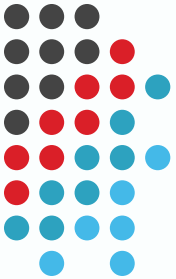
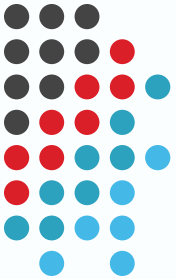


論理、確率、機械学習

佐藤泰介

札幌 June 2013

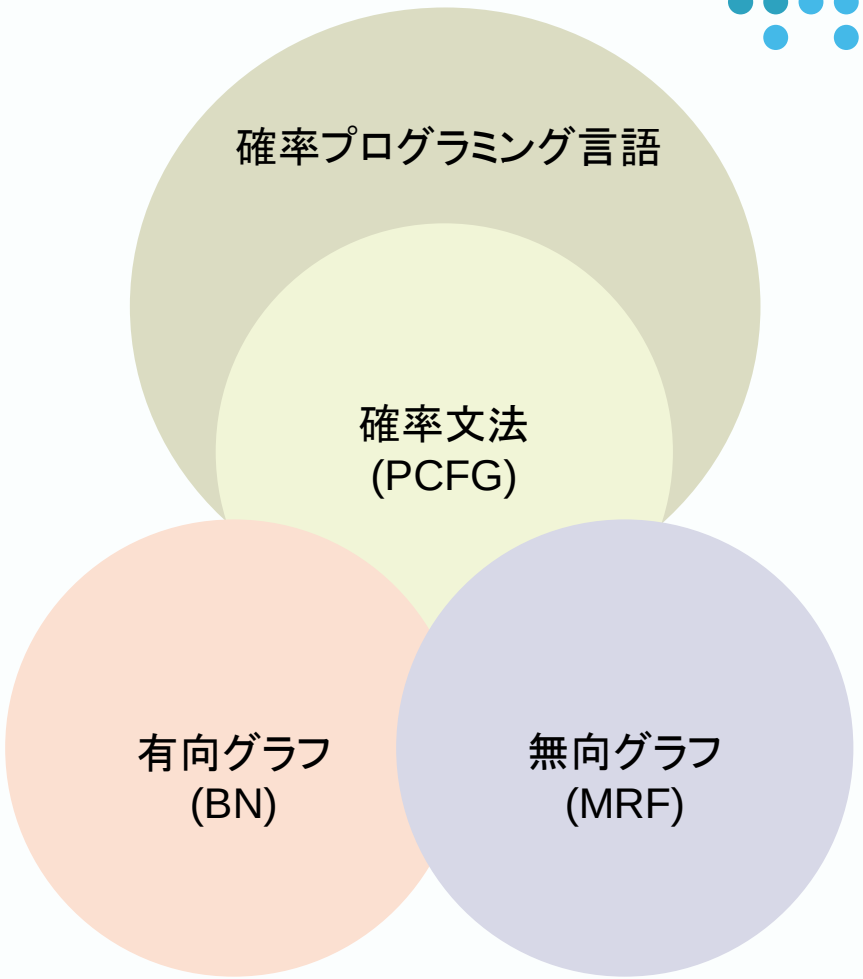
概略



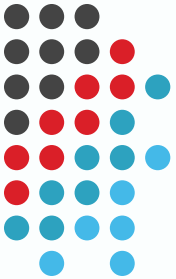
- 論理と確率を統合した高水準言語による機械学習を実現する
  - 理論的興味と界面の必然性
- 関連研究の紹介
  - 統計関係学習(SRL)– 北米
  - 確率論理学習(PLL)– 日欧
- どこまで統合できたか:PRISM言語[Sato,Kameya'97]
  - 意味レベル:最小モデルと確率空間
  - 計算レベル:命題論理と確率計算
  - 推論レベル:アブダクションと統計的推論
  - 学習レベル:非Bayes推定とBayes推定

確率モデルの世界

- 統計学で扱う名前のある分布
- 確率文法(PCFG)
  - 規則(rule), 制約(constraint)
    - 再帰あり
    - 生成的
- グラフィカルモデル
  - グラフ表現
    - 再帰なし
    - 生成的(BN), 判別的(MRF)
  - プレート表現
    - 繰り返しあり
    - 生成的
- 確率プログラミング(PP)言語
  - 再帰、条件付き分岐, データ構造あり



モデルの高度化を目指して



- 素性ベースの機械学習（1990年代）
  - 決定木,ベイズネット（BN）,マルコフ確率場（MRF）,SVM
- 関係，論理概念の取り込み（2000年代）
  - グラフィカルモデル＋関係 ➔ 統計的関係学習（SRL）
  - 帰納論理プログラミング＋確率 ➔ 確率論理学習（PLL）
- グラフからプログラムへ（2010年代）
  - 基本分布＋汎用プログラミング言語
    ➔ 確率プログラミング（PP）

代表的SRL/PLL/PP

Acronym	Year	Full name	Main authors
PHA/ICL	’93 ‘97	Probabilistic Horn Abduction / Independent Choice Logic	D. Poole
PRISM	’95	Programming In Statistical Modeling	T. Sato
SLPs	’96 ’01	Stochastic Logic Programs	S. Muggleton J. Cussens
IBAL	’97	n/a	A. Pfeffer
RBNs	’97	Relational Bayesian Networks	M. Jaeger
PRMs	’98	Probabilistic Relational Models	D. Koller, N. Friedman
P-log	‘04	n/a	C. Baral, M. Gelfond
CFDs	’04	Case Factor Diagrams	D. McAllester, M. Collins
Dyna	’04		J. Eisner
MLNs	’04	Markov Logic Networks	P. Domingos
BLOG	’05	Bayesian Logic	B. Milch S. Russell
MEBNs	’06	Multi-entity Bayesian Networks	K. Laskey
ProbLog	’07	Probabilistic Prolog	L. De Raedt
Church	’08	n/a	N. Goodman, J. Tenenbaum
Factorie	’08	n/a	A. McCallum
Figaro	’09	n/a	A. Pfeffer
PITA	’11	Probabilistic Inference with Tabling and Answer subsumption	F. Riguzzi

BN（ベイズネット）
– Going to New York

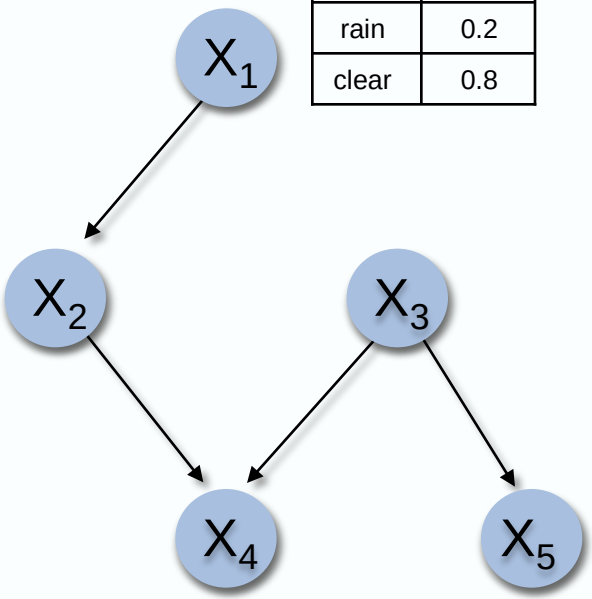
X_1: Weather in Tokyo
$x_1 \in$ { rain, clear }
X_2: departure delay
$x_2 \in$ { yes, no }
X_3: Weather in NY
$x_3 \in$ { rain, clear }
X_4: arrival delay
$x_4 \in$ { yes, no }
X_5: musical-cancelled
$x_5 \in$ { yes, no }

CPT

x_1	$P(x_1)$
rain	0.2
clear	0.8

CPT

	$P(x_2 \mid x_1)$	$P(x_2 \mid x_1)$
x_2 / x_1	rain	clear
yes	0.3	0.1
no	0.7	0.9

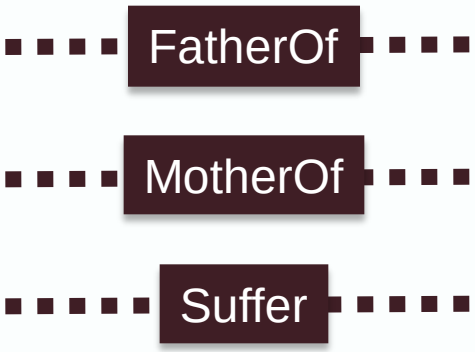


BN_{NY}

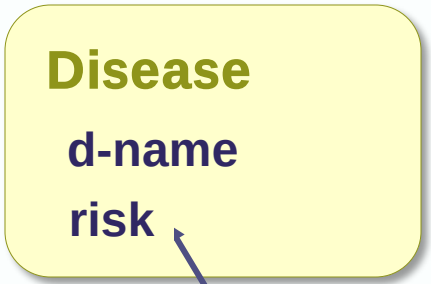
$$p(X_1=x_1,X_2=x_2,X_3=x_3,X_4=x_4,X_5=x_5)$$
$$= p(x_1)p(x_2|x_1)p(x_4|x_2,x_3)p(x_3)p(x_5|x_3)$$

