Machine Learning 1

Henrik Tscherny

27. Februar 2022

Inhaltsverzeichnis

1 Supervised learning

Ziel: Finde in einer Funktionsfamilie, z.B. lineare Funktionen, eine Funktion z.B. $f = 5x + 3$ welche eine gewichtete Summe über input-output-Paaren minimiert, dabei sollte die Funktion die gegebenen Daten bestmöglich beschreiben ohne dabei zu komplex zu sein

Beispiel:

- $g : \Theta \to Y^X$ -> Ordnet jeder Messung zu ob gesund oder krank
  $g_\theta(x_s) = \begin{cases} 0, & \sum_{j=1}^{n} \Theta_j x_j > 0 \\ 1, & sonnst \end{cases}$, für ein Sample $x_s \in \mathbb{R}^m$
  - $x \in X = \mathbb{R}^n$: Inputs -> Messung mit m Messpunkten
  - $y \in Y = \{0, 1\}$: Labels -> gesund = 1, krank = 0
  - $\theta \in \Theta = \mathbb{R}^n$: Parameter -> z.B. Koeffizienten einer linearen Funktion

Die Qualität einer gelernten Funktion kann u.a. durch folgende Größen beschrieben werden:

- **Accuracy** (Verhältnis der korrekten Voraussagen zur Gesamtanzahl)
  $\frac{L(0,0)+L(1,1)}{|S|}$
- **Error ratio** (Verhältnis der falschen Voraussagen zu Gesamtanzahl)
  $\frac{L(0,1)+L(0,1)}{|S|}$
- **Precision** (Von den positiv getesteten, wie viele sind tatsächlich positiv) False-Positive-Rate
  $\frac{L(1,1)}{L(1,0)+L(1,1)}$
- **Recall/Sensitivity** (Von den tatsächlich positiven, wie viele wurden tatsächlich positiv getestet) False-Negative-Rate
  $\frac{L(1,1)}{L(0,1)+L(1,1)}$

1.1 Formal

Optimieren einer Funktionsfamilie $g : \Theta \to \{0,1\}^X$, damit dies einfacher ist, optimieren einer Relaxation $f : \Theta \to \mathbb{R}^X$. Sei L eine Loss-function $L : \mathbb{R} \times \{0,1\} \to \mathbb{R}_0^+$ welche g im Bezug zu f definiert.
$\forall \theta \in \Theta \forall x \in X : g_\theta(x) \in \underset{\hat{y} \in \{0,1\}}{\operatorname{argmin}} L(f_\theta(x), \hat{y})$
Bsp. Loss-function:

0-1-Loss: $L = \begin{cases} 0, & sample = label \\ 1, & sonnst \end{cases}$, der Loss ist 0% wenn das Label stimmt

Definiere des Weiteren:

- S: Samples
- X: Attributspace
- $x : S \to X$: bildet ein konkretes Sample mit seinen Attributen ab
- $y : S \to \{0,1\}$ gibt einem konkreten Sample ein Label

Das Tupel $T = (S, X, x)$ nennt man **unlabled data**
Das Tupel $T = (S, X, x, y)$ nennt man **labeled data**

Damit die Komplexität der zu lernenden Funktion begrenzt wird, führen wir zusätzlich noch einen **Regularizer**. Komplexität kann in diesem Fall z.B. die Anzahl der Koeffizienten oder die Länge einer Formel bemessen werden. Der Einfluss des Regularizers wird durch einen Parameter λ gesteuert
$R : \Theta \to \mathbb{R}_0^+$ und $\lambda \in \mathbb{R}_0^+$
Das **supervised learning problem** kann dann wie folgt formuliert werden:

$$\inf_{\theta \in \Theta} \lambda R(\theta) + \frac{1}{|S|} \sum_{s \in S} L(f_\theta(x_s), y_s)$$

- der Regularizer wird durch λ gewichtet
- es wird die Summe der individuellen Loss-Werte minimiert
- die Loss-Summe wird über die Anzahl der Samples Normalisiert, das macht man, damit man lediglich die Parameter übertragen muss, wenn man das Model weitergibt
- Da der Regularizer zum Gesamtloss addiert wird, wird versucht diesen Term ebenfalls möglichst klein zu halten

Das **separation problem** ist definiert durch:

$\inf_{\theta \in \Theta} R(\theta)$
$\forall s \in S : f_\theta(x_s) = y_s$

- finden des minimalen Regularizers
- alle Daten sind korrekt gelabeled

Das **bounded separability problem** lautet:

$R(\theta) \leq m$
$\forall s \in S : f_\theta(x_s) = y_s$

- finden eines Regularizers welcher die Komplexität für jeden Parameter unter einer Schranke m hält

Das **inference problem** (Anwenden des trainierten Modells) kann nun wie folgt formuliert werden:

$$\min_{y' \in \{0,1\}^S} \sum_{s \in S} L(\hat{f}(x_s), y'_s) = \sum_{s \in S} \min_{y' \in \{0,1\}^S} L(\hat{f}(x_s), y'_s)$$

2 Disjunktive Normalformen (DNF'S)

Probleme können ebenfalls als logische Gleichungen interpretiert werden, diese Gleichungen können dann in eine DNF (Mit oder verbundene und-Terme) umgeformt werden.

2.1 Formal

- $\Gamma = \{(V_0, V_1) \in 2^V \times 2^V | V_0 \cap V_1 = \emptyset\}$
  Jede Variable kann entweder negiert oder nicht-negiert vorkommen
- $\Theta = 2^\Gamma$
- $\forall x \in \{0,1\}^V : f_\theta(x) = \bigwedge_{(V_0,V_1) \in \theta} \prod_{v \in V_0} (1 - x_v) \prod_{v \in V_1} x_v$
  Definition einer DNF, veroderte negierte und nicht-negierte Variablen

Beispiel $\{(\emptyset, \{v_1, v_2\}), (\{v_1\}\{v_3\})\} = \theta \in \Theta \rightarrow f_\theta(x) = x_{v_1} x_{v_2} \vee (1 - x_{v_1}) x_{v_3}$

Des Weiteren können Regularizer für DNF's definiert wenn, um deren Komplexität zu bemessen:

- $R_d(\theta) = \max_{(V_0,V_1)\in\theta} (|V_0| + |V_1|)$
  Tiefe der Formel, e.g. Anzahl der Variablen des längsten und-Terms
- $R_l(\theta) = \sum_{(V_0,V_1)\in\theta} (|V_0| + |V_1|)$
  Länge der Formel, e.g. Gesamtanzahl der Variablen in der Gesamtformel (auch doppelte zählen)

