確率プログラミングの発展と記号的確率モデリング言語PRISM

佐藤泰介・AIRC

統計的機械学習の中心である確率モデルは複雑化するにつれ, 構築・検証が困難になる. この問題を解決するのもとして欧米では確率プログラミング（probabilistic programming）が発展しつつある. 確率プログラミングはベイジアンネットの自然な発展形であり, この講演では日本で開発された論理型の確率プログラミングシステムであるPRISM2.3を使い、幾つかの具体例を紹介する.

略歴

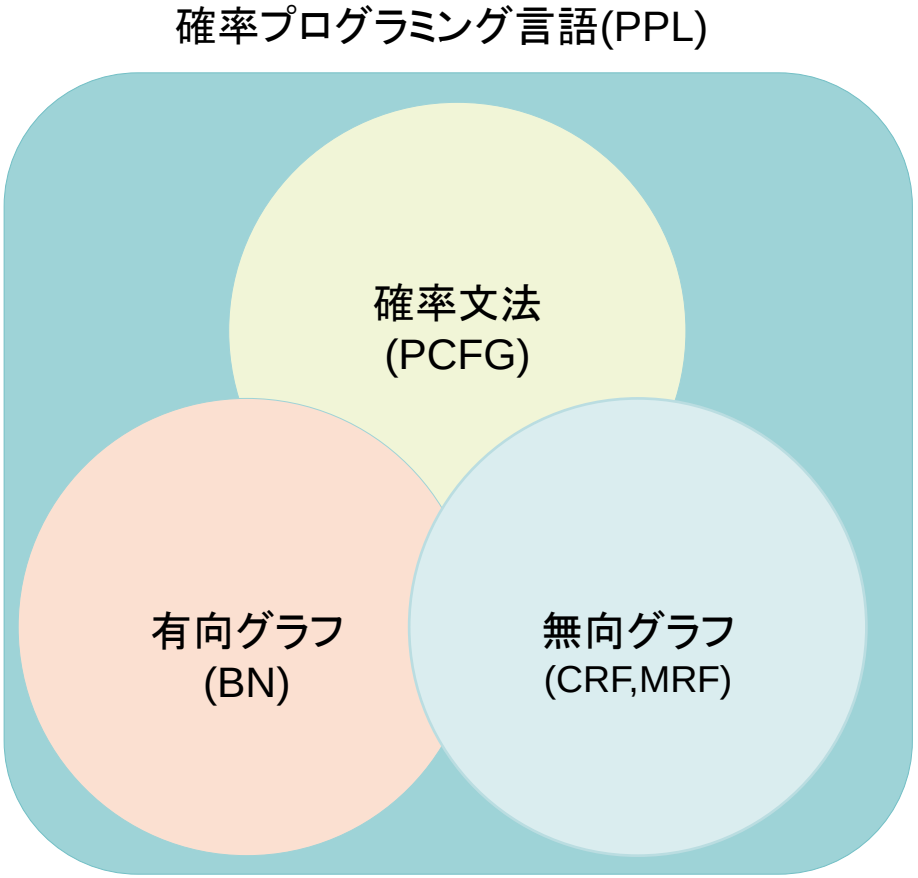
- 1975年東工大大学院修士課程修了, 同年通産省工技院電子技術総合研究所入所
- 1995年より2015年まで東工大大学院情報理工学研究科教授を務め、同年経産省産業技術総合研究所に招聘研究員として移り現在に至る. 工博.
- 論理と確率を融合した柔らかい人工知能の研究に従事.

今日の概要

- 確率プログラミング言語の発展
  - 確率モデリングの世界
  - 確率プログラミング言語
- 記号的確率モデリング言語PRISMによる確率モデリング
  - ナイーブベイズ、ベイジアンネット、CRFによる判別モデリング
  - 混合マルコフ連鎖による系列クラスタリング
  - DistMultモデルによる知識グラフの完備化
  - 確率文脈自由文法による構文解析
  - 接頭辞確率文脈自由文法によるプラン認識
- Edwardによる確率モデリング…