PRM（確率関係モデル）[Friedman+'99] – ベイズネットをRDBに埋め込む

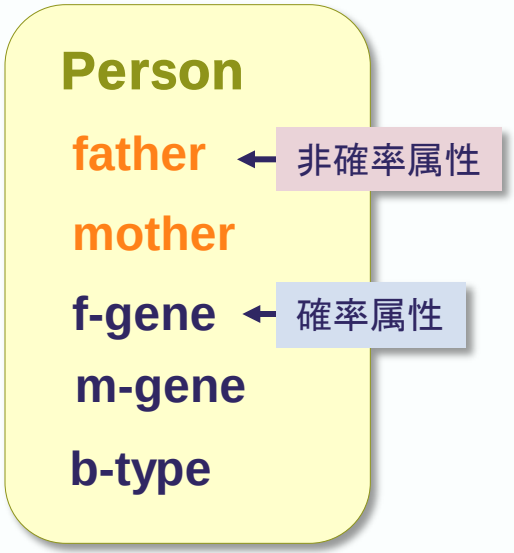
- 関係図式（schema）
  - オブジェクトとそれらの関係を記述する語彙
    - クラス（class）
    - 属性（attribute）
    - 関係（relaiton）



関係

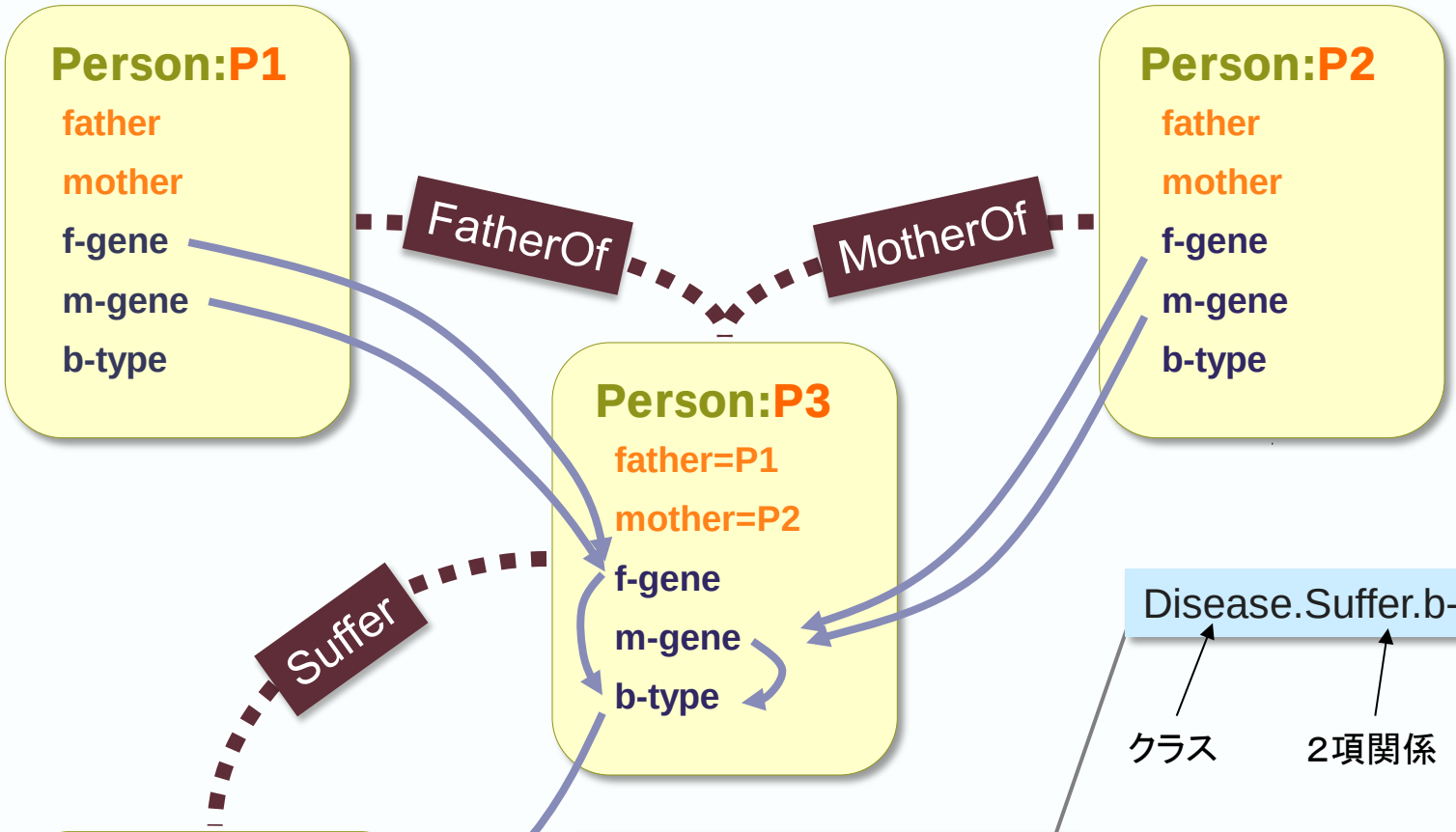


属性



クラス（表に対応）

ベイズネット生成例



Disease.Suffer.b-type

クラス　2項関係　属性

d-name, b-type	P(r | d, b)
Cholera, A	middle
Cholera, B	middle
Cholera, O	high
Cholera,AB	low

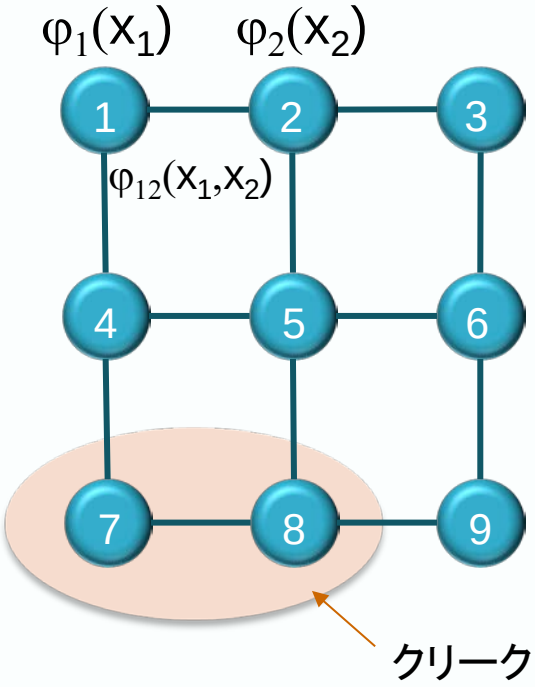
x.rはオブジェクトxと関係 r にある属性値の集合を表す
⇒ **max(x.r)** などの集合関数も利用できる

from
http://www.dadamo.com/

"Eat Right For Your Type"
Peter J. D'Adamo, N. D. 1996.

MRF（マルコフ確率場）

Ising モデル



- $p(x) \propto \Pi_\alpha f_\alpha(x_\alpha)$, α: clique ($x_{\{1,2\}} = \{x_1, x_2\}$)
  $x_i = +1, -1$
- 無限グラフでは相転移を持つ

$$p(x) = Z^{-1} \varphi_1(x_1) \varphi_{12}(x_1, x_2) \varphi_2(x_2) \cdots$$
$$= Z^{-1} \exp - E$$
$$E = \sigma_1 x_1 + \sigma_{12} x_1 x_2 + \sigma_2 x_2 + \cdots$$