Beispiel $\theta = \{(\emptyset, \{0\}), (\{0\}, \{3\}), (\{0,3\}, \{1,2\}))\}$
$\rightarrow f_\theta(x) = x_0 \vee (1 - x_0)x_3 \vee (1 - x_0)(1 - x_3)x_1x_2 \rightarrow R_l(\theta) = 7,\ R_d(\theta) = 4$

Das **Supervised learning problem of DNF's** kann wie folgt formuliert werden:

$\min_{\theta\in\Theta} R(\theta)$
$\forall s \in S : f_\theta(x_s) = y_s$

- Der Unterschied ist lediglich, dass ein min statt eines inf benutzt wird
- min: kleinstes Element einer Menge und muss in der Menge selbst liegen
- inf: größte untere Schranke, muss nicht in der Menge selbst liegen, es müssen nur alle Elemente kleiner sein Des Weiteren erhält man das **bounded depth/length DNF Problem** indem man den Regularizer aus dem bounded sparability problem mit R_d / R_l austauscht

Das bounded length/depth DNF problem ist NP-hard
Beweis durch Reduktion des set cover problems auf das bounded length/depth DNF problem (Haussler):
Was ist ein Set-Cover:

- Sei S eine eine Menge
- Sei $\Sigma \subseteq 2^S,\ \emptyset \notin \Sigma$ ein Cover, gdw. $\bigcup_{U\in\Sigma} U = S$
- $m \in \mathbb{N}$
- Die Entscheidung ob ein $\Sigma' \subseteq \Sigma$ existiert, s.d. $|\Sigma'| \leq m$ nennt man das **set cover problem** (S, Σ, m)

Beispiel:
Sei $S = \{1,2,3,4,5,6\}$, dann ist ein mögliches Cover $\Sigma = \{\{1,2,3\}, \{4,5,6\}\}$ (m=2)

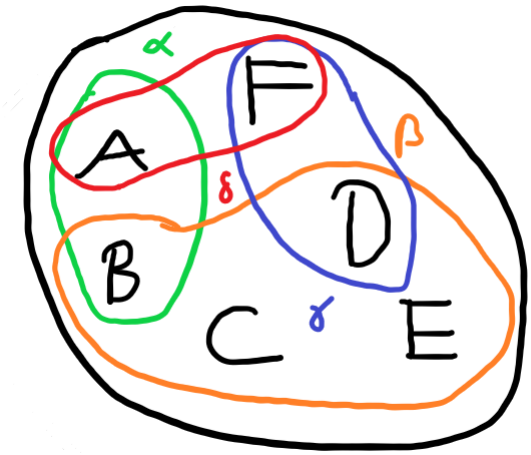
2.2 Beweis der NP-hardness von set-cover

Wir zeigen das set-cover NP-hard ist indem wir es auf bounded length/depth DNF reduzieren (Haussler)

- Sei (S', Σ, m) eine Instanz von set-cover
- Definiere nun **Haussler data** (S, X, x, y) s.d.:
  - $S = S' \cup \{1\}$, wir fügen S ein spezielles distinktes Element 1 hinzu
  - $X = \{0,1\}^{\Sigma}$
  - wir definieren uns eine Funktion welche angibt ob ein Element in einer Menge vorkommt wie folgt:
    $\forall s \in S' \, \forall \sigma \in \Sigma : \; x_s(\sigma) = \begin{cases} 0, & s \in \sigma \\ 1, & \text{otherwise} \end{cases}$
    Beispiel:
    - $S = S' \cup \{1\} = \{2,3\} \cup \{1\} = \{1,2,3\}$
    - $\Sigma = \{\{2\}, ..., \{1,2\}, ..., \{2,3\}\}$
    - $x_2(\{2,3\}) = 0, \; x_3(\{2\}) = 1$
  - $x_1 = 1^{\Sigma}$, das spezielle Element kommt nirgends vor
  - $y_1 = 1$ und $\forall s \in S' : \; y_s = 0$, wir definieren das label das spezielle Element 1, für alle anderen 0
- z.z. Lemma: $\bigcup_{\sigma \in \Sigma'} = S' \Leftrightarrow \forall s \in S' : \prod_{\sigma \in \Sigma'} x_s(\sigma) = 0$
  set-cover kann umgeschrieben werden in ein Produkt mittel der Funktion x_s (das Produkt verhält sich wie ein logisches UND)
  - $\bigcup_{\sigma \in \Sigma'} = S'$
  - $\Leftrightarrow \forall s \in S' \, \exists \sigma \in \Sigma' : \; s \in \sigma$, für jedes Sample ex. eine TM mit diesem Sample
  - $\Leftrightarrow$ für dieses Sample gilt somit $x_s(\sigma) = 0$, d.h. es ex. ein Sample für die die Funktion 0 ist
  - $\Leftrightarrow$ Existenz kann mittel des logischen UND's repräsentiert werden $\forall s \in \Sigma' : \; \prod_{\sigma \in \Sigma'} x_s(\sigma) = 0$
- **Beweis NP-hardness**:
  z.z.: $\exists \, \Sigma' \subseteq \Sigma$ von S' mit $|\Sigma'| \leq m \Leftrightarrow \exists \, \theta \in \Theta : \; R(\theta) \leq m$ und $\forall s \in S : \; f_\theta(x_s) = y_s$

d.h., es ex. ein Lösung von set-cover mit bound m gdw. es Parameter θ mit Komplexität $\leq$ m gibt und alle Samples korrekt inferred werden

- ($\Rightarrow$) wenn Set-Cover eine Lösung hat, dann gibt es eine DNF dazu
  - Sei $\Sigma' \subseteq \Sigma$ ein Cover von S mit $\Sigma'| \leq m$
  - Sei $V_0 = \emptyset$, $V_1 = \Sigma'$, d.h wir definieren das Cover als Menge der nicht-negierten Variablen einer DNF
  - $\forall x' \in X : \ f_\theta(x') = \prod_{\sigma \in \Sigma'} x'(\sigma)$, (DNF mit nur positiven Variablen)
  - siehe Lemma muss dieses Produkt gleich 0 sein $\forall s \in S' : \ f(x_s) = 0$
  - Laut Definition gilt zudem $f(1^\Sigma) = 1$
  - daraus folgt, dass alle Daten richtig gelabelt wurden $\forall s \in S' : f(x_s) = y_s$
  - Da wir $V_1 = \Sigma'$ und $V_0 = \emptyset$ gesetzt haben, ist der Regularizer auch gleich $R(\theta) = |\Sigma'| \leq m$
- ($\Leftarrow$) wenn DNF gegeben wie lautet das Set-Cover
  - Sei $\theta \in \Theta$ s.d. alle Daten richtig inferred werden und $R(\theta) \leq m$
  - TODO: verstehe Beh. am Anfang
  - TODO: what is going on here ?