確率モデリングの世界

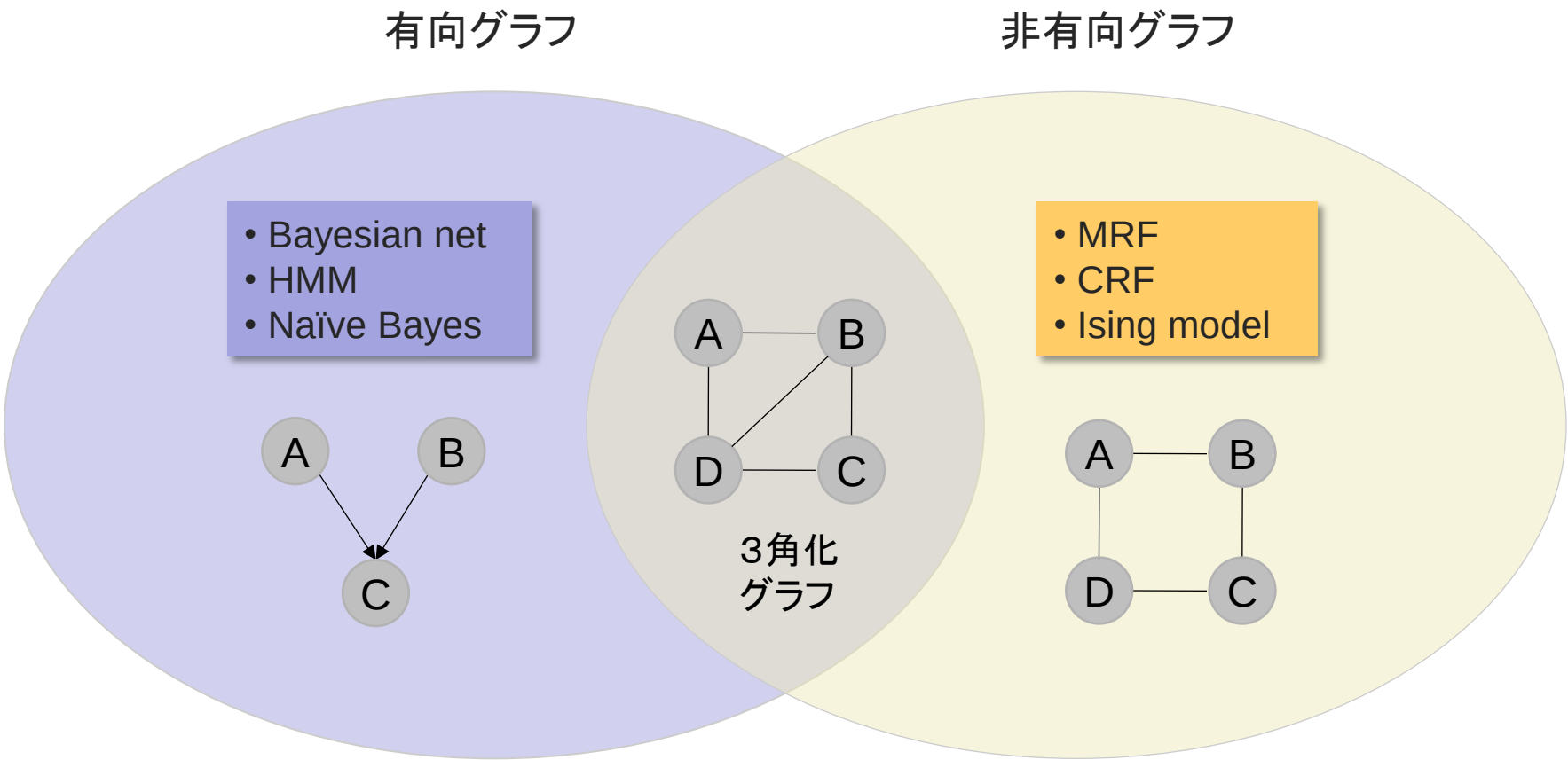
- 統計学の既存の分布を使う
- 確率文法（PCFG)
  - 規則（rule）, 制約（constraint）
    - 再帰あり
    - 生成的
- グラフィカルモデル
  - グラフ表現
    - 再帰なし
    - 生成的(BN), 判別的（CRF)
  - プレート表現
    - 繰り返しあり
    - 生成的
- 確率プログラミング（PP）
  - 再帰, 条件付き分岐, データ構造あり