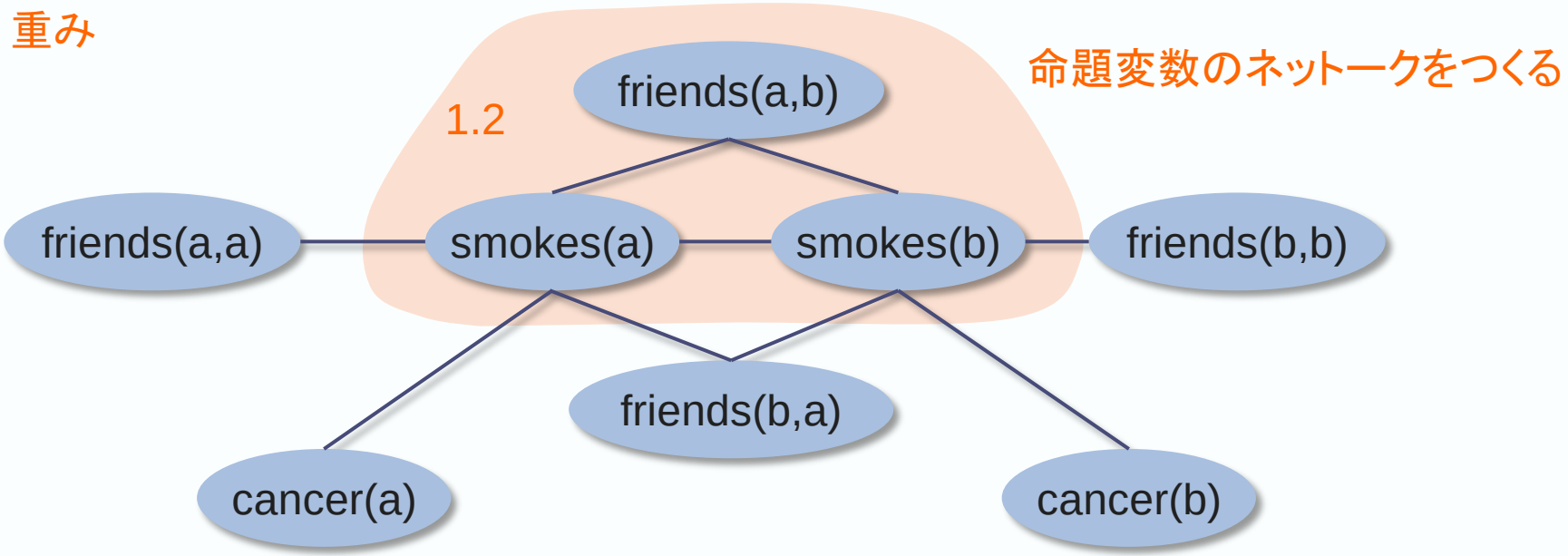
- Log-linear モデル: $p(x) = Z^{-1} \exp(\Sigma_i w_i F_i(x))$
  機械学習，自然言語処理で人気

MLN[Richardson & Domingos'05]

0.5: $\forall x$ smokes(x) $\Rightarrow$ cancer(x)

1.2: $\forall x,y$ friends(x,y) $\Rightarrow$ (smokes(x) $\Leftrightarrow$ smokes(y))

重み



- 領域 = {a, b }
- 基底アトム= 確率的命題変数で {0,1} を取る
- $p(x) = Z^{-1} \exp(\Sigma_i w_i F_i(x))$ where F_i : 節の基底代入例
  x : Herbrand 解釈（$F_i(x) = 1$ if x |= F_i else =0）

MLNの推論と学習

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(x)\right)$$

$n_i(x)$: 開式F_i の基底代入例の内，可能世界 x で成り立つものの数

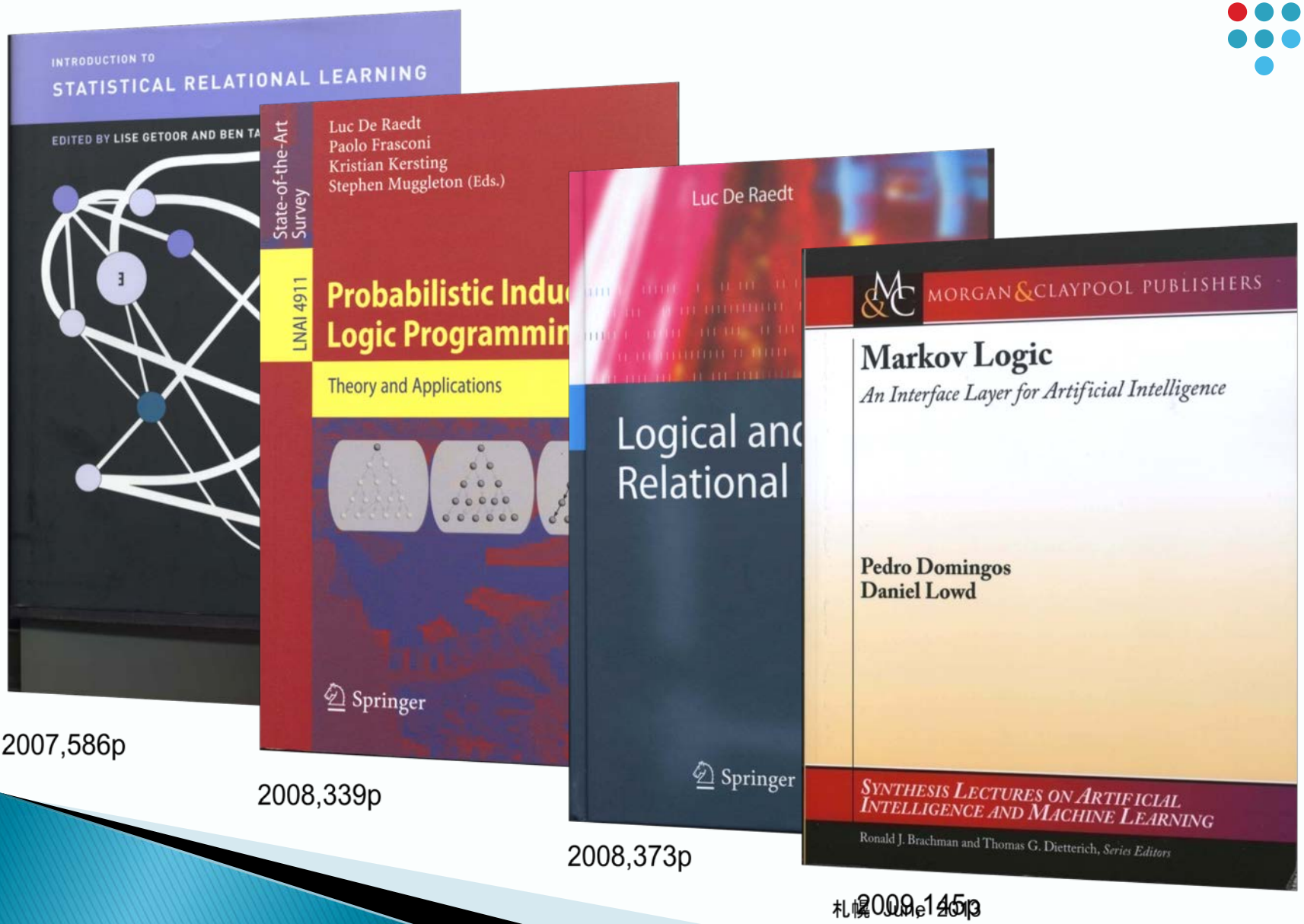
- 正規化定数 Zは計算困難
- 確率推論:
  - MPE 推論
    - MaxWalkSAT を使用
  - 条件付き確率計算
    - MC-SAT (slice sampling+SampleSAT)
  - 改良・発展形
    - LazySAT, Lazy-MC-SAT:必要に応じてMNLに展開
    - CPI (cutting plane inference) [Riedel, UAI-08]:整数線形計画法に基づく
- 重み学習
  - Markov blanket に基づく疑似尤度を山登り
  - 判別タスクはvoted perceptron, scaled conjugate gradient, 準ニュートン法

入力(smoke(a))　出力(cancer (a))

$$\mathrm{argmax}_y P(x, y) = \mathrm{argmax}_y \frac{1}{Z} \exp\left(\sum_i w_i n_i(x, y)\right)$$

$$= \mathrm{argmax}_y \sum_i w_i n_i(x, y)$$

SRL/PLL教科書



2007,586p

2008,339p

2008,373p

2009,145p

札幌 June 2013

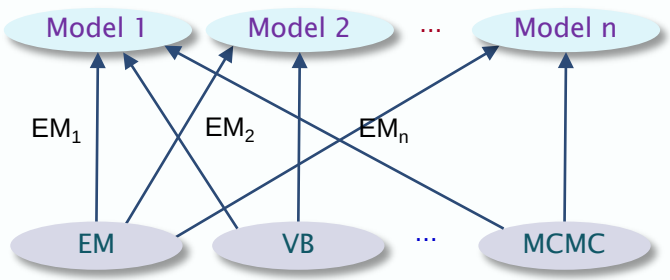
今のAIに欠けているもの

Table 1: Examples of interface layers.