Erklärung v2

Wir wollen zeigen die Np-hardness von mDNF indem wir zeigen, dass es ein Cover von S mit Bound m gibt, gdw. es eine m bounded DNF gibt, welche die Elemente korrekt labelt

- $\Rightarrow$ (wenn Set-Cover eine Lösung hat, dann gibt es eine DNF dazu)
  - Sei ein Cover $\Sigma' = \{\alpha, \beta, \gamma\}$ mit $m = 3$ und $S = \{A, B, C, D, E, F\}$ gegeben, wie lautet eine DNF, s.d. $f(x_s) = 0$?
  - Existiert für jedes Element aus S ein Cover aus Σ' s.d. das Element aus S in einem Cover von Σ' liegt ?
  - z.B. liegt A in α oder in β oder in γ ?
  - für die Abfrage benutzen wir die durch die Haussler-Data definierte Funktion $x_s(\sigma)$, JA Antwort = 0
  - Darum können wir ein Produkt benutzten um ein logisches oder zu simulieren
  - $f_\theta(x_s) = \prod_{\sigma \in \Sigma'} x_s(\sigma)$
  - da wir ein Cover für S gegeben haben, muss es für jedes $s \in S$ auch eine Menge in Σ' geben, s.d. $s \in \sigma$, d.h. die Funktion ist 0 für alle Sample
  - nur das spezielle Element liegt nirgendwo drin (per Definition), d.h. es ist die einzige zu mappende NEIN Instanz des Problems
  - $\rightarrow$ Die angegebene Funktion labelt alle Elemente (ob sie covert sind oder nicht) richtig

  $\Leftarrow$ (wenn DNF, dann Set-Cover)
  - Die von Haussler Definierte Funktion x_s muss für alle Elemente des Covers 0 sein (per Definition)
  - Nehmen wir nun alle Elemente für die die DNF, 0 ergibt und vereinigen diese, so erhalten wir ein valides Set-Cover für S
  - das spezielle Element wird also nicht in das Cover aufgenommen, so wie es sein soll per Definition

3 BTD's

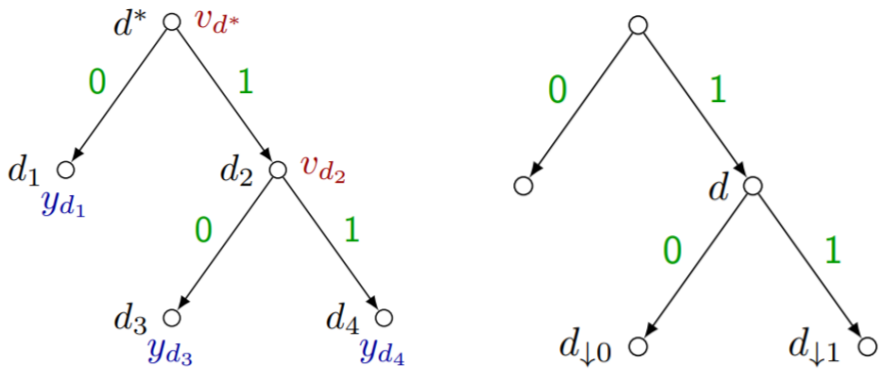
3.1 Formal

Ein V-variant BTD kann als Tupel $\theta = (V, Y, D, D', d\ *, E, \delta, v, y)$ wie folgt dargestellt werden

- eine Menge von Variablen: $V \neq \emptyset$

- eine Menge von Werten: $Y \neq \emptyset$
- einem (Sub)Tree $(D \cup D', E)$
  - eine Menge von Inner-Nodes: D
  - eine Menge von Leaves: D'
  - ein Wurzelknoten: d^*
  - Kanten: E
- Kantenfunktion: $\delta : E \to \{0, 1\}$
  - Jede Inner-Node ($d \in D$) hat genau 2 ausgehende Kanten
  - $e = (d, d')$ mit $\delta(e) = 0$
  - $e' = (d, d'')$ mit $\delta(e') = 1$
- Variablenfunktion: $v : D \to V$, Weißt Inner-Nodes eine Variable zu
- Wertefunktion: $y : D' \to Y$, Weißt Blättern einen Wert zu

Des Weiteren definieren wir noch den **Nachfolgeknoten von d**, wenn wir den **Weg j** nehmen als $\mathbf{d_{\downarrow j}}$



(a) BTD (b) $d_{\downarrow}$

Für den Subtree von θ mit der Wurzel d schreibt man $\theta[d]$
Jeder Subtree von θ mit Wurzel d ist selbst wieder ein V-variante Y-valued BTD
Für jeden BTD θ lässt sich ebenfalls wieder eine Funktion f_θ wie folgt definieren:

$$f_\theta : \{0,1\}^V \to Y, \quad \forall x \in \{0,1\}^V \; f_\theta(x) = \begin{cases} y(d^*), & D = \emptyset \\ f_{\theta[d^*_{\downarrow 0}]}(x), & D \neq \emptyset \wedge x_v(d^*) = 0 \\ f_{\theta[d^*_{\downarrow 1}]}(x), & D \neq \emptyset \wedge x_v(d^*) = 1 \end{cases}$$

Diese Funktion löst den BTD nacheinander rekursiv auf und betrachtet dabei in jeder Iteration einen kleineren Sub-BTD, bis dieser nur noch aus einen einzelnen Knoten besteht

- ist der BTD leer, d.h. es gibt nur eine Node, d.h. die Wurzel ist zugleich ein Leave, dann nehme den Wert davon, Ende der Rekursion
- ist der Baum nicht leer, dann überprüfe die Variable an der Wurzel $x_v(d^*)$ und dann:
  - nimm den linken Sub-BTD (0-Pfad) falls $x_v = 0$
  - nimm den rechten Sub-BTD (1-Pfad) falls $x_v = 1$
- Fahre mit dem Sub-BTD rekursiv weiter fort

Wie auch Für DNF's kann für BTD's ein Regularizer angegeben werden, dieser gibt die **Tiefe** des BTD's an und ist ebenfalls rekursiv definiert:

$$R(\theta) = \begin{cases} 0, & D = \emptyset \\ 1 + \max\{R(\theta[d^*_{\downarrow 0}]), R(\theta[d^*_{\downarrow 1}])\}, & \text{else} \end{cases}$$