確率モデリングの今昔

- 素性ベースの機械学習（1990年代～）
  - 決定木,SVM,グラフィカルモデル,DNN
- 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）
  - 深層学習との融合

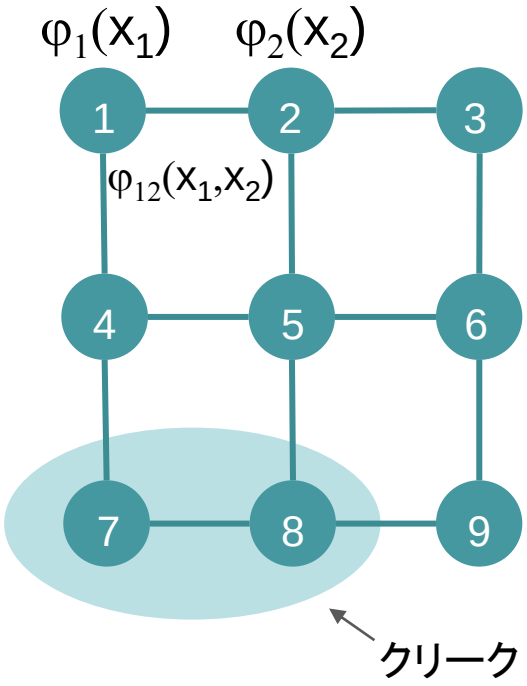
グラフィカルモデル



条件付き独立性の
表現力による分類

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

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

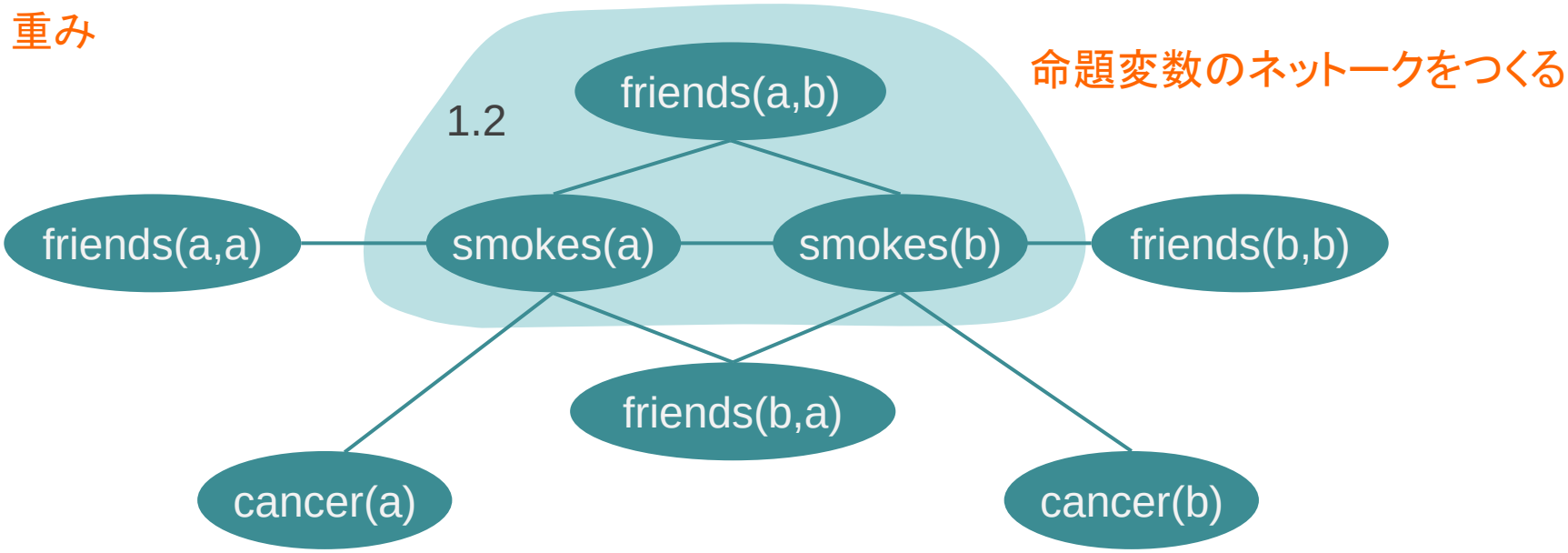
- Log-linear モデル: $p(x) = Z^{-1} \exp(\Sigma_i w_i F_i(x))$
  機械学習，自然言語処理で人気