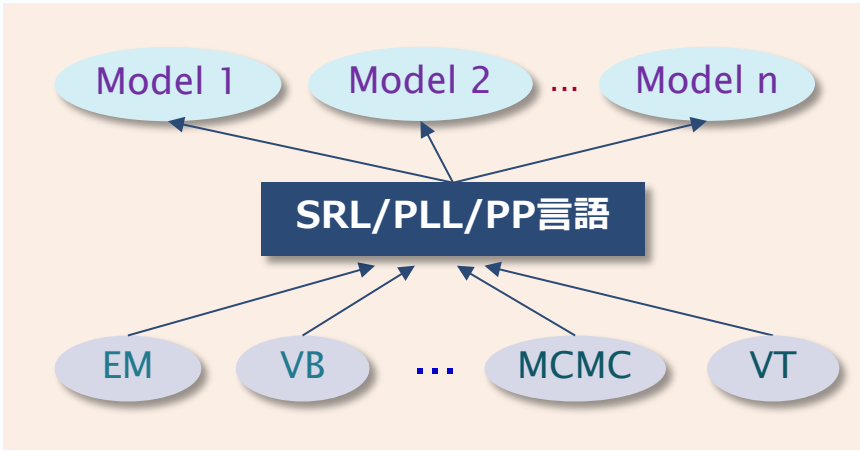
Field	Interface Layer	Below the Layer	Above the Layer
Hardware	VLSI design	VLSI modules	Computer-aided chip design
Architecture	Microprocessors	ALUs, buses	Compilers, operating systems
Operating systems	Virtual machines	Hardware	Software
Programming systems	High-level languages	Compilers, code optimizers	Programming
Databases	Relational model	Query optimization, transaction mgmt.	Enterprise applications
Networking	Internet	Protocols, routers	Web, email
HCI	Graphical user interface	Widget toolkits	Productivity suites
AI	???	Inference, learning	Planning, NLP, vision, robotics

From
Domingos, P.: What's Missing in AI: The Interface Layer. In: Cohen, P. (ed.) Artificial Intelligence: The First Hundred Years, AAAI Press (2006)

言語界面が求められている



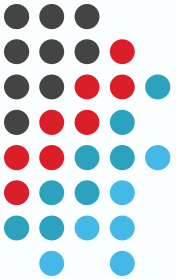
Old style



- Eisnerの指摘：models are too big now

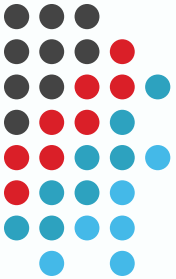
- モデル毎の一品生産 ➔ 高級言語を界面としたモデリング（上流）と計算・学習(下流)の分離

確率プログラミングの流れ



- 確率処理のプリミティブを持ったプログラミング言語
  - 論理型　ICL,PRISM,ProbLog,PITA
  - 関数型　BLOG,IBALL,Church
  - オブジェクト指向　Figaro
  - 手続き型　Dyna,Factorie,Infer.NET
- 実現方式の違いがある
  - Library[Factorie,Figaro] か独立言語[PRISM,Dyna,Alchemy]か
  - 汎用の計算・学習アルゴリズムが組み込み [PRISM,MLNs]か否か[R,Dyna,BLOG]
- PPAML by DARPA (2013) ➔機械学習の「民主化」

中間のまとめ



- 北米ではグラフィカルモデルの拡張として関係概念を機械学習に，EUや日本では論理プログラミングに確率概念を2000年代に取り入れた
- 機械学習の大規模化に伴い界面としての確率言語の必要性が認識されつつある
- 総じて論理式を使うことはあるものの論理と確率の統合はadhocなものが多かった
- 意味論レベルからの統合の試みをPRISM言語を例としてこれから紹介する
  - PRISM ＝ Prolog+統計的機械学習
  - 東工大，CUNY，Roskildeの研究者が連携して開発

論理と確率

- ともに1930年代に別々に整備され発展する
  - 確率P(α)は命題(＝確率事象)αを含む項，P(α)=a は高階の命題
    ➔両者を統合した体系は高い非可解性を持つ
  - 論理の前提(命題は無関係) vs. 統計の前提(RVは独立でない)
- 論理式に勝手に確率を割り振ることはできない
  - $0 \leqq P(\alpha) \leqq 1$, $P(\alpha \vee \beta) = P(\alpha) + P(\beta)$ if α, β 排他的, $P(\sim\alpha) = 1 - P(\alpha)$
  - $P(\alpha) = P(\beta)$ if $\vdash \alpha \Leftrightarrow \beta$, $P(\alpha) = 1$ if $\vdash \alpha$
- PCFGを勝手に論理プログラムに昇格させても確率モデルにならない

0.6: s(X):- p(X),p(X). 0.3:p(a). 0.2:q(a).
0.4: s(X):- q(X). 0.7:p(b). 0.8:q(b).

➔ P(s(a))+P(s(b)) < 1

(from Machine Learning, 44, 245-271, 2001)

  - 導出の失敗を考慮に入れていない
  - 節の確率と中のアトムの確率との関係が不明(合成意味論なし)
- 一貫した統合の仕方が求められている

表現定理 [Fenstad'67]

Let

L : countable 1st ord. lang. w.o. "="

α : (possibly open) formulas in L

$p(\alpha)$: probability given to α s.t.

$$\begin{cases} p(\alpha) = p(\beta), \text{ if } \vdash \alpha \Leftrightarrow \beta \\ p(\alpha) = 1, \text{ if } \vdash \alpha \end{cases}$$

Then

$$p(\alpha) = \int_S p_M(\alpha(M))\, dP(M)$$

where

S : set of H-interpretations M for L'

L' : L with "special constants"

P : σ-additive prob. measure on S

p_M : prob. measure on $\{\alpha(M) \mid \alpha \text{ in } L\}$

$\alpha(M)$: $\{a \in [I \mapsto \mathrm{Dom}_M] \mid M \models_a \alpha\}$

I: index of variables (= nat. nums)

- **可能多世界意味論:**
  閉論理式 αに対し p(α) は αを真にする世界の確率の和

$$p_M(\alpha(M)) = 1 \text{ if } M \models \alpha$$
$$= 0 \text{ o.w.}$$

- **頻度解釈:**
  α が自由変数xを持つとき $p_M(\alpha(M))$ はMの人口とαである個体の比に成っている

Representations of Probabilities Defined on First Order Languages, J.E.Fenstad, Set, Models and Recursion Theory, J.E.Crossley (ed.), North-Holland, pp156-172, 1967.

可能多世界と確率

(x_1,x_2)	Prob. $M_{1,2,3}$
(a,a)	0.3
(a,b)	0.1
(b,a)	0.2
(b,b)	0.4

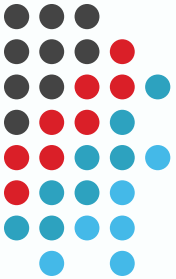
S	R(a,a)	R(b,b)	R(a,b)	R(b,a)	Prob.
M_1	f	t	f	f	0.2
M_2	t	f	f	t	0.3
M_3	f	f	t	f	0.5

その他の世界の確率は0

- Herbrand universe = {a,b}
- 可能多世界 = $\{M_1,M_2,M_3\}$ = H解釈達（基底アトムへの真偽の割り当て）
- 左図：列の出現確率宇，右図：可能世界の確率

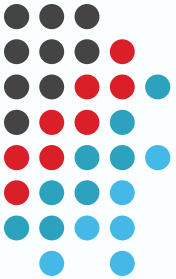
$$P(\exists x_2\ R(a,x_2)) = P(M_1)+P(M_2) = 0.2+0.3 = 0.5$$
$$P(\forall x_2\ R(a,x_2)) = 0.0$$
$$P_{M1}(\neg\ R(a,x_2)) = P_{M1}(\{\ (a,a),(a,b)\ \}) = 0.3+0.1 = 0.4$$

分布意味論[Sato'95]



- Fenstadの定理の（前半部の）論理プログラムへの適用：単純な確率分布を論理プログラムの最小モデル意味論により拡大し，可能多世界の確率分布を定める（最小モデルと確率論の統合）
- プログラム DB = F ∪ R
  - F：定められた述語記号を持つ基底アトムの全体
  - R：定節の集合，ただしF のアトムは説のボディにのみ出現
  - $P_F(\ )$：有限分布の無限積からなるFのHerbran解釈の確率測度
- $P_F(\ )$ を最小モデル意味論とKolmogorovの拡張定理を使って，DBのHerbran解釈（可能多世界）の完全加法的確率測度$P_{DB}(\ )$に拡大
  - F' ~ P_F：$P_F(\ )$からsampleされた基底アトム達
  - M(R ∪ F')：R ∪ F' の最小モデルが一意に定まる
    ➔ Herbran解釈を値とするrandom vectorが定まる
  - $P_{DB}(\ )$：M(R ∪ F') から誘導された確率測度
- 現在PLLの標準的意味論（PRISM,ProbLog,LPAD,PITA）

PRISM [Sato+'97]