Quasi, **Wie lang ist der längste Pfad im Baum, beginnend von der Wurzel**

Ebenfalls kann für BTD's analog das supervised learning bzw. das bounded depth BTD problem definiert werden

3.2 NP-hard Beweis

Dazu reduzieren wir das das NP-schwere **exact cover by 3-sets** Problem auf das **bounded depth BTD** Problem

Ein **exact cover** ist ein cover bei welchem alle Elemente paarweise disjunkt sind

exact cover by 3-sets problem:

- Sei S eine Menge
- Sei $\Sigma \subseteq 2^S \setminus \{\emptyset\}$
- Lässt sich ein $\Sigma' \subseteq \Sigma$ finden, s.d. es nur Mengen der Größe 3 gibt, welche S exakt überdecken (dazu muss S natürlich ein vielfaches von 3 Elemente enthalten) Warum müssen es 3 sein geht doch auch mit einer anderen Zahl ?

Definitionen:

- Sei (S', Σ) eine Instanz des exact cover by 3-sets problem

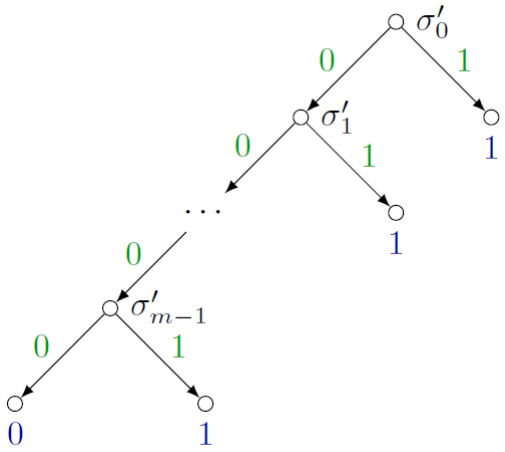
- Sei $|S'| = 3n$ mit $n \in \mathbb{N}$
- Wir konstruieren eine Instanz des m-bounded depth BTD Problems wie folgt:
  - Sei $V = \Sigma$, die Variablen des BTD's sind die möglichen Mengen welche zum Überdecken benutzt werden können
  - $S = S' \cup \{0\}$, wir fügen ein distinktes Element 0 hinzu
  - Sei $x : S \to \{0,1\}^{\Sigma}$, eine Funktion welche angibt ob ein Element in einer Menge vorkommt (analog wie bei DNF, nur invertiert)
  - Das spezielle Element 0 kommt dabei nirgends vor, d.h. $x_0 = 0$
  - $y : S \to \{0,1\}$, weißt dem speziellen Element 0 das Label 0 zu, allen anderen das Label 1, d.h. $y_0 = 0$
  - $m = n$

Wir zeigen nun, dass es ein exact cover gibt gdw. das bounded BTD problem eine Lösung hat Beweis:

- ($\Rightarrow$)
  - Sei $\Sigma' \subseteq \Sigma$ eine Lösung des exact cover Problems
  - Wir definieren uns eine beliebige Ordnung der TM's des Covers und ordnen sie demzufolge in einem BTD untereinander im 0-Pfad an $\sigma' : [n] \to \Sigma'$
  - da wir m Variablen haben ist der Baum auch nur m tief, d.h. $R(\theta) = m$
  - der BTD entscheidet alle Label korrekt
    - $f_\theta(x_0) = 0 = y_s$ (Warum ?)
    - Da an jedem Knoten 3 Elemente auf 1 gemappt werden, sind am Ende $3m$ Elemente gemappt, und das sind alle Elemente, ausgenommen 0
- ($\Leftarrow$)
  - Sei $\theta = (V, Y, D, D', d^*, E, \delta, \sigma, y')$ ein BTD, die Variablenfunktion wurde mit σ ausgetauscht
  - Wir nehmen an, dass jeder 1-Pfad zu einem Blatt führt, mit dem Wert 1 $y'(d_{\downarrow 1}) = 1,\ \forall d \in D$
  - Damit ist $f_\theta(x) = 1$, wenn es entlang des Weges im BTD, mindestens einmal das Element in einer Covermenge vorkommt (wird jeweils an den Knoten abgefragt, daher kann die Struktur wie ein großes UND aufgefasst werden)

- $\forall x \in X : f_\theta(x) \begin{cases} 1, & \exists j \in [N] : x(\sigma_j) = 1 \\ 0, & \text{else} \end{cases}$
- durch Definition von x_s gilt, $x(\sigma_j) = 1 \Leftrightarrow s \in \sigma_j$
- damit gibt es ein gültiges Set-Cover, d.h.
  $\bigcup_{j=0}^{N-1} \sigma_j = S'$, s.d. alle Labels außer für 0 gleich 1 sind
  $(\forall s \in S' : y_S = 1)$
- außerdem gilt $N = m$, da:
  * $|S'| = 3m$ nach Definition
  * $= \left| \bigcup_{j=0}^{N-1} \sigma_j \right|$
  * $\leq \sum_{j=0}^{N-1} |\sigma_j|$
  * $= \sum_{j=0}^{N-1} 3 = 3N \leq 3m$

    Da alle Covermengen die Größe 3 haben
- Somit gibt es keine Überschneidungen, d.h. das Cover ist exakt
  $\forall \{j, l\} \in \binom{[N]}{2} : \sigma_k \cap \sigma_l = \emptyset$
- damit ist $\bigcup_{j=0}^{N-1} \sigma_j$ eine Lösung für das exact cover by 3-sets problem



Erklärung v2

Wir zeigen, dass das Lernen von BDTs NP-hard ist, indem wir das Cover-by-3-Sets Problem darauf reduzieren