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 $x \models F_i$ else $=0$）

人工知能ブーム

- 巨大IT企業がAIに取り組んでいる
  - AIプロジェクト: Google Brain, Microsoft Adam, IBM Watson
  - 買収: Deepmind by Google, Face.com by Facebook,…
  - 自動運転,自動キャプション,自動顔認識,…
  - (自動)知識ベース: Freebase, yago, WIKIDATA, KnowledgeVault,…
  - 2017年5月AlphaGo最強棋士に勝利
- 深層学習廻りの動き
  - グラフィカルモデル➔ 関係，論理，複雑システム
  - 研究プロジェクト: PPAML(2013-2017) by DARPA
  - 表現学習: ICLR会議の勃興
- 求められているのは？
  - Interface (P. Domingos )
  - Logic combined with probability (S.Russel )
  - 汎用の界面言語の設計・実装が必要

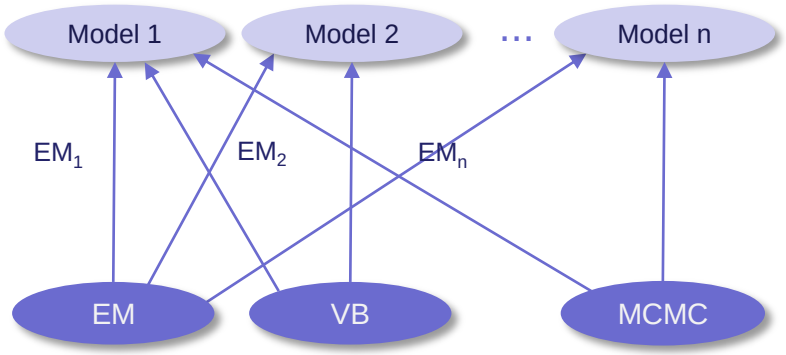
今のAIに欠けているもの

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)

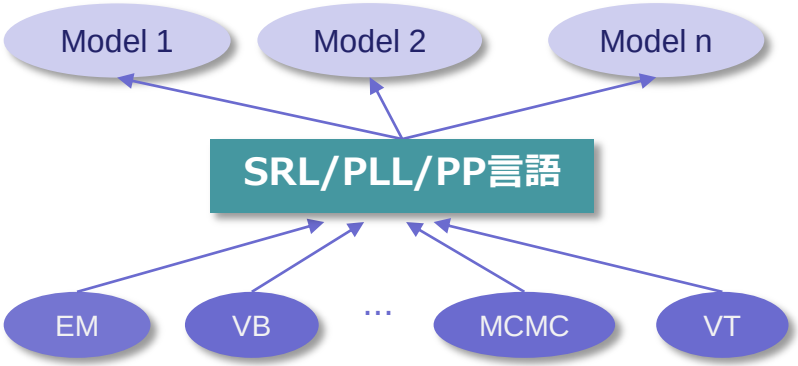
界面が求められている



現状

- 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

界面が求められている



- 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言語

- 確率処理のプリミティブ（サンプリング、確率計算、確率学習、ベイズ推論など）を持った汎用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,Venture
  - 手続き型　Dyna,Factorie,Infer.NET,Stan
  - その他　Edward
- 独自言語　または　ライブラリ
  - PPAML by DARPA (2013-2017) http://www.darpa.mil/program/probabilistic-programming-for-advancing-machine-learning

