t)} := \left(x_1^{(t)}, \dots, x_n^{(t)}, 0, \dots, 0, \dots, 0, \underbrace{\Delta}_{\text{Komponente } n+t}, 0, \dots, 0 \right).$$

Der Parameter Δ wird später gesetzt.

Behauptung 1: Perzeptron klassifiziert $\bar{x}^{(t)}$ genauso wie $x^{(t)}$.

1. Perzeptron summiert bzw subtrahiert Gegenbeispiele.
2. Beachte

$$\langle w^{(t)}, x^{(t)} \rangle = \langle \bar{w}^{(t)}, \bar{x}^{(t)} \rangle$$

Wir übersetzen auch v in $\bar{v}/\|\bar{v}\|$ mit

$$\bar{v} = \left(v_1, \dots, v_n, \frac{f(x^{(0)}) \cdot d_0}{\Delta}, \dots, \frac{f(x^{(T-1)}) \cdot d_{T-1}}{\Delta} \right).$$

Es ist $\|v\| = 1$ und deshalb folgt $\|\bar{v}\| = \sqrt{1 + \frac{\|d\|^2}{\Delta^2}}$.

$$\begin{aligned} f(x^{(t)}) \cdot \frac{\langle \bar{v}, \bar{x}^{(t)} \rangle}{\|\bar{v}\|} &= f(x^{(t)}) \cdot \left(\frac{\langle v, x^{(t)} \rangle}{\|\bar{v}\|} + \Delta \cdot \frac{f(x^{(t)}) \cdot d_t}{\|\bar{v}\| \cdot \Delta} \right) \\ &= f(x^{(t)}) \cdot \frac{\langle v, x^{(t)} \rangle}{\|\bar{v}\|} + \frac{d_t}{\|\bar{v}\|} \\ &\geq f(x^{(t)}) \cdot \frac{\langle v, x^{(t)} \rangle}{\|\bar{v}\|} + \frac{\rho - f(x^{(t)}) \cdot \langle v, x^{(t)} \rangle}{\|\bar{v}\|} \\ &= \frac{\rho}{\|\bar{v}\|} \end{aligned}$$

Die Beispiele $\bar{x}^{(0)}, \dots, \bar{x}^{(T-1)}$ sind linear trennbar mit Margin $\rho/\|\bar{v}\|$.

1. Es ist $\|\bar{x}^{(t)}\|^2 \leq R^2 + \Delta^2$.
2. Die Anzahl der Aktualisierungen ist beschränkt durch

$$\frac{(R^2 + \Delta^2) \cdot \|\bar{v}\|^2}{\rho^2} = \frac{(R^2 + \Delta^2) \cdot (1 + \frac{\|d\|^2}{\Delta^2})}{\rho^2}.$$

3. Wähle $\Delta^2 := R \cdot \|d\|$ ✓

Winnow versus Perzeptron

VC-Dimension und Margin

$$\text{LINEAR}_\rho(\text{Def}) := \{ f(*) = \text{sign}(\langle w, * \rangle) : \text{Margin}_f(\text{Def}) \leq \rho, w \in \mathbb{R}^n \}$$

ist die Konzeptklasse der Threshold-Funktionen $f : \text{Def} \rightarrow \{-1, 1\}$ mit $\text{Margin} \geq \rho$ für die Teilmenge $\text{Def} \subseteq \mathbb{R}^n$. Dann folgt

Es gelte $\|x\|_2 \leq R$ für alle $x \in \text{Def}$

$$\text{VC}(\text{LINEAR}_\rho(\text{Def})) \leq \min\left\{ \frac{R^2}{\rho^2}, n + 1 \right\}.$$

(Beweis: Satz 4.2 im Textbuch „Foundations of Machine Learning“)

Wenn $\frac{R^2}{\rho^2} = o(n)$, dann hängt die VC-Dimension

- von der 1-Norm der Beispiele und dem Margin ρ ab,
- nicht aber von der Dimension n .

Wann ist der Perzeptron-Algorithmus gut?