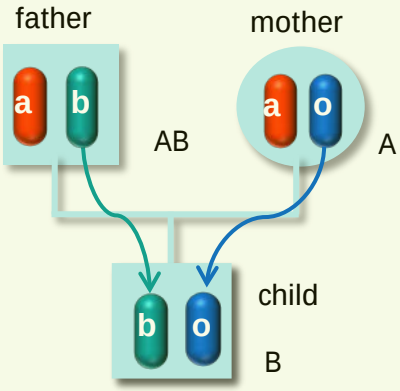


- 分布意味論の一つの実装
- Titech(学習),CUNY(B-Prolog),Roskilde(応用)の連携で開発
- 構文: Prolog + msw/2 (確率的選択を行う, abducibles)
  - F = 基礎 msw アトムと基礎分布 $P_F(\cdot|\theta)$
    R = Prolog節達 (頭に msw アトムを持たない)
    DB = F U R + 宣言文 for $P_F(\cdot|\theta)$
    - 効率的計算のためDBに幾つかの条件を仮定している
- 述語論理でモデルを定義, 命題論理で確率計算と学習を行う
  ➔ 宣言的確率モデリングと効率的確率推論を両立させる
- 実装の特徴:効率的探索のためのテーブリング機構と確率学習のための命題論理レベルの単一学習アルゴリズムを備える
- 普遍性:生成的離散確率モデル(BN,PCFGなど)を統合している
  且つ専用推論アルゴリズムを含め吸収している
  (非ベイズ, ベイズを含め)他にない豊富な統計推論機能を提供

ABO 血液型プログラム

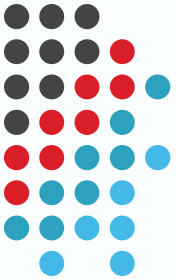
```
values(abo,[a,b,o],[0.5,0.2,0.3]).

btype(X):- gtype(Gf,Gm), pg_table(X,[Gf,Gm]).

pg_table(X,Gtype):-
   ((X=a;X=b),(GT=[X,o];GT=[o,X];GT=[X,X])
   ; X=o,GT=[o,o]
   ; X=ab,(GT=[a,b];GT=[b,a])).

gtype(Gf,Gm):- msw(abo,Gf),msw(abo,Gm).
```

両親からの遺伝は独立



述語論理レベルでモデルを定義し，命題論理レベルで確率計算を行う

確率の命題化計算
– 分布意味論に基づいた確率計算



iff(DB) |= btype(a)
<=> gtype(a,a) v gtype(a,o) v gtype(o,a)
gtype(a,a) <=> msw(abo,a) & msw(abo,a)
gtype(a,o) <=> msw(abo,a) & msw(abo,o)
gtype(o,a) <=> msw(abo,o) & msw(abo,a)

最小モデルで成立している

観測アトムをmsw述語の
AND/ORに還元したものを
説明グラフと呼ぶ

P(btype(a)) = P(gtype(a,a)) + P(gtype(a,o)) + P(gtype(o,a))
P(gtype(a,a)) = P(msw(abo,a))P(msw(abo,a))

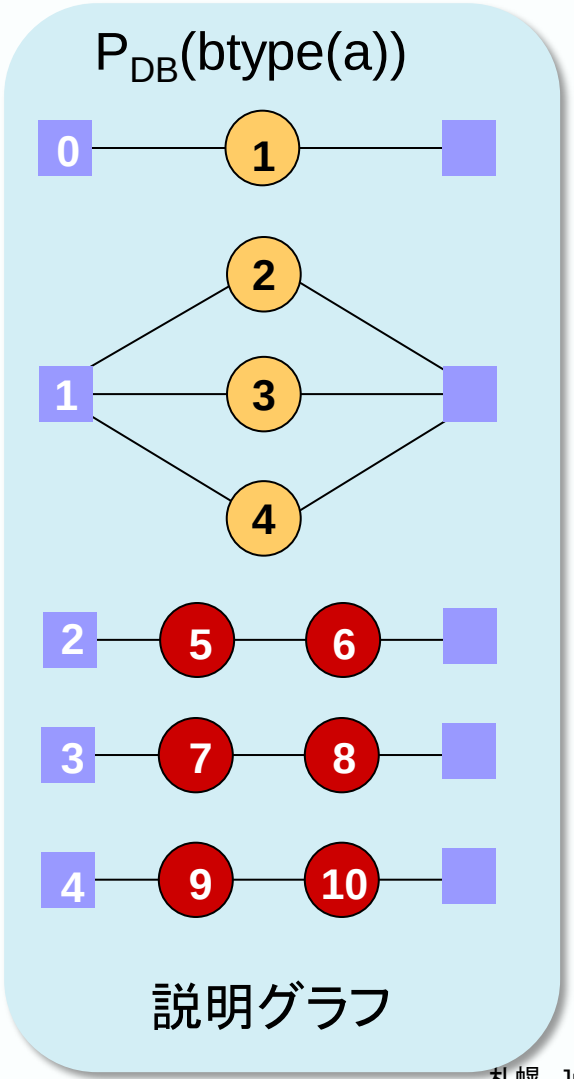
分布意味論では，両辺は同一の
確率変数なので等号が成立，但し
背反生，独立性の仮定を使う

btype(a)$^{0.55}$
<=> gtype(a,a)$^{0.25}$ v gtype(a,o)$^{0.15}$ v gtype(o,a)$^{0.15}$
gtype(a,a)$^{0.25}$<=> msw(abo,a)$^{0.5}$& msw(abo,a)$^{0.5}$
gtype(a,o)$^{0.15}$<=> msw(abo,a)$^{0.5}$& msw(abo,o)$^{0.3}$
gtype(o,a)$^{0.15}$<=> msw(abo,o)$^{0.3}$& msw(abo,a)$^{0.5}$

説明グラフを使い，内側確率を
下の層から計算する

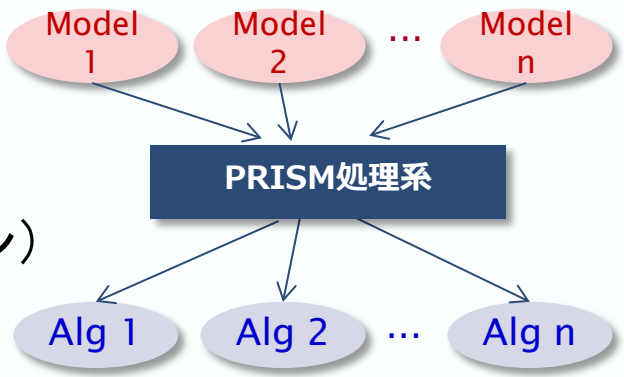
効率の追求
– ダイナミックプログラミング

- 説明グラフは細かい部分グラフの集まり ➔部分グラフを共有すれば部分計算も共有される
- PRISMはテーブリングを使い初めから部分グラフが共有されている説明グラフを作る
  - テーブリングは答えを保存し後で再利用するメカニズム
- ダイナミックプログラミングで部分計算を共有しつつ下から上へ確率計算を行う
- BP, forward-backward等既存アルゴリズムを統合している



PRISM処理系：多様な機能

- PRISMの機械学習機能（拡張中）
  1. Forward サンプリング
  2. 確率計算（周辺確率，条件付き確率）
  3. Viterbi 推論（最尤の説明，アブダクション）
  4. Hindsight 計算（スムージング）
  5. 最尤推定（EM），MAP 推定
  6. 変分ベイズ学習（VB）[Sato+ '09]
  7. MCMCによるベイズ推論（周辺尤度推定，予測分布推定）
  8. Viterbi学習（VT），VB-VTによるパラメータ学習
  9. モデルスコア: BIC, Cheeseman-Stutz スコア，変分自由エネルギー
  10. D-PRISM：判別モデル（regression, linear-chain CRF)の推論
  11. 疑似乱数（一様，正規），統計量（平均，分散，etc.）の組み込み述語

