Strukturelle Modelle in der Bildverarbeitung
Markovsche Modelle auf Bäumen

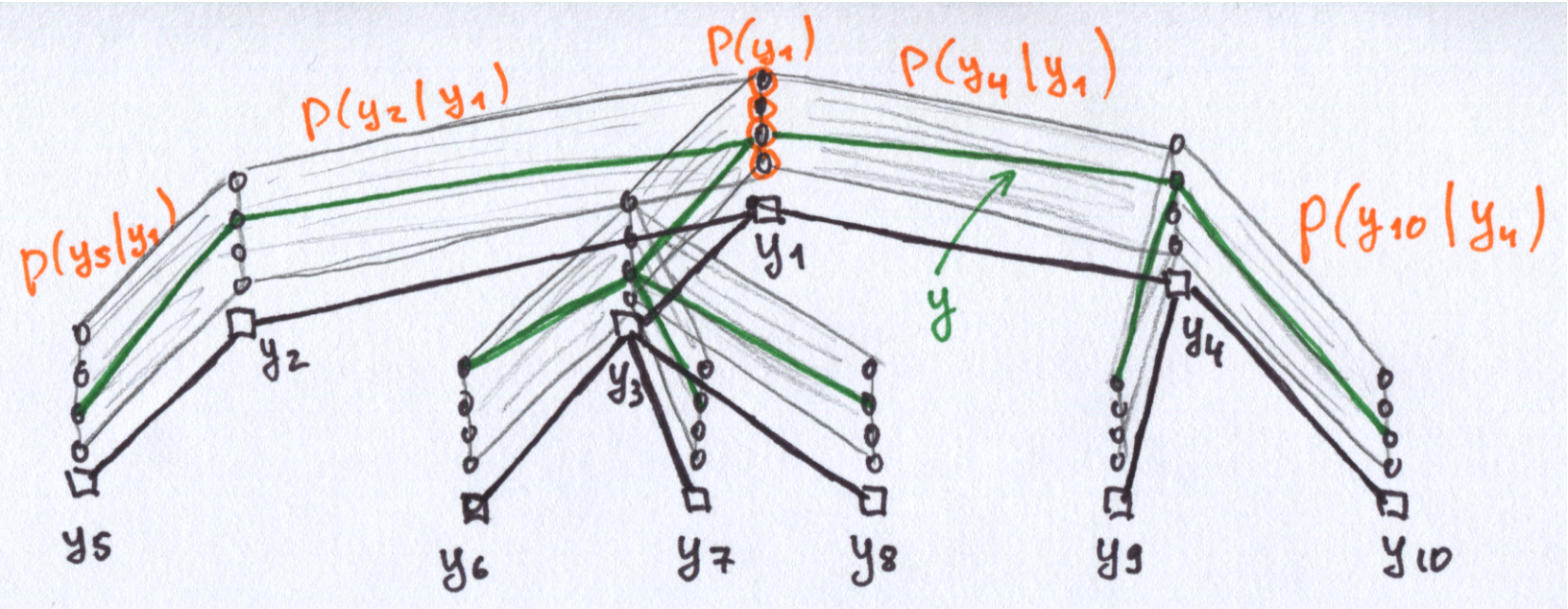
D. Schlesinger – TUD/INF/KI/IS

- Markovsche Modelle auf Bäumen
- Dynamische Programmierung für Bäume
- Lernen
- Dynamische Programmierung für allgemeine Graphen

Markovsche Modelle auf Bäumen

Die „gewöhnliche“ Parametrisierung (Ketten-ähnlich):

Die diskreten Variablen $y_i \in K$ sind durchnummeriert,
jede Variable (außer einer – Wurzel) hat einen „Vorgänger“ $j(i) < i$



Wahrscheinlichkeit einer Zustandskonfiguration $y = (y_1, y_2 \dots y_n)$ ist

$$p(y) = p(y_1) \cdot \prod_{i=2}^{n} p(y_i | y_{j(i)})$$

Parametrisierung durch marginale Wahrscheinlichkeiten (auch Ketten-ähnlich):

$$p(y) = p(y_1) \cdot \prod_{i=2}^{n} p(y_i|y_{j(i)}) = p(y_1) \cdot \prod_{i=2}^{n} \frac{p(y_i, y_{j(i)})}{p(y_{j(i)})} = \frac{\prod\limits_{i,j(i)} p(y_i, y_{j(i)})}{\prod\limits_{i} p(y_i)^{n(i)-1}}$$

$n(i)$ ist die Anzahl der „zu y_i benachbarten“ Variablen (Vorgänger oder Nachfolger).