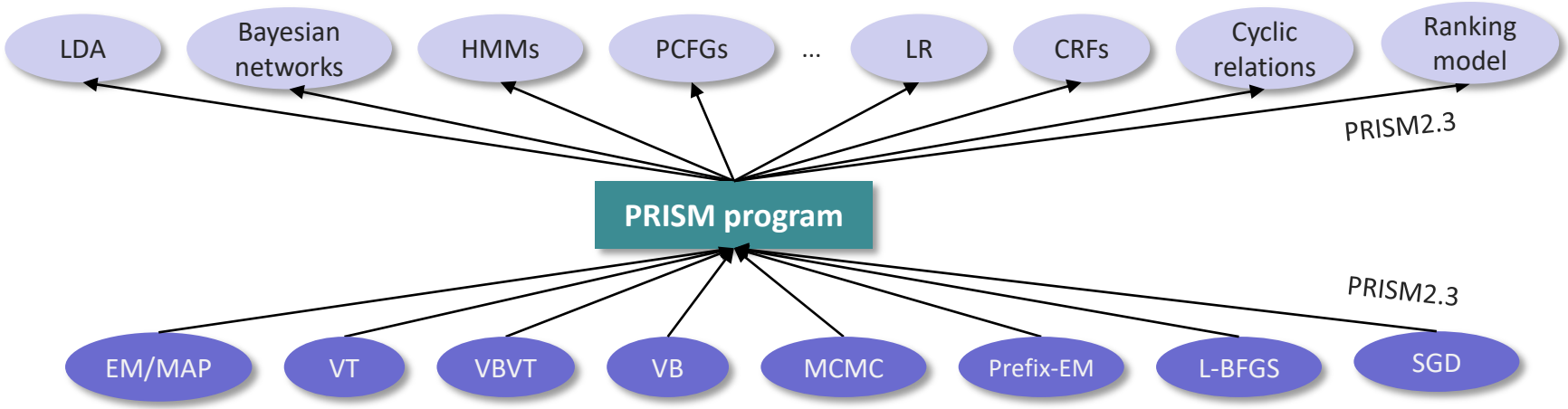
機械学習用高級言語(PPL)

言語	発表年	研究者
PHA/ICL	’93, ’97	D. Poole
PRISM	’95, ’97	T. Sato, Y. Kameya
SLP	’96, ’01	S. Muggleton, J. Cussens
PRMs	’98	D. Koller, N. Friedman
LPADs	‘04	J. Vennekens, S. Verbaeten, M. Bruynooghe
P-log	‘04	C. Baral, M. Gelfond
CFDs	’04	D. McAllester, M. Collins
Dyna	’04	J. Eisner
MLNs	’04	P. Domingos
BLOG	’05	B. Milch, S. Russell
MEBN	’06	K. Laskey
ProbLog/ProbLog2	’07, ’13	L. De Raedt, D. Fierens,…
Church	’08	N. Goodman, J. Tenenbaum
Factorie	’08	A. McCallum
Figaro	’09	A. Pfeffer
PITA	’11	F. Riguzzi
PSL	‘12	A. Kimmig, L. Getoor,…
ProPPR	‘13	W. Y. Wang, K .Mazaitis, W. W. Cohen
Venture	‘14	V. Mansinghka, D. Selsam, Y. Perov
Saul	‘15	P. Kordjamshidi, D. Roth, H. Wu
Edward	‘16	D.Tran,D.M.Blei,..

PRISM [Sato+'97]

- 分布意味論に基づくPrologの機械学習用拡張
- 述語論理でモデルを定義し，命題論理で確率計算と学習を行う
- Titech（学習）,CUNY（B-Prolog）,Roskilde（応用）の連携で開発,産総研AIRCで開発継続
- 普遍性：Turing machine + 確率学習
- 効率的探索機構と汎用学習アルゴリズム
- 記号的モデリング（BN,HMM,PCFG,CRF,Ranking…）
- 豊富な機械学習メニュー(EM,VB,VT,MCMC,…)

PRISM2.3 (http://rjida.meijo-u.ac.jp/prism/)



- PRISM2.2
  - 生成的確率モデリング(LDA, NB, BN, HMM, PCFG,…)
  - 判別的確率モデリング(logistic regression, linear-chain CRF, CRF-CFG,…)
- PRISM2.3(今年8月リリース)
  - 順序学習(Learning to rank by SGD)