- Die zu Covernde Menge hat ein drei oder ein Vielfaches von 3 Elementen, sonnst kann es ja keine Lösung zu dem Problem geben
- außerdem definieren wir wie bei Haussler Data wieder eine Funktion welche angibt ob ein Element in einer Teilmenge des Covers liegt, nur dass der Output diesmal geflippt ist (1 wenn $s \in \sigma$, 0 sonnst)
- $\Rightarrow$ (hat man ein Cover, so gibt es einen Baum der die Elemente korrekt entscheidet)
  - Wir konstruieren den BDT so, dass wir an jedem Knoten fragen: liegt s in dieser Teilmenge des Covers ?
  - Die Antwort der Frage ist dabei immer der Wert der Funktion $x_s(\sigma)$
    - wenn JA: gehe auf dem 1 Pfad dieser führt immer zu einem Blatt mit $y = 1$
    - wenn NEIN: gehe auf dem 0 Pfad zu nächsten Frage
  - man schreitet nun solange tiefer in den Baum hinein bis eine Frage mit JA beantwortet wird
  - kann man keine Frage mit ja beantworten, so ist nur noch das spezielle Element übrig, dieses wird dann auf $y = 0$ gemappt (links unten im Baum)
  - da wir immer fragen, ob $s \in \sigma$ und $|\sigma| = 3$ 'verbrauchen' wir pro Frage 3 mögliche Elemente, somit ist der Baum nur m tief
  - Note: die Reihenfolge der Fragen im Baum ist egal
- $\Leftarrow$ (Sei ein BTD gegeben, wie sieht das Set-Cover aus)
  - wir gehen einfach für jedes Element durch den Baum durch und fragen ob das Element zum Cover gehört
  - vereinigt man alle diese Elemente für die der Baum JA sagt, erhält man das Cover
  - nur das spezielle Element ist hier wieder ausgenommen

4 lernen linearer Funktionen

Die Problemstellung ist es, eine lineare Funktion zu finden, welche 2 Klassen (Label 0 oder 1) von Punkten trennt. Dabei sind die jeweiligen Koordinaten der Datenpunkte die Attributswerte, diese sind aus $\mathbb{R}$

4.1 Formal

- $X = \mathbb{R}^V$ Attribute sind reell (z.B. Gewicht, Größe, etc.)
- $x : S \to \mathbb{R}^V$
- $y : S \to \{0, 1\}$ Es gibt 2 Klassen
- da wir lineare Funktionen lernen:
  - $\Theta = \mathbb{R}^V$, reelle Koeffizienten der linearen Funktion
    z.B. $f(x) = ax + b$ mit $\Theta = (a\ b)^\top$
  - $f : \Theta \to \mathbb{R}^X$
  - $\forall\theta \in \Theta \forall\hat{x} \in X : f_\theta(\hat{x}) = \langle\theta, \hat{x}\rangle = \sum_{v \in V} \theta_v \hat{x}_v$
    (abstrakte Form einer linearen Funktion)

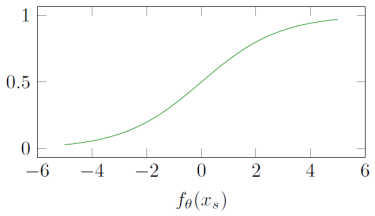
Zufallsvariablen:

- $s \in S$, Sample s der Samplemenge S
- X_s mit $x_s \in \mathbb{R}^V$, Attributsvektor von s
- Y_s mit $y_s \in \{0.1\}$, Label von s
- Θ_v mit $\theta_v \in \mathbb{R}$, Parameter der zu lernenden linearen Funktion

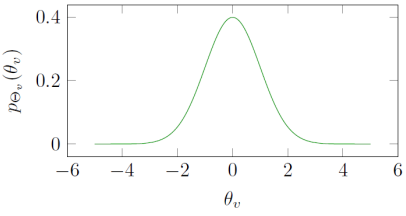
über diese Zufallsvariablen können nun folgende Aussagen getroffen werden:

- $P(X, Y, \Theta) = \prod_{s \in S}(P(Y_s|X_s, \Theta)P(X_s)) \prod_{v \in V} P(\Theta_v)$
  Was genau ist das ?
- $P(\Theta|X, Y) = \frac{P(X, Y, \Theta)}{P(X, Y)} = \frac{P(Y|X, \Theta)P(X))P(\Theta)}{P(X, Y)}$
  Wsk. das unsere Modelparameter stimmen (Likelihood), unter gegebenen Attributwerten und Labels
  $\propto P(Y|X, \Theta)P(\Theta) = \prod_{s \in S} P(Y_s|X_s, \Theta) \prod_{v \in V} P(\Theta_v)$
  Warum ist das prop (Wsk vom Label unter Modelparam und Attr. Wert)?

Um das beste Modell zu finden benutzen wir logistische Regression, dafür benötigen wir die folgende Funktion:



(c) $\forall s \in S : p_{Y_s|X_s,\Theta}(1) = \frac{1}{1+2^{-f_\theta(x_s)}}$



(d) $\forall v \in V : p_{\Theta_v}(\theta_v) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-\theta_v^2}{2\sigma^2}}$

Man kann nun Zeigen, dass das Finden der besten Parameter θ, unter gegebenen Attributen x und Labeln y, äquivalent zum supervised learning Problem ist, sofern man die Lossfunktion, den Regularizer und λ entsprechend wählt:

- $\underset{\theta\in\mathbb{R}^m}{\operatorname{argmax}}\, p_{\Theta|X,Y}(\theta, x, y) \Leftrightarrow \inf_{\theta\in\Theta} \lambda R(\theta) + \frac{1}{|S|}\sum_{s\in S} L(f_\theta(x_s), y_s)$
- wenn für L,R und λ gilt:
  - $\forall r \in \mathbb{R}\, \forall \hat{y} \in \{0,1\} : L(r,\hat{y}) = -\hat{y}r + \log(1+2^r)$
  - $\forall \theta \in \Theta : R(\theta) = \|\theta\|_2^2$
  - $\lambda = \frac{\log e}{2\sigma^2}$
- Dieses Problem wir auch das **l_2-regularized logistic regression problem** genannt (mit Bezug auf x, y, σ)

Beweis:

- $\underset{\theta\in\mathbb{R}^m}{\operatorname{argmax}}\, p_{\Theta|X,Y}(\theta, x, y)$
- $= \underset{\theta\in\mathbb{R}^m}{\operatorname{argmax}} \prod_{s\in S} p_{Y_s|X_s,\Theta}(y_s, x_s, \theta) \prod_{v\in V} p_{\Theta_v}(\theta_v)$
- $= \underset{\theta\in\mathbb{R}^m}{\operatorname{argmax}} \sum_{s\in S} \log\, p_{Y_s|X_s,\Theta}(y_s, x_s, \theta) + \sum_{v\in V} \log\, p_{\Theta_v}(\theta_v)$
  gleich zur Zeile darüber, da der Logarithmus monoton ist und somit sich zwar der Wert des Maximums ändert, nicht aber die Stelle (arg), Umformung der Produkte in Summen mittels: $\log(x \cdot y) = \log(x) + \log(y)$
- $\log p_{Y_s|X_s,\Theta}(y_s, x_s, \theta)$

- $= \; y_s \log p_{Y_s|X_s,\Theta}(1, x_s, \theta) + (1 - y_s) \log p_{Y_s|X_s,\Theta}(0, x_s, \theta)$
  Wie ? irgendwas mit gegenwsks ?