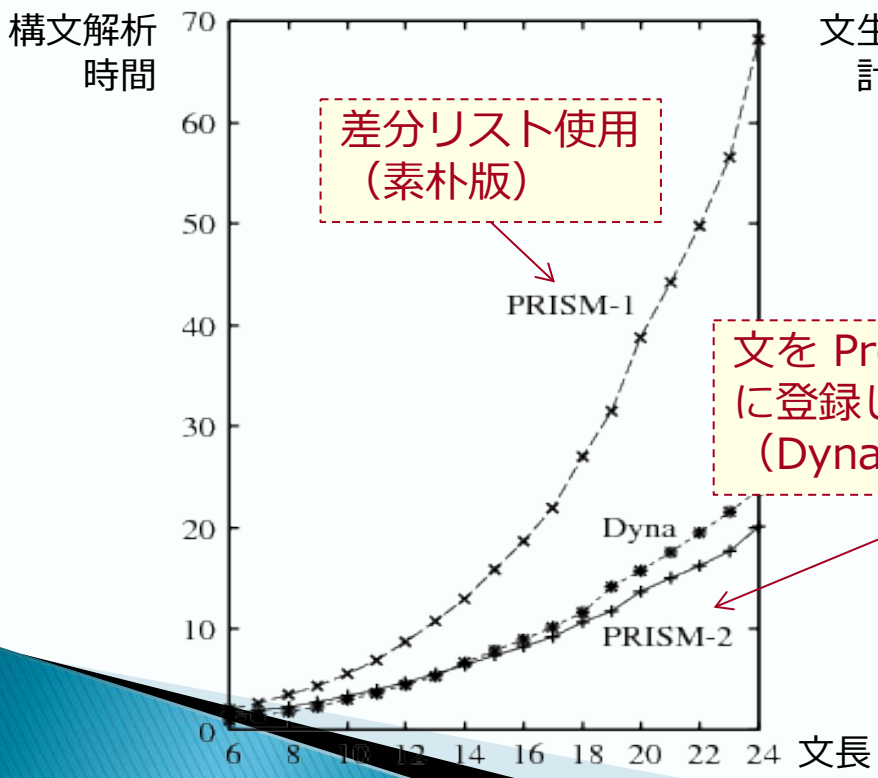


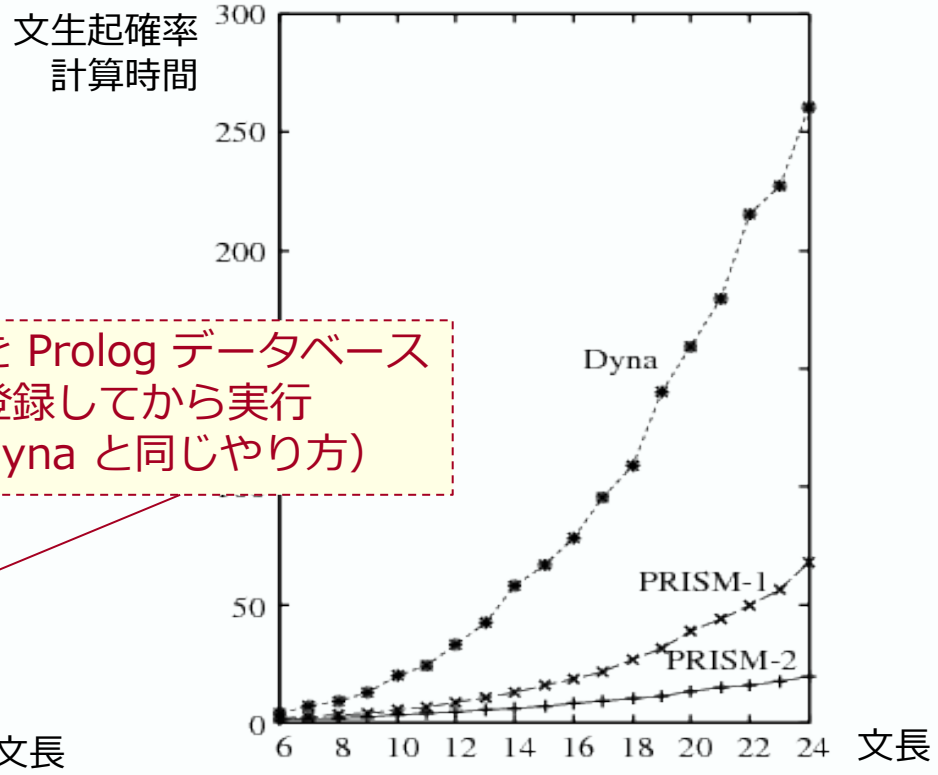
既存の確率推論アルゴリズムと同オーダの計算量

いったんモデルを記述すれば，ユーザはこれらの機能をすべて使える

PRISM処理系：性能比較

- Penn Treebank における構文解析 (Viterbi)・文生起確率計算 [亀谷ら, 07]
  - 比較対象: Dyna [Eisner et al., HLT/EMNLP-05]
    - ボトムアップチャートパーザ（優先度付きキュー使用）に基づく動的計画法アルゴリズム記述言語およびその処理系
    - アルゴリズム記述を C++ プログラムに翻訳して実行

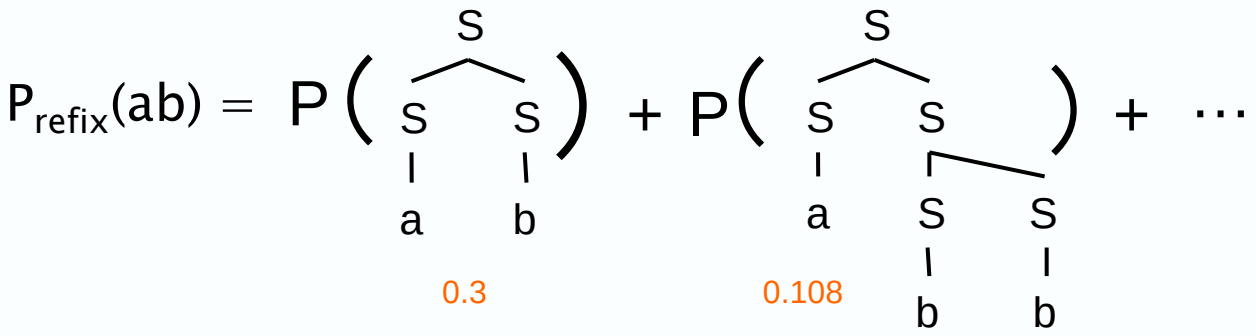




確率の無限和

- 接頭辞(prefix)u の確率は無限和:$\Sigma_{uw}P(uw)$ 但しuwは文
  - 確率文脈自由文法

    S ➔ a:0.5 | b:0.3 | S S:0.2



  - プラン認識に応用
- ループがあるマルコフ連鎖の到達確率は無限和
  - モデルチェッキングに出現
- 循環的説明グラフにより統一的に計算される

循環的説明グラフ

- 事象(ゴール) ➔ 説明グラフ ➔ 方程式 ➔ 確率

$PCFG_2$
S ➔ a:0.4 | b:0.3 | S S:0.2 | S:0.1

```
values(s,[[a],[b],[s,s],[s]],
        set@[0.4,0.3,0.2,0.1]).

pre_pcfg(L):-
    pre_pcfg([s],L,[]).
pre_pcfg([A|R],L0,L2):-
   ( nonterminal(A) ➔
       msw(A,RHS),
       pre_pcfg(RHS,L0,L1)
   ;   L0=[A|L1] ),
   ( L1=[] ➔ L2=[]
   ; pre_pcfg(R,L1,L2) ).
pre_pcfg([],L,L).
```

接頭辞プログラム

?- prprob(pre_pcfg([s],[a,b],[]),P)

⬇ テーブル探索

循環説明グラフ

```
pre_pcfg([s],[a,b],[])
   ⇔ pre_pcfg([s,s],[a,b],[]) & msw(s,[s,s])
     v pre_pcfg([s],[a,b],[]) & msw(s,[s])
pre_pcfg([s,s],[a,b],[])
    ⇔ pre_pcfg([a],[a,b],[b]) & pre_pcfg([s],[b],[]) &
       msw(s,[a])
     v pre_pcfg([s,s],[a,b],[]) & msw(s,[s,s])
     v pre_pcfg([s],[a,b],[b]) & pre_pcfg([s],[b],[]) &
       msw(s,[s])
     v pre_pcfg([s],[a,b],[]) & msw(s,[s])
  …
```

⬇ 確率方程式を解く

P = 0.05

強連結成分 (SCC)