ABO 血液型プログラム

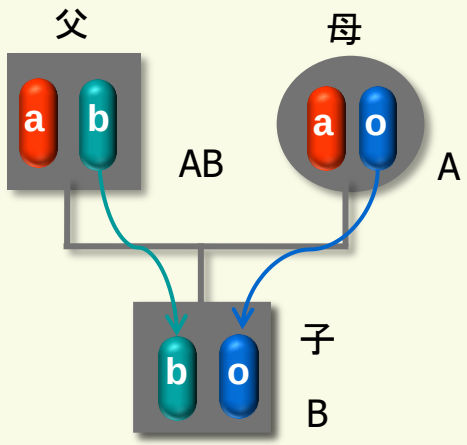
```
values(gene,[a,b,o],[0.5,0.2,0.3]).
```

msw(gene,a) が確率0.5で真

```
btype(P):-
   gtype(X,Y),
   ( X=Y → P=X ; X=o → P=Y ; Y=o → P=X ; P=ab ).

gtype(X,Y):-
   msw(gene,X),msw(gene,Y).
```

父（左）と母（右）からの遺伝子の確率的遺伝をシミュレートする



確率の命題化計算(PPC)

0.55
btype(a)
0.25 0.15 0.15
<=> gtype(a,a) v gtype(a,o) v gtype(o,a)

0.25 0.5 0.5
gtype(a,a) <=> msw(abo,a) & msw(abo,a)

0.15 0.5 0.3
gtype(a,o) <=> msw(abo,a) & msw(abo,o)

0.15 0.3 0.5
gtype(o,a) <=> msw(abo,o) & msw(abo,a)

説明グラフ for btype(a)
describes how it is proved
by probabilistic choice made
by msw-atoms

積和計算 of probabilities in
a bottom-up manner
starting from probabilities
associated with msw
atoms

説明グラフは非循環グラフ
➔ DPが可能

PPC+DP の範囲:
前向き後ろ向き計算,
信念伝播,
内側外側計算

多様なモデリングと学習法

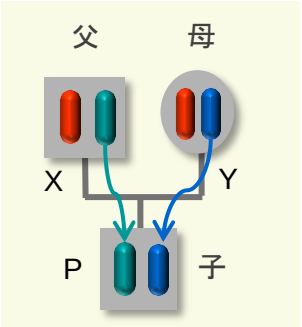
- 頻度論的確率モデル: プログラムは同時分布 $p(x,y|\theta)$ を定義, 但し x は隠れ変数 で y は観測変数 (x から y が生成される)
  - 確率パラメータ θ を y から推定
    - MLE $\theta^* = \mathrm{argmax}_{\theta} P(y|\theta)$ ← EM/MAP
    - $\theta^* = \mathrm{argmax}_{\theta} P(x^*,y|\theta)$ & $x^* = \mathrm{argmax}_{x} p(x,y|\theta^*)$ ← VT
- ベイズ的確率モデル: プログラムは $p(x,y|\theta)p(\theta|\alpha)$ を定義する
  - 周辺尤度 $p(y|\alpha) = \Sigma_x \int p(x,y,\theta|\alpha)\, d\theta$, 事後分布 $p(\theta|y,\alpha)$, 予測分布 $\int p(x|y_{new},\theta,\alpha)p(\theta|y,\alpha)d\theta$ を計算する
    - variational Bayes (VB) ← VB, VB-VT
    - MCMC ← Metropolis-Hastings
- 判別的モデル(条件付き確率場): プログラムは条件付き分布 $p(x|y,\theta)$ を定義する
    - $\theta^* = \mathrm{argmax}_{\theta} p(x|y,\theta)$ ← LBFG
- 順序(選好)モデル:プログラムは同時分布 $p(x,y|\theta)$ を定義する
    - 選好順序 $y_1 \succ y_2$, $y_3 \succ y_4$, ... が与えられたとき 確率パラメータ θ s.t. $p(y_1|\theta)>p(y_2|\theta)$, $p(y_3|\theta)>p(y_4|\theta)$,… を学習 ← SGD