- $= \; y_s \log \frac{p_{Y_s|X_s,\Theta}(1,x_s,\theta)}{p_{Y_s|X_s,\Theta}(0,x_s,\theta)} + p_{Y_s|X_s,\Theta}(0, x_s, \theta)$
  erhalten durch das Ausklammern von y_s, Anwendung von $\log(x) - \log(y) = \log \frac{x}{y}$

- $\frac{p_{Y_s|X_s,\Theta}(1,x_s,\theta)}{p_{Y_s|X_s,\Theta}(0,x_s,\theta)} = \frac{\frac{1}{1+2^{-\langle x_s,\theta\rangle}}}{1-\frac{1}{1+2^{-\langle x_s,\theta\rangle}}} = \frac{1}{1+2^{-\langle x_s,\theta\rangle}-1} = 2^{\langle x_s,\theta\rangle}$
  Im Nenner ist das Gegenereignis vom Zähler, daher 1-..., da wir logistische Regression benutzen setzen wir hier die Formel dafür ein

- $\underset{\theta\in\mathbb{R}^m}{\operatorname{argmin}} \sum_{s\in S}(-y_s\langle\theta, x_s\rangle + \log(1 + 2^{\langle\theta,x_s\rangle})) + \frac{\log e}{2\sigma^2}\|\theta\|_2^2$
  how ? detailed explain

4.2 Algorithmus

Das l_2-regularized logistic regression problem kann mittel des **steepest descent Algorithmus** gelöst werden mit einem *Toleranzparameter* $\epsilon \in \mathbb{R}_0^+$

- $\theta := 0$
  initialisiere θ

- repeat:

  - $d := \nabla\varphi(\theta)$
    berechne die Schrittrichtung (Gradient)
  - $\eta := \underset{\eta'\in\mathbb{R}}{\operatorname{argmin}}\, \varphi(\theta - \eta' d)$ (line search)
    Suche in Schrittrichtung d jene Schrittdistanz η für welche der Loss+Regularizer am kleinsten ist
  - $\theta := \theta - \eta d$
    gehe von θ in Richtung d um η Einheiten, dort ist dann das neue θ
  - if $\|d\| < \epsilon$
    ist die gelaufene Strecke kleiner als unserer Toleranzparameter bricht ab, da wir uns nah genug am Minimum befinden

    * return θ

ToDO Letzter Beweis von linear function Kapitel

5 Semi-supervised / Unsupervised Learning

Details ergänzen

Anderes als beim Supervised learning ist beim Semi/Unsupervised learning nicht jedem Sample ein Label zugeordnet, außerdem muss anderes als beim supervised learning auch nicht jede Labelzuodnung gültig sein. Darum verwenden wir im Weiteren eine Menge $\mathcal{Y} \subseteq \{0,1\}^S$ genannt **feasible labelings** für welche zudem gilt:

- $\mathcal{Y} = \{0,1\}$, Spezialfall entspricht unlabled data
- $|\mathcal{Y}| = 1$, Spezialfall entspricht labled data
- nicht-triviale (d.h. andere als die zwei Fälle oberhalb) erlaubt es endliche Strukturen zu kodieren, z.B. Maps, Äquivalenzrelationen oder Ordnungen

5.1 learning und inference

Sei folgendes gegeben:

- $T = (S, X, x, \mathcal{Y})$, constrained data
- $\Theta \neq \emptyset$ mit $f : \Theta \to \mathbb{R}^X$
- $R : \Theta \to \mathbb{R}_0^+$, Regularizer
- $L : \mathbb{R} \times \{0,1\} \to \mathbb{R}_0^+$, Lossfunction
- $\lambda \in \mathbb{R}_0^+$, regularizarion Parameter

Daraus lässt sich nun das **learning and inference Problem** definieren:

$$\min_{y \in \mathcal{Y}} \inf_{\theta \in \Theta} \lambda R(\theta) + \frac{1}{|S|} \sum_{s \in S} L(f_\theta(x_s), y_s)$$

Den Spezialfall wenn $\mathcal{Y} = \{\hat{y}\}$ nennt man **supervised learning problem**. Den Spezialfall wenn $\Theta = \{\hat{\theta}\}$ nennt man **inference problem**, dieses kann wie folgt formuliert werden:

$$\min_{y \in \mathcal{Y}} \sum_{s \in S} L(f_{\hat{\theta}}(x_s), y_s)$$

Dieses inference Problem kann in ein binäres lineares Optimierungsproblem umgewandelt werden:

- $\operatorname{argmin}_{y \in \mathcal{Y}} \sum_{s \in S} L(f_{\hat{\theta}}(x_s), y_s)$

- $= \underset{y \in \mathcal{Y}}{\operatorname{argmin}} \sum_{s \in S} y_s L(f_{\hat{\theta}}(x_s), 1) + (1 - y_s) L(f_{\hat{\theta}}(x_s), 0)$
- $= \underset{y \in \mathcal{Y}}{\operatorname{argmin}} \sum_{s \in S} y_s (L(f_{\hat{\theta}}(x_s), 1) - L(f_{\hat{\theta}}(x_s), 0)$

6 Klassifizierung

Bei der Klassifizierung versuchen wir eine Abbildung φ zu finden, welche Elemente einer Menge A exakt ein **Klassenlabel** aus einer Menge B zuweist, d.h. $\varphi : A \to B$

6.1 Formal

Suche jene $\varphi : A \to B$ aus $\varphi \subseteq A \times B$ für die gilt:

- $\forall a \in A \, \exists b \in B : (a, b) \in \varphi$
  Jedes Element aus A muss ein Klassenlabel erhalten
- $\forall a \in A, \, \exists b, b' \in B : (a, b) \in \varphi \wedge (a, b') \in \varphi \Rightarrow b = b'$
  Jedes Element aus A hat exakt ein Klassenlabel

für diese Abbildung gilt somit auch:
$\forall a \in A : \sum_{b \in B} y_{ab} = 1,$
d.h. die Zuweisungsflag y ist genau einmal 1, also wird nur genau ein Klassenlabel zugewiesen, $y(a, b) = 1$ bedeutet dabei nicht, dass die Klassifikation von a als b korrekt ist, sondern lediglich, dass die gelernte Funktion a als b klassifiziert
Wir betrachten bei der Klassifizierung **constrained data** $(S, X, x, \mathcal{Y})$ mit:

- $S = A \times B$, Paare aus Elementen und Klassenlabeln
- $X = B \times \mathbb{R}^V$, Class-Sampleattribute-pairs, s.d. das erste Element **immer** die Klasse ist
  $\forall a \in A \, \forall b \in B \, \exists \hat{x} \in \mathbb{R}^V : x_{ab} = (b, \hat{x})$
  z.B. x = (label, atr_1, atr_2, ...) -> Label wird immer am Anfang appended
- $\mathcal{Y} = \left\{ y \in \{0, 1\}^S \,\middle|\, \forall a \in A : \sum_{b \in B} y_{ab} = 1 \right\}$
  jedes Sample wird zu genau einem Label gemappt

Wir nehmen uns nun eine **Menge linearer Funktionen mit separaten Koeffizienten**
$\forall\theta \in \Theta\, \forall b \in B\, \forall \hat{x} in \mathbb{R}^V : f_\theta((b, \hat{x})) = \sum_{v \in V} \theta_{bv}\hat{x}_v = \langle \theta_b, \hat{x} \rangle$
Aus dieser Funktionsschar wählen wir uns dann mittels des Klassenlabels b die benötigte Funktion aus. Z.B. gäbe es bei der Klassifizierung von handschriftlichen Ziffern 10 verschiedene Funktionen

Seien folgende Zufallsvariablen gegeben:

- X_{ab}, Attributsvektor
- Y_{ab}, Decisionflag
- θ_{bv}, Parameter
- $\mathcal{Z} \subseteq \{0,1\}^{A \times B}$, feasible descisions (wir suchen $\mathcal{Z} = \mathcal{Y}$)

Für diese ZF's lassen sich dann wieder Supervised Learning $P(\Theta|X, Y, Z)$ und Inference $P(Y|X, Z, \theta)$ formulieren
Die ZF's sind dabei für $P(Y|X)$ logistisch verteilt, für die Parameter Θ normal verteilt und für $P(Z|Y$ gleich verteilt
uniform distribution
$\forall \mathcal{Z}\{0,1\}^{A\times B}\, \forall y \in \{0,1\}^{A\times B} : p_{Z|Y}(\mathcal{Z}, y) \propto \begin{cases} 1 & y \in \mathcal{Z} \\ 0 & \text{else} \end{cases}$ Man stellt fest, dass das Finden der besten Parameter θ unter Attributen x und Decisions y in $|B|$ separate unabhängige Teilprobleme aufgeteilt werden kann, diese können dann auch separat gelöst werden.
Beweis:

- $\underset{\theta \in \mathbb{R}^{B\times V}}{\operatorname{argmax}}\, p_{\Theta|X,Y,Z}(\theta, x, y, \mathcal{Y})$
- $= \underset{\theta \in \mathbb{R}^{B\times V}}{\operatorname{argmin}} \sum_{(a,b)\in A\times B} (-y_{ab} f_\theta(x_{ab}) + \log(1 + 2^{f_\theta(x_{ab})})) + \frac{\log e}{2\sigma^2}\|\theta\|_2^2$
  flipping sign macht aus max ein min, einsetzen der Formel für Supervised learning, dort dann einsetzen von normal und log. distribution, aus Produkt mach Summe mit Logarithmus
- $\min_{\theta \in \mathbb{R}^{B\times V}} \sum_{(a,b)\in A\times B} (-y_{ab}\langle \theta_b, x'_{ab}\rangle + \log(1 + 2^{\langle \theta_b, x'_{ab}\rangle})) + \frac{\log e}{2\sigma^2}\|\theta\|_2^2$
  einsetzen der weighted sum für f_θ

- $= \min_{\theta \in \mathbb{R}^{B \times V}} \sum_{b \in B} \Big(\sum_{a \in A} (-y_{ab} \langle \theta_b, x'_{ab} \rangle + \log(1 + 2^{\langle \theta_b, x'_{ab} \rangle})) \Big) + \frac{\log e}{2\sigma^2} \|\theta\|_2^2$
  aufteilen der Summe über Paaren in 2 Summen
- $= \sum_{b \in B} \min_{\theta \in \mathbb{R}^{B \times V}} \Big(\sum_{a \in A} (-y_{ab} \langle \theta_b, x'_{ab} \rangle + \log(1 + 2^{\langle \theta_b, x'_{ab} \rangle})) \Big) + \frac{\log e}{2\sigma^2} \|\theta\|_2^2$
  rausziehen der Summe da konstant ???

Für jede constrained data, Parameter, und Descision gibt es eine Lösung für das Inference Problem, wenn bestimmte Bedingungen gelten:

- $\min_{y \in \mathcal{Y}} \sum_{(a,b) \in A \times} L(f_\theta(x_{ab}), y_{ab})$ hat eine Lösung wenn:
  - $\forall a \in A : \varphi(a) \in \max_{b \in B} \langle \theta_b, x'_{ab} \rangle$, jedes Element bekommt irgendein Label zugeordnet (aus den Maximas der Funktionsschar)
  - $\forall (a, b) \in A \times B : \hat{y}_a = 1 \Leftrightarrow \varphi(a) = b$, Wenn das Klassenlabel b dem Element a zugeordnet wurde so muss das Mapping das gleiche Ergebnis repräsentieren

Beweis:

- Todo

7 Partitionierung

Bei der Partitionierung versucht man eine Menge in kleinere disjunkte Teilmengen so zu zerlegen, ohne die Größte, die Anzahl oder irgendwelche andere Eigenschaften davor zu wissen.