Es gibt keine Wurzel mehr, die Variablen müssen nicht explizit nummeriert werden.

Gegeben sei ein zyklenfreier Graph $V = (R, E)$ mit

- der Menge R der Knoten ($r \in R$ bezeichnet einen Knoten)
- der Menge $E = \{\{r, r'\}\}$ der Kanten

y_r ist die Zufallsvariable, die dem Knoten r entspricht.

$$p(y) = \frac{\prod\limits_{\{r,r'\}\in E} p(y_r, y_{r'})}{\prod\limits_{r\in R} p(y_r)^{n(r)-1}} = \prod_{\{r,r'\}\in E} p(y_r, y_{r'}) \cdot \prod_{r\in R} p(y_r)^{1-n(r)}$$

Und noch etwas allgemeiner:

$$p(y) = \frac{1}{Z} \prod_{\{r,r'\} \in E} g_{rr'}(y_r, y_{r'}) \cdot \prod_{r \in R} q_r(y_r) \quad \text{mit der Partition Funktion } Z.$$

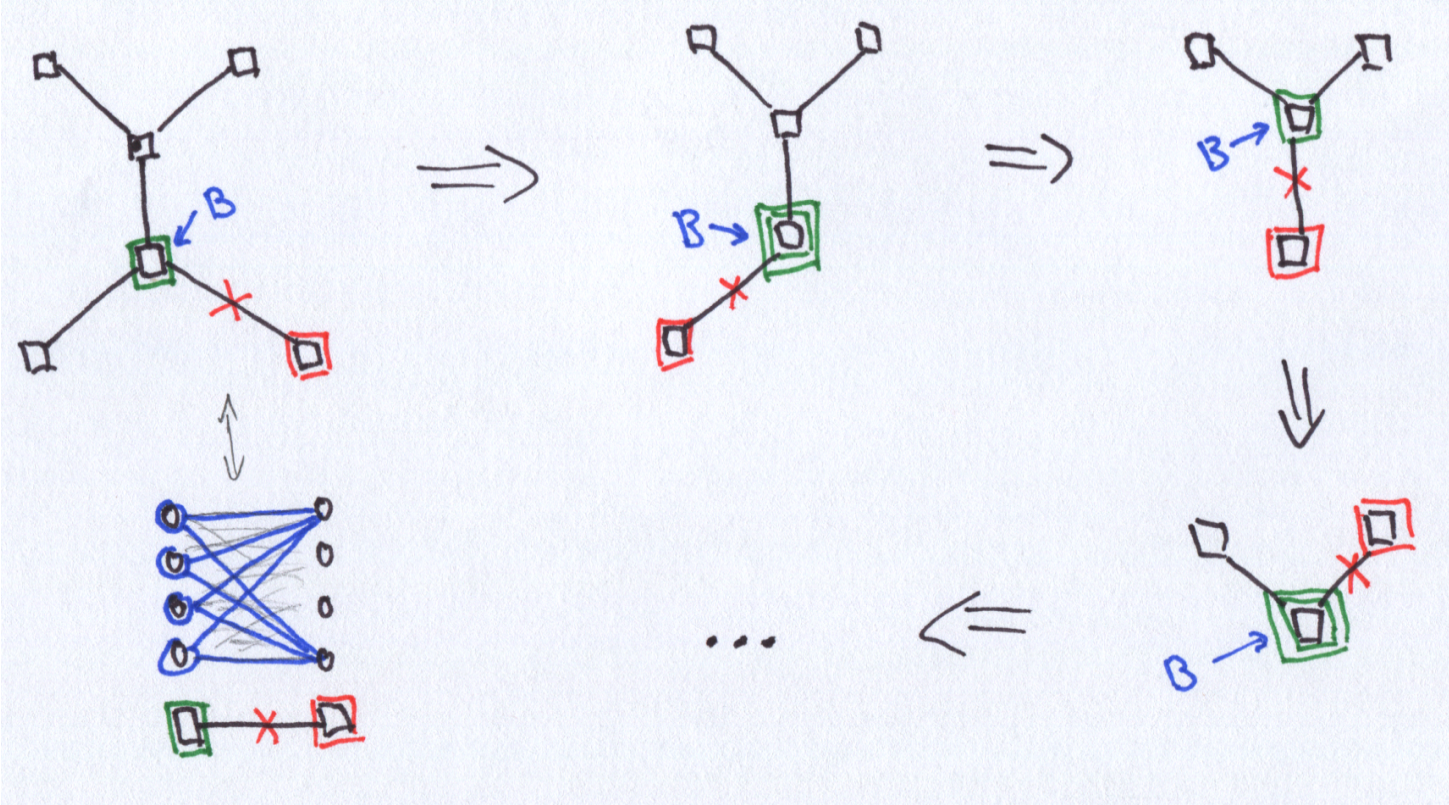
Zusätzliche Anmerkungen zu Parametrisierungen:

- Ausgangspunkt ist $p(y)$.
- Für eine beliebige $p(y)$ **existieren** alle $p(y_r)$, $p(y_r, y_{r'})$, $p(y_r|y_{r'})$ etc. (selbst wenn die entsprechende Kante nicht existiert).
- Für Bäume ergeben sich daraus verschiedene Parametrisierungen. Sie definieren **ein und dieselbe** Wahrscheinlichkeitsverteilung $p(y)$. Sie sind aber von einander **nicht unabhängig**, z.B. $p(y_r|y'_r)$ in einer Parametrisierung soll $p(y_r, y_{r'})$ und $p(y_{r'})$ in der anderen entsprechen.
- Vielmehr, in **einer** Parametrisierung sind die Parameter von einander nicht unbedingt unabhängig, z.B. $\sum_{k'} p(y_r{=}k, y_{r'}{=}k') = p(y_r{=}k)$ muss gelten (marginal constraints).
- Für einen beliebigen Satz von q und g existiert genau **eine** $p(y)$, für eine $p(y)$ existieren **mehrere** Parametrisierungen mit q und g.
- Entsprechen q und g irgendwelchen Wahrscheinlichkeitsverteilungen, so sind diese nicht unbedingt die marginalen von der daraus entstehenden $p(y)$. Wenn aber dabei die marginal constraints erfüllt sind, dann doch (aber nur für Bäume).

Dynamische Programmierung für Bäume

In zwei Worten – es ist dasselbe, wie bei Ketten.

Variablen werden nach einander eliminiert (durch Bellmansche Funktionen ersetzt).



Dies funktioniert sowohl für SumProd Probleme (Partition Funktion, marginale Wahrscheinlichkeiten), als auch für die Suche nach der a-posteriori wahrscheinlichsten Zustandskonfiguration (MinSum Problem).

Zusätzlich zum „gewöhnlichen“ Lernen wird nach der **Baumstruktur** auch gesucht.

Gegeben sei eine Lernstichprobe der Zustandskonfigurationen $L = (y^1, y^2 \dots y^l)$
Gesucht wird:

- **der Graph**, d.h. die (zyklenfreie) Menge der Kanten E,
- die numerischen Parameter **für diesen Graphen**, d.h.
  - $q_r(k) = p(y_r{=}k)$ für alle $r \in R$ und
  - $g_{rr'}(k,k) = p(y_r{=}k, y_{r'}{=}k')$ für alle $\{r, r'\} \in E$

Die Menge der zu optimierenden Parameter hängt vom Graphen ab.