- (a) Perzeptron ist fast-optimal, wenn $VC(\text{LINEAR}_\rho(\text{Def})) \approx \frac{R^2}{\rho^2}$.
- ▶ Insbesondere ist R^2 wie auch ρ^2 höchstens polynomiell in n .
 - ▶ In allen anderen Fällen: Benutze die lineare Programmierung.
- (b) Ein Vergleich mit dem Halbierungs-Algorithmus für den Beispielraum $X = \{0, 1\}^n$:
- ▶ Man kann zeigen, dass es $2^{\Theta(n^2)}$ viele verschiedene Threshold-Funktionen mit ganzzahligen Koeffizienten gibt.
 - ★ Die überwältigende Mehrheit dieser Threshold-Funktionen besitzt somit Koeffizienten, die im Absolutbetrag exponentiell groß sind.
 - ★ Der Perzeptron-Algorithmus addiert bzw. subtrahiert Beispiele \implies die meisten Threshold-Funktionen erfordern $2^{\Omega(n)}$ Gegenbeispiele!
 - ▶ Der Halbierungs-Algorithmus lernt jede der $2^{\Theta(n^2)}$ Threshold-Funktionen „locker“ nach $O(n^2)$ Gegenbeispielen.
 - ★ Die Auswertung seiner Hypothesen ist allerdings sehr aufwändig.

Winnow versus Perzeptron

$\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\|x\|_\infty$ und $\|w\|_2 \leq \|w\|_1 \leq \sqrt{n}\|w\|_2 \implies$
Die Gegenbeispielzahlen unterscheiden sich maximal um Faktor n .

- (a) Winnow reagiert gutmütiger auf Beispiele mit vielen großen Komponenten:
- ▶ Für Disjunktionen mit k (aus n möglichen Variablen)
 - ★ genügen $O(k \cdot \log_2^2 n)$ Gegenbeispiele für Winnow und
 - ★ sind bis zu $\Omega(n)$ Gegenbeispiele für Perzeptron erforderlich.
- (b) Wer mag die Zielfunktion $\text{sign}(\langle w, x \rangle)$?
- ▶ Winnow mag es, wenn w nur wenige große Komponenten besitzt,
 - ▶ Perzeptron mag Gewichtsvektoren mit vielen ungefähr gleichgroßen Komponenten.

Zusammenfassung

Online-Algorithmen: Zusammenfassung

(a) Gegenbeispielzahl:

$$\begin{aligned} VC(\mathcal{C}) \leq \text{Tiefe}(\mathcal{C}) &= \text{Gegenbeispiel}(\mathcal{C}) \\ &\leq \text{Gegenbeispiel}_{\text{Halbierung}}(\mathcal{C}) \leq \log_2 |\mathcal{C}| \end{aligned}$$

(b) Effiziente Online-Algorithmen „liefern“ effiziente PAC-Algorithmen
▶ Keine übertriebenen Erwartungen an Online-Algorithmen.

(c) **Weighted-Majority**: Für die Gesamtbenotung K ist

$$K \leq (1 + \varepsilon) \cdot K_{\text{opt}} + \frac{\ln n}{\varepsilon}$$

▶ Anwendungen: Winnow, AdaBoost, Universal Portfolio Algorithmus.

(d) **Winnow** und **Perzeptron**: $\leq \frac{R^2}{\text{Margin}_f^2(\text{Def})}$ Gegenbeispiele (bis auf logarithmische Faktoren)

- ▶ Winnow passt sich der Komplexität der Zielfunktion an.
- ▶ Beide, Winnow und Perzeptron sind fehler-tolerant.

Pleiten, Pech, Pannen und geniale Ideen

- 1943 McCulloch und Pitts führen Threshold-Funktionen ein, um Neuronen zu modellieren.
- 1958 Rosenblatt stellt den Perzeptron-Algorithmus vor.
- 1969 Minsky und Pappert dämpfen die Euphorie: Das XOR ist nicht durch eine Threshold-Funktion darstellbar.
- 1960 In der Steuerungstheorie wird die Methode „Backpropagation“ vorgeschlagen.
- 1986 Rumelhart, Hinton und Williams wenden Backpropagation auf neuronale Netzwerke an.
- 1995 Einfachere Methoden (Support-Vektor Maschinen) laufen den neuronalen Netzwerken den Rang ab.
- 2009 Deep Learning ist erfolgreich: Verbesserte Hardware, mehr und bessere Beispiele, verbesserte Software.

Und die Zukunft?