組み込み述語

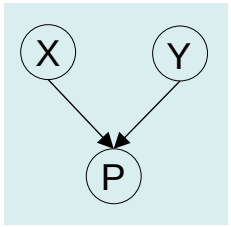
- 前向きサンプリング-- sample, get_samples
- 確率計算 -- prob,probf,…
- パラメータ学習 -- learn (with learn_mode=ml,vb,ml_vt,vb_vt)
  - EM/MAP
  - VT
- ベイズ推論
  - VB
  - VBVT
  - MCMC (by MH) -- mcmc,marg_mcmc_full,predict_mcmc_full,…
- 相互依存モデル
  - prefix-EM -- lin_prob,lin_learn,lin_viterbi,nonlin_prob,…
- 条件付き確率場(CRF) -- crf_prob,crf_learn,crf_viterbi,…
- 順序学習
  - SGD for PRISM -- *rank_learn*,*rank*
- Viterbi推論 (最尤解推論) -- viterbi,viterbig,n-viterbi,…
- モデル基準 (BIC, Cheesman-Stutz, VFE) -- learn_statistics
- スムージング -- hingsight,chindsight

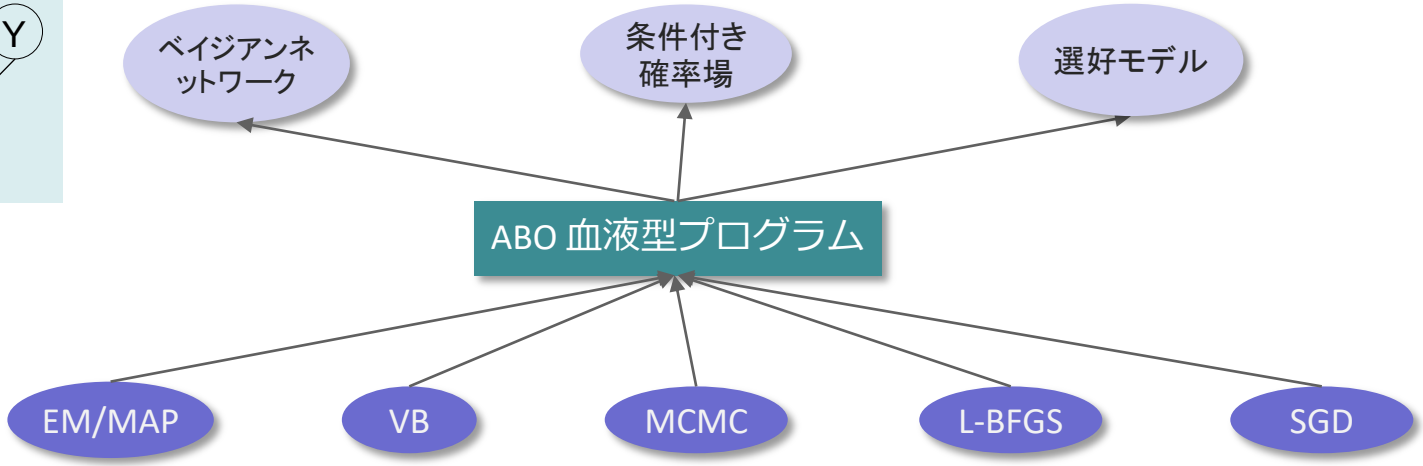
一回プログラムを書けば全部適用できる

例




```
btype(P):-
   gtype(X,Y),
   ( X=Y → P=X ; X=o → P=Y
   ; Y=o → P=X ; P=ab )
gtype(X,Y):-
   msw(gene,X),msw(gene,Y)
```





Sample session 1

- 説明グラフ と 確率計算

blood.psm

```
values(gene,[a,b,o],[0.5,0.2,0.3]).
btype(P):-
    gtype(X,Y),
    ( X=Y -> P=X ; X=o -> P=Y
    ; Y=o -> P=X ; P=ab ).
gtype(X,Y):- msw(gene,X),msw(gene,Y).
```

built-in predicate

```
?- prism(blood)
loading::blood.psm.out

?- show_sw
Switch gene: unfixed_p: a (p: 0.500) b (p: 0.200) o (p: 0.300)

?- probf(btype(a))
btype(a) <=> gtype(a,a) v gtype(a,o) v gtype(o,a)
gtype(a,a) <=> msw(gene,a) & msw(gene,a)
gtype(a,o) <=> msw(gene,a) & msw(gene,o)
gtype(o,a) <=> msw(gene,o) & msw(gene,a)

?- prob(btype(a),P)
P = 0.550
```