Maximum Likelihood:

$$\ln p(L; E, q, g) = \sum_l \ln p(x^l; E, q, g) =$$

$$= \sum_l \left[\sum_{r \in R} \Big(1 - n(r)\Big) \ln q_r(y_r^l) + \sum_{rr' \in E} \ln g_{rr'}(y_r^l, y_{r'}^l) \right] \to \max_{E,q,g}$$

$$\text{s.t.} \quad \sum_k q_r(k) = 1, \quad \sum_{kk'} g_{rr'}(k,k') = 1, \quad \sum_{k'} g_{rr'}(k,k') = q_r(k)$$

„Verschachtelte“ Optimierung:
Zunächst betrachte man, die Aufgabe auf einem fixierten Baum,
dann setzt man die gefundene Lösung in die Optimierung bezüglich der Struktur ein.

Sei der Baum gegeben:

$$\sum_l \left[\sum_{r \in R} \Big(1 - n(r)\Big) \ln q_r(y_r^l) + \sum_{rr' \in E} \ln g_{rr'}(y_r^l, y_{r'}^l) \right] =$$

$$\sum_{r \in R} \Big(1 - n(r)\Big) \sum_k p_r^*(k) \ln q_r(k) + \sum_{rr' \in E} \sum_{kk'} p_{rr'}^*(k, k') \ln g_{rr'}(k, k') \to \max_{q,g}$$

$$\text{s.t. ...} \quad \sum_{k'} g_{rr'}(k, k') = q_r(k)$$

mit den aus der Lernstichprobe berechneten Statistiken $p_r^*(k)$ und $p_{rr'}^*(k, k')$.

Wegen den Nebenbedingungen ist es nicht möglich, die Summanden von einander getrennt zu optimieren. Ausweg:

- Gehe zu der Parametrisierung mit den bedingten Wahrscheinlichkeitsverteilungen. Sie sind mit einander nicht gekoppelt.
- Beweise $p(y_r|y_{r'}) = p^*(y_r|y_{r'})$ (durch das Schennonsche Lemma).
- Daraus ergibt sich automatisch $g_{rr'}(k, k') = p_{rr'}^*(k, k')$ und $q_r(k) = p_r^*(k)$.

Man setze $g_{rr'}(k,k') = p^*_{rr'}(k,k')$ und $q_r(k) = p^*_r(k)$ in die Zielfunktion ein:

$$\sum_{r\in R}\Big(1-n(r)\Big)\sum_k p^*_r(k)\ln p^*_r(k) + \sum_{rr'\in E}\sum_{kk'} p^*_{rr'}(k,k')\ln p^*_{rr'}(k,k') =$$

$$= \sum_{r\in R}\sum_k p^*_r(k)\ln p^*_r(k) +$$

$$+ \sum_{rr'\in E}\left[\sum_{kk'} p^*_{rr'}(k,k')\ln p^*_{rr'}(k,k') - \sum_k p^*_r(k)\ln p^*_r(k) - \sum_k p^*_{r'}(k)\ln p^*_{r'}(k)\right]$$

Das ist die Qualität (log-Likelihood) des optimalen Parametersatzes für einen fixierten Baum. Dies soll jetzt bezüglich der Baumstruktur optimiert werden.

Die erste Summe $\sum_{r\in R}$ hängt von der Baumstruktur nicht ab. Man bezeichne

$$c_{rr'} = \sum_{kk'} p^*_{rr'}(k,k')\ln p^*_{rr'}(k,k') - \sum_k p^*_r(k)\ln p^*_r(k) - \sum_k p^*_{r'}(k)\ln p^*_{r'}(k)$$

Man erhält die Aufgabe der Suche nach dem maximalen aufspannenden Baum:

$$\sum_{rr'\in E} c_{rr'} \to \max_E$$

Dynamische Programmierung für allgemeine Graphen

Man betrachte den folgenden Prozess der Erzeugung eines Graphen:

Die Knoten werden nach und nach in den Graphen eingefügt (ein Knoten am Anfang).

Der neu eingefügte Knoten wird mit einem vollverbundenen Teilgraphen durch die Kanten verbunden. Dieser vollverbundene Teilgraph besteht aus maximal w Knoten.