7.1 Formal

eine **Partition** $\Pi \subseteq 2^A$ ist ein Zerlegung von A in nicht-leere disjunkte Teilmengen, alle Elemente von A müssen dabei in einer Teilmenge vorkommen $\bigcup_{\pi \in \Pi} \pi = A$

Für Partitionen lässt sich ebenfalls eine Äquivalenzrelation definieren:

$\forall a, a' \in A : a \equiv_\Pi a' \Leftrightarrow \exists U \in \Pi : a \in U \wedge a' \in U$

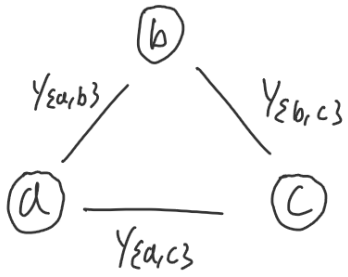
Zwei Elemente a und a' sind gleich in Π wenn sie in der gleichen Teilmenge der Partition liegen
gilt dies Für alle Elemente aus A, so sind zwei Partitionen gleich
Wir können dieses Äquivalenzrelation als Entscheidung y wie folgt ausdrücken

- $y : \binom{A}{2} \to \{0,1\}$
  - $y_{\{a,b\}} = 1$, a und b sind in der gleichen TM
  - $y_{\{a,b\}} = 0$, a und b sind in unterschiedlichen TM's

Die Menge dieser Entscheidungen sind alle jene, welche folgende Gleichung erfüllen:

$\forall a \in A \, \forall b \in A \setminus \{a\} \, \forall c \in A \setminus \{a,b\} : y_{\{a,b\}} + y_{\{b,c\}} - 1 \leq y_{\{a,c\}}$

Wir lernen die Partitionen indem wir die Äquivalenzen von Elementen einer Men-



$y_{\{a,b\}}$	$y_{\{b,c\}}$	$y_{\{a,c\}}$	$y_{\{a,b\}} + y_{\{b,c\}} - 1 \leq y_{\{a,c\}}$
0	0	0	✓
0	0	1	✓
0	1	0	✓
0	1	1	✓
1	0	0	✓
1	0	1	✓
1	1	0	✗
1	1	1	✓

ge als Entscheidungen von Constrained Data abbilden.

7.2 Formal

Constrained Data: (S, X, x, Y):

- $S = \binom{A}{2}$
- $\mathcal{Y} = \left\{ y : \binom{A}{2} \to \{0,1\} \middle| \, y_{\{a,b\}} + y_{\{b,c\}} - 1 \leq y_{\{a,c\}} \right\}$
- $X = \mathbb{R}^V$

7.3 Algorithmen

7.3.1 Greedy-Moving

Sei $\text{move}_{aU}[\Pi] = \Pi \setminus \{U\} \setminus \{W \in \Pi | \, a \in W\}$
$\cup \{U \cup \{a\}\}$

$$\cup \bigcup_{W \in \Pi \mid a \in W \wedge \{a\} \neq W} \{W \setminus \{a\}\}$$

Die Operation das Element $a \in A$ in die Untermenge $U \in \Pi \cup \{\emptyset\}$ innerhalb der Partition Π zu verschieben

$\Pi' = \text{greedy-moving}(\Pi)$

- choose $(a, U) \in \underset{(a',U') \in A \times (\Pi \cup \{\emptyset\})}{\operatorname{argmin}} \varphi(y^{\text{move}_{a'U'}[\Pi]}) - \varphi(y^{\Pi})$, suche die beste Move-Operation aus
- if $\varphi(y^{\text{move}_{a'U'}[\Pi]}) - \varphi(y^{\Pi}) < 0$, tatsächliche Verbesserung liegt vor ?
  - $\Pi' := \text{greedy-moving}(\text{move}_{aU}\Pi)$, commit result
- else
  - $\Pi' = \Pi$, disgard result

Das Problem mit diesem Algorithmus ist, dass er sich in jedem kleinen lokalen Minima verfängt und dort terminiert, um das Problem zu beheben wurde der Algorithmus mit der Technik von Kernighan und Lin verbessert. Dabei werden anstelle von einzelnen Moves, längere Sequenzen von Moves ausgeführt, es wird dann die Sequenz genommen welche den größten kumulativen abstieg erzielt. So ist es Möglich innerhalb der Sequenz auch möglich aus Perspektive der Kostenfunktion einen Rückschritt zu machen, wenn dies dann in der Summe eine größere Verbesserung ermöglicht.

$\Pi' = \text{greedy-moving-kl}(\Pi)$

- $\Pi_0 := \Pi$
- $\delta_0 := 0$
- $A_0 := A$, active set
- $t := 0$
- repeat
  - choose $(a_t, U_t) \in \underset{(a,U) \in A_t \times (\Pi \cup \{\emptyset\})}{\operatorname{argmin}} \varphi(y^{\text{move}_{a'U'}[\Pi_t]}) - \varphi(y^{\Pi_t})$, wähle die beste Move-Operation unter allen Elementen welche sich noch im Active-Set befinden
  - $\Pi_{t+1} := \text{move}_{a_t U_t}[\Pi_t]$, führe Move-Operation aus
  - $\delta_{t+1} := \varphi(y^{\Pi_{t+1}}) - \varphi(y^{\Pi_t})$, berechne Verbesserung

- $A_{t+1} := A_t \setminus \{a_t\}$, entferne das benutzte Element aus dem Active-Set
- $t := t + 1$

- until $A_t = \emptyset$

- $\hat{t} := \min \operatorname{argmin}_{t' \in \{0,\dots,|A|\}} \sum_{\tau=0}^{t'} \delta_t$, Wähle die Stelle (argmin) in der Sequenz aus an der beste Verbesserung ist, gibt es mehrere, dann nehme die kürzeste (min)

- if $\sum_{\tau=0}^{t'} \delta_t < 0$ Sequenz bringt Verbesserung (siehe Tabelle)
  - $\Pi' :=$ greedy-moving-kl($\Pi_{\hat{t}}$), commit result

- else
  - $\Pi' = \Pi$, disgard result

t	δ_t	$\sum \delta_t$
0	0	0
1	-2	-2
2	1	-1
3	-8	-9
4	10	-19

8 Wahrscheinlichkeit

8.1 Example 1

Seien A und B zwei Würfel und sei $X = A + B$ die Summe beider Würfel

- $P(A = 2, B = 3, X = 7) = 0$, da es unmöglich ist, das alle 3 sich im angegebenen Zustand befinden
- $P(A = 2, B = 3, X = 5) = \frac{1}{36}$, genau 1 Ereignis (von 36) erfüllt alle Zustände
- $P(X = 5 | A = 2, B = 3) = 1$, die Wsk. dass $A + B = 5$ wenn wir wissen, das $A = 2$ und $B = 3$ ist natürlich 1
- $P(A, B | X = 5) = \frac{1}{4}$, es gibt nur 4 verschiedene Kombinationen von A und B, s.d. $A + B = 5$
  $X = A + B \to 5 = 1 + 4 = 2 + 3 = 3 + 2 = 4 + 1$

8.2 Example 2

- $P(B|A) = \frac{P(A,B)}{P(A)} = \frac{P(A,B)}{\sum P(A,B)}$
- $P(A, B) = P(A|B)P(B)$
- $P(A, B) = P(A \cup B)$
- $P(A, B, C) = P(A|B, C)P(B, C) = P(A|B, C)P(B|C)P(C)$