- SCCは半順序をなす ➔ DP が可能

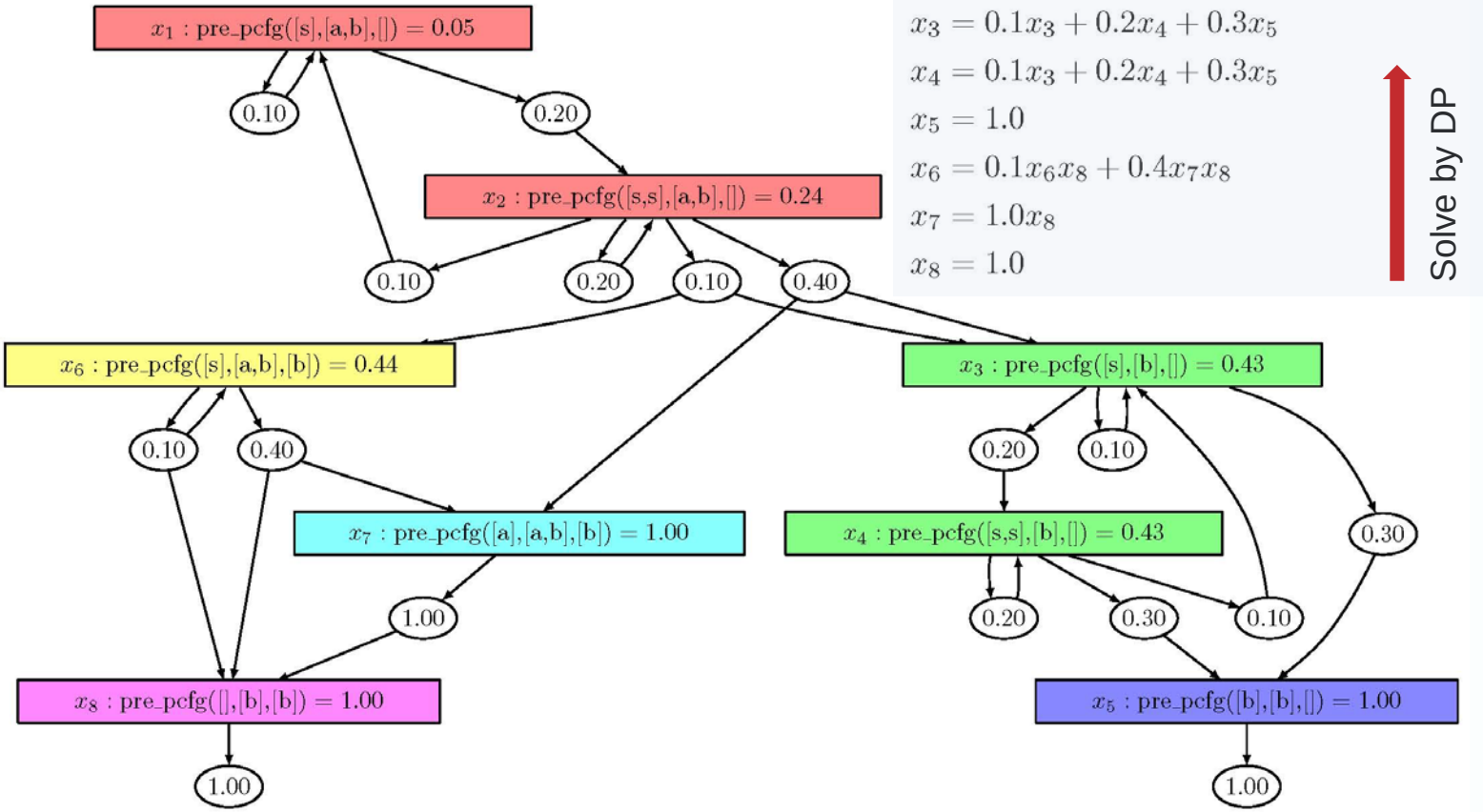
SCC

SCC

SCC

pre_pcfg([s],[a,b],[])
  ⇔ pre_pcfg([s,s],[a,b],[]) & msw(s,[s,s]) v pre_pcfg([s],[a,b],[]) & msw(s,[s])
pre_pcfg([s,s],[a,b],[])
  ⇔ pre_pcfg([a],[a,b],[b]) & pre_pcfg([s],[b],[]) & msw(s,[a])
   v pre_pcfg([s,s],[a,b],[]) & msw(s,[s,s])
   v pre_pcfg([s],[a,b],[b]) & pre_pcfg([s],[b],[]) & msw(s,[s])
   v pre_pcfg([s],[a,b],[]) & msw(s,[s])
pre_pcfg([s],[b],[])
  ⇔ pre_pcfg([b],[b],[]) & msw(s,[b]) v pre_pcfg([s,s],[b],[]) & msw(s,[s,s])
   v pre_pcfg([s],[b],[]) & msw(s,[s])
pre_pcfg([s,s],[b],[])
  ⇔ pre_pcfg([b],[b],[]) & msw(s,[b]) v pre_pcfg([s,s],[b],[]) & msw(s,[s,s])
   v pre_pcfg([s],[b],[]) & msw(s,[s])
pre_pcfg([b],[b],[])
pre_pcfg([s],[a,b],[b])
  ⇔ pre_pcfg([a],[a,b],[b]) & pre_pcfg([],[b],[b]) & msw(s,[a])
   v pre_pcfg([s],[a,b],[b]) & pre_pcfg([],[b],[b]) & msw(s,[s])
pre_pcfg([a],[a,b],[b]) ⇔ pre_pcfg([],[b],[b])
pre_pcfg([],[b],[b])

確率方程式

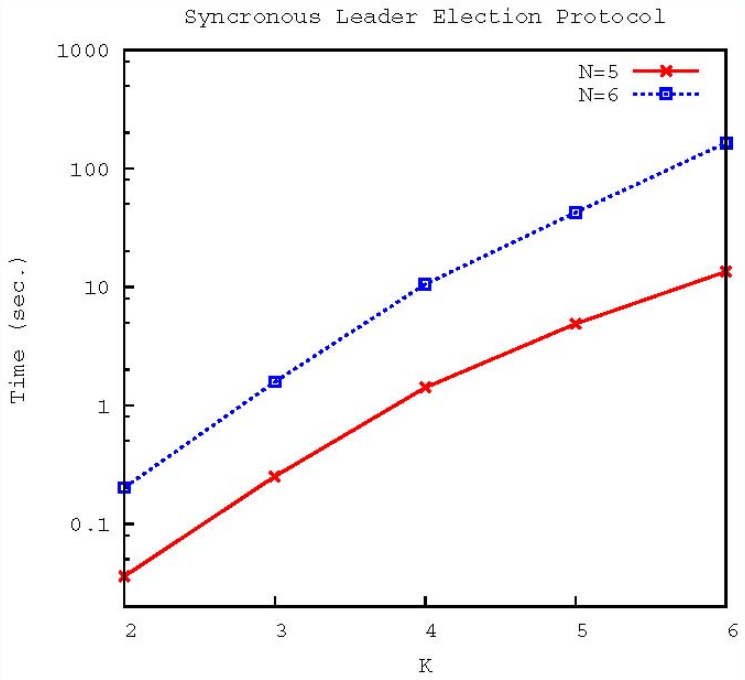


$$x_1 = 0.1x_1 + 0.2x_2$$
$$x_2 = 0.1x_1 + 0.2x_2 + 0.1x_3x_6 + 0.4x_3x_7$$
$$x_3 = 0.1x_3 + 0.2x_4 + 0.3x_5$$
$$x_4 = 0.1x_3 + 0.2x_4 + 0.3x_5$$
$$x_5 = 1.0$$
$$x_6 = 0.1x_6x_8 + 0.4x_7x_8$$
$$x_7 = 1.0x_8$$
$$x_8 = 1.0$$

Solve by DP

プロトコルの検証

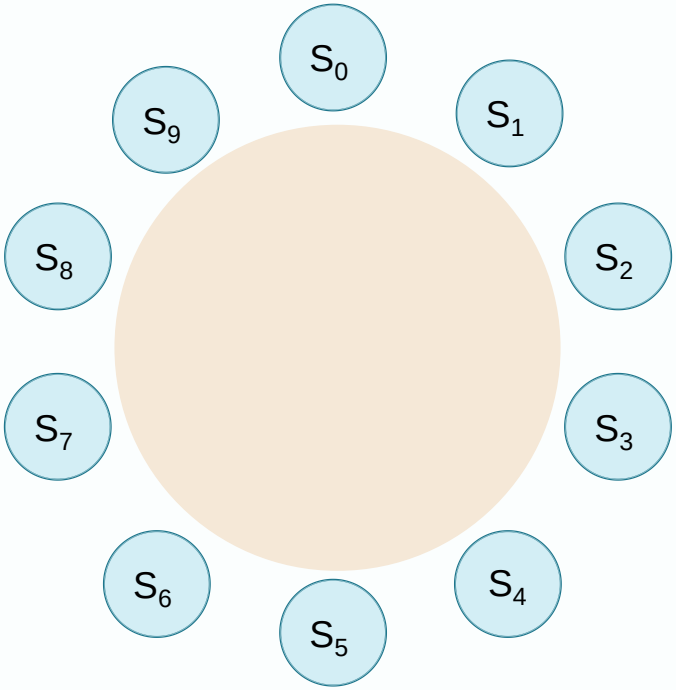
PRISM model checker (http://www.prismmodelchecker.org/casestudies/synchronous_leader.php) - checking reachability of the Synchronous Leader Election Protocol [Itai+ 90] より