Nachdem alle Knoten eingefügt sind, werden manche Kanten entfernt.

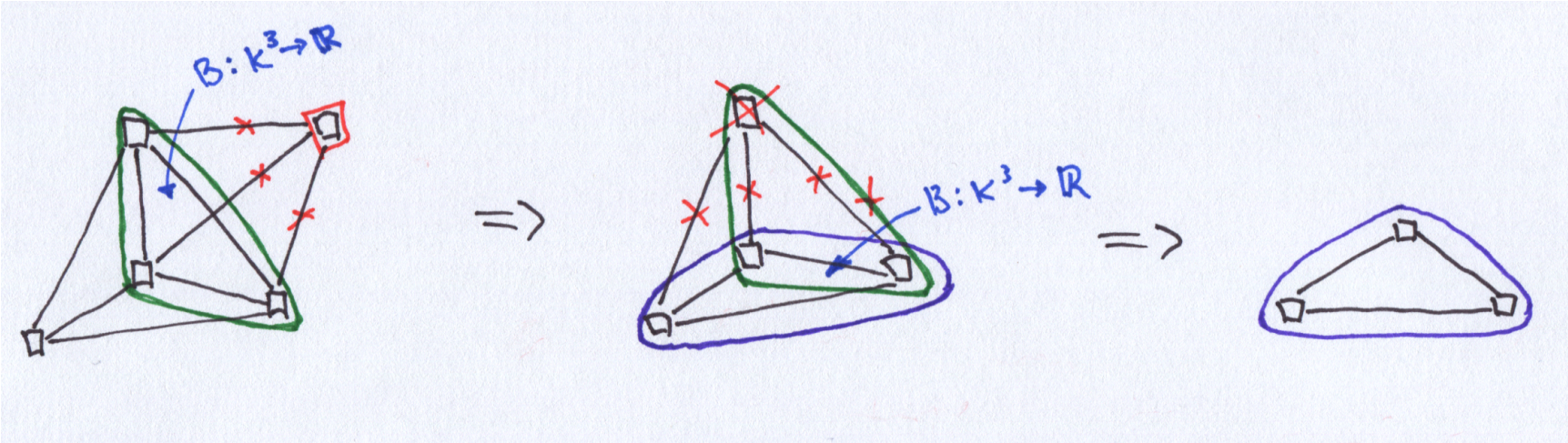
Gegeben sei ein Graph. Seine **Breite** (treewidth) ist die kleinste Zahl w so, dass der Graph durch den wie oben beschriebenen Prozess erzeugt werden kann – partieller w-Baum.

Beispiele:
Ketten, Bäume: $w = 1$
Zyklen, Simple Netze: $w = 2$
Gitter: $n \times m$: $w = \min(n, m)$

Bei einem fixierten w kann in polynomieller Zeit beantwortet werden, ob ein gegebener Graph die Breite w hat – polynomiell in n, allerdings exponentiell in w
$\rightarrow$ die Aufgabe der Bestimmung von w ist NP-vollständig.

Dynamische Programmierung für allgemeine Graphen

Die Idee der Dynamischen Programmierung: wenn die Reihenfolge der Knoten Bekannt ist, kann man die Knoten in der umgekehrten Reihenfolge eliminieren. Die Bellmansche Funktionen haben dabei die Ordnung maximal w, d.h. $B : K^w \to \mathbb{R}$.



Die Dynamische Programmierung hat die Zeitkomplexität $O(nK^{w+1})$

Beispiele:
Kette: eliminiert wird der „erste“ Knoten, $w = 1$, $B : K \to \mathbb{R}$, $O(nK^2)$
Baum: eliminiert wird immer ein Blatt, alles andere – dasselbe
Zyklus: eliminiert wird ein beliebiger Knoten, $w = 2$, $B : K \times K \to \mathbb{R}$, $O(nK^3)$
Das obige Beispiel: $w = 3$, $B : K^3 \to \mathbb{R}$, $O(nK^4)$