N=number of processors
[1..K] = candidate leader ids
CPU: 2.67Ghz

```
values(ini,[0-5],uniform).  % K=6

itai_rodeh(N):- inistate(State0,N), state_tran(State0,IniState,N), reach(IniState,N).
inistate(X,N):-numlist(1,N,Ns),maplist(M,IniPState,( IniPState=[M,0,'*','*',true,1,read(_)] ),Ns,X).
endstate(X):-  ( X==[] ➔ true ; X=[[_,3|_]|Y], endstate(Y) ).
state_tran(State1,State2,N):- ( some_U_true(State1) ➔ SomeU=true ; SomeU=false ),
  rec_maplist(State1,SomeU,State2,Msgs,N),send_Msgs(Msgs,State2).
some_U_true(X):-  ( X=[[_,_,_,_,true|_]|_] ➔ true ; X=[_|Y],some_U_true(Y) ).
rec_maplist([X|X2],SomeU,[Y|Y2],[Msg|Msg2],N):-process_state_tran(X,SomeU,Y,send(Msg),N),
    rec_maplist(X2,SomeU,Y2,Msg2,N).
rec_maplist([],_,[],[],_).
send_Msgs(Msgs,State2):-
    Msgs=[Msg1|Rest],append(Rest,[Msg1],Msgs2),send_Msgs_each(Msgs2,State2).
send_Msgs_each([],[]).
send_Msgs_each([Msg|X],[ProcessState|Y]):-
    ProcessState=[_,_,_,_,_,_,read(Msg)],send_Msgs_each(X,Y).
process_state_tran(X,SomeU,Y,send(Msg2),N):- N_1 is N-1,
  X = [M,S1,PId1,_V1 ,U1,C1,read(Msg1)],
  Y = [M,S2,PId2,Msg2,U2,C2,read(_)],
  ( S1==0 ➔ msw(ini,K), S2=1, U2=true, Msg2=K, PId2=K, C2=1
  ; S1==1, U1==true,  PId1¥==Msg1, C1<N_1
        ➔ S2=1, U2=true,  Msg2=Msg1, PId2=PId1, C2 is C1+1
  ; S1==1, U1==true,  PId1==Msg1,  C1<N_1
        ➔ S2=1, U2=false, Msg2=Msg1, PId2=0,    C2 is C1+1
  ; S1==1, U1==false,           C1<N_1
        ➔ S2=1, U2=false, Msg2=Msg1, PId2=PId1, C2 is C1+1
  ; S1==1, U1==true,  PId1¥==Msg1, C1=N_1
        ➔ S2=2, U2=true,  Msg2=0,    PId2=0,    C2 is C1
  ; S1==1, U1==true,  PId1==Msg1,  C1=N_1
        ➔ S2=2, U2=false, Msg2=0,    PId2=0,    C2 is C1
  ; S1==1, U1==false,           C1=N_1 ➔ S2=2, U2=false, Msg2=0,    PId2=PId1, C2 is C1
  ; S1==2, SomeU==true  ➔ S2=3, U2=false, Msg2=0, PId2=0, C2 is C1
  ; S1==2, SomeU==false ➔ S2=0, U2=false, Msg2=0, PId2=0, C2 is 1
  ; S1==3 ➔ S2=3, U2=false, Msg2=0, PId2=0, C2 is C1
  ; true ).
reach(State1,N):- ( endstate(State1) ➔ true ; state_tran(State1,State2,N), reach(State2,N) ).
```

皿廻せゲーム



- スパゲティが一皿S_0の前にある
- S_iは自分が最後の人ならば皿をget
- そうでなければ0.5の確率で右か左の人に皿を廻す
- どこに座ると有利か？

```
values(clockwise,[yes,no],set@[0.5,0.5]).

win_dish(X):-            % X wins the dish
    filter_not(X,[1,2,3,4,5,6,7,8,9],R), win_dish(0,X,R).

win_dish(Y,X,R):-     % X: last node, Y: current node
   Y ¥== X, filter_not(Y,R,R2),  msw(clockwise,A),
   ( A = yes -> Z is (Y+1) mod 10 ; Z is (Y+9) mod 10 ),
   ( X = Z, R2 = [] -> true
   ; win_dish(Z,X,R2) ).
```

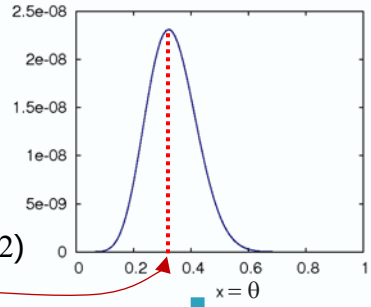
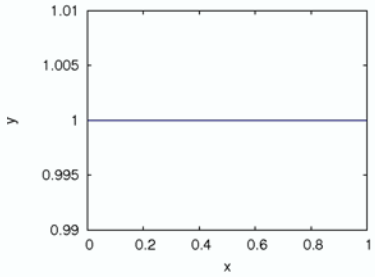
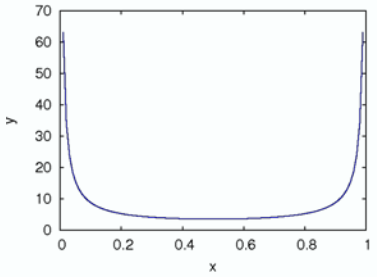
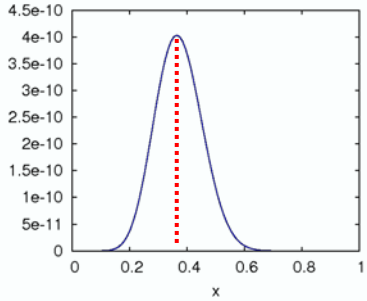
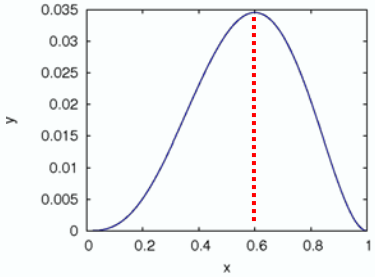
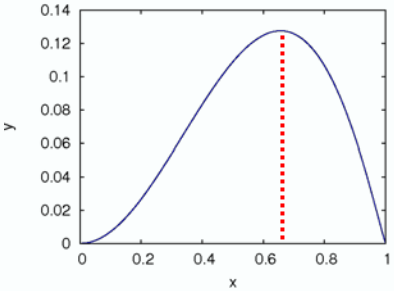
```
?- numlist(1,9,_Xs),maplist(X,P,prprob(win_dish(X),P),_Xs,Ps)
   Ps = [0.111111,0.111111,0.111111,0.111111,0.111111,0.111111,0.111111,0.111111,0.111111]
```

ベイズ推論

- パラメータに事前分布を導入：$P(x,y,\theta)=P(x,y|\theta)P(\theta)$
  - x:隠れ変数 (構文木), y:データ (文)
  - θ:パラメータ, P(θ):事前分布 (信念)
- 事後分布$P(\theta|y) = \Sigma_x P(x,y|\theta)P(\theta)/P(y)$は観測による信念の変化を表す
- データ不足やモデル選択に有効
- 基本課題:
  - 周辺尤度の計算 $P_M(y) = \int \Sigma_x P_M(x,y|\theta)P_M(\theta)d\theta$
    - ➔ モデル選択 (best M = argmax_M $P_M(y)$)
  - 予測分布の計算 $P(x|y_{new},y) = \int P(x|y_{new},\theta)P(\theta|y)d\theta$
    - ➔ Viterbi $x^* = \text{argmax}_x P(x|y_{new},y)$

コイン投げのベイズ推論：

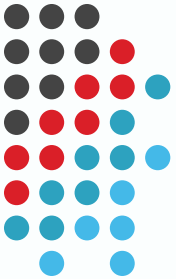
コイン投げ尤度 : $L(\theta) = P(HHHTT \mid \theta) = \theta^3 (1-\theta)^2$ (θ = P(表が出る), パラメータ)
事前分布 : $P(\theta) = \mathrm{Beta}(\alpha,\beta) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}$ (θ は確率変数, α,β, 超パラメータ)
事後分布 : $P(\theta \mid HHHTT) \propto \theta^{3+\alpha-1}(1-\theta)^{2+\beta-1}$ (HHHTT を見た後のθ の分布)

	$\alpha=10,\beta=20$	$\alpha=\beta=1$	$\alpha=0.1,\beta=0.1$
事前分布			
事後分布			

mode θ
$= \alpha-1/(\alpha+\beta-2)$
$= 9/28$

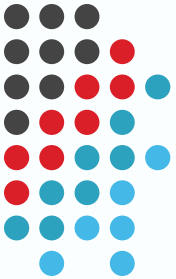
(MLE)

PRISMのベイズ推論



- PRISMではDirichlet分布を事前分布として導入
  - $P_D(\theta_A|\alpha_A) \propto \theta_1^{\alpha_1-1} \cdots \theta_k^{\alpha_k-1}$
- 周辺尤度や予測分布の計算に3つの組み込み述語を提供
  - VB(変分ベイズ)
    - 変分法+DPによる周辺尤度の近似計算，高速
  - VB-VT(VB+Viterbi学習)
    - VBより粗い近似，更に高速
  - MCMC
    - Metropolis-Hastings法による説明のサンプリング，低速
- 特徴
  - モデルによらず適用可能(ベイズ推論アルゴリズムの統合)
  - 他言語にない豊富さ(VB,VB-VT,VT)，手軽さ

後半のまとめ



- 高水準モデリング言語による機械学習の例として PRISM[Sato,Kameya'97]を紹介
  - 意味論，確率計算，論理推論，ベイズ推論を各レベルで統合
  - 述語レベルでモデリング，命題レベルで確率推論，学習を行う
    - 確率事象を探索により説明グラフ(圧縮命題論理式)に還元
    - 説明グラフ上で確率計算，学習アルゴリズムが動く
    - モデリング言語としての汎用性とダイナミックプログラミングによる効率性を実現
- 課題
  - scaleさせるためのデータ構造の自動コンパイル
  - 判別的モデリングとの統合

札幌 June 2013