Sample session 2
- 最尤推定 と Viterbi 推論

```
?- D=[btype(a),btype(a),btype(ab),btype(o)],learn(D)
Exporting switch information to the EM routine ... done
#em-iters: 0(4) (Converged: -4.965)
Statistics on learning:
        Graph size: 18
        Number of switches: 1
        Number of switch instances: 3
        Number of iterations: 4
        Final log likelihood: -4.965

?- prob(btype(a),P)
P = 0.598

?- viterbif(btype(a))
btype(a)  <= gtype(a,a)
gtype(a,a)  <= msw(gene,a) & msw(gene,a)
```

Sample session 3
- VB と MCMC によるベイズ推論

```
?- D=[btype(a), btype(a), btype(ab), btype(o)], set_prism_flag(learn_mode,vb),learn(D).
#vbem-iters: 0(3) (Converged: -6.604)
Statistics on learning:
        Graph size: 18
        …
        Final variational free energy: -6.604

?- D=[btype(a), btype(a), btype(ab), btype(o)],
marg_mcmc_full(D,[burn_in(1000),end(10000),skip(5)],[VFE,ELM]), marg_exact(D,LogM)
VFE = -6.604
ELM = -6.424
LogM = -6.479

?- D=[btype(a), btype(a), btype(ab),btype(o)], predict_mcmc_full(D,[btype(a)],[[_,E,_]]),
print_graph(E,[lr('<=')])
btype(a)  <= gtype(a,a)
gtype(a,a)  <= msw(gene,a) & msw(gene,a)
```

Sample session 4

- Rank learning of {o≻a, o≻b, b≻ab}

P(btype(o))>P(btype(a)), P(btype(o))>P(btype(b)), P(btype(b))>P(btype(ab))

```
?- show_sw.
Switch gene: unfixed: a (p: 0.5) b (p: 0.2) o (p: 0.3)

?- D=[[btype(o),btype(a)],[btype(o),btype(b)],[btype(b),btype(ab)]],rank_learn(D).
#sgd-iters: 0.........100.......(10000) (Stopped: 0.105)
Graph size: 36
        Number of switches: 1
        Number of switch instances: 3
        Number of iterations: 10000
        Final log likelihood: 0.191

?- show_sw.
Switch gene: unfixed:  a (p: 0.112) b (p: 0.131) o (p: 0.757)
```

➔ P(btype(o))=0.625 > P(btype(a))=P(btype(b))=0.230 > P(btype(ab))=0.020

```
?- rank([btype(ab),btype(o)],X).
X = [btype(o),btype(ab)]
```

Sample session 5

- 条件付き確率場(CRF) : P(G|P)

```
btype(P):- btype(P,_).
btype(P,G):- G=[X,Y],
   gtype(X,Y),
   ( X=Y -> P=X ; X=o -> P=Y ; Y=o -> P=X ; P=ab ).
gtype(X,Y):- msw(gene,X),msw(gene,Y).
```

不完全データ P が観測される

完全データ (P,G) が観測される
最尤の G for P を推定する

```
?- get_samples(100,btype(_,_),D),crf_learn(D),crf_viterbig(btype(a,G)).
#crf-iters: 0#12
(7) (Converged: -92.443)
Statistics on learning:
        Graph size: 36
        Number of switches: 1
        Number of switch instances: 3
        Number of iterations: 7
        Final log likelihood: -92.443
G = [a,a]
```

$G = \mathrm{argmax}_Z P(\mathrm{btype}(a,Z) \mid \mathrm{btype}(a))$

既存言語との比較：MLN

-MLN（判別的）vs. PRISM（生成的）

- UW-CSEデータ
  - 米国ワシントン大学コンピュータサイエンス学部内の教師, 学生, 講義, 論文の関係を示したデータセット．全部で 2,673 個のデータが入っており, 5分野 (AI, graphics, language, systems, theory) に分けられている.
  - 例：taughtBy(Course7, Person415, Spring_0304),
    courseLevel(Course52, Level_400), advisedBy(Person265, Person168)
- 学習実験(MLN)
  - UW-CSEの各分野Xにつき, XからadvisedBy関係を除いたものと残り4分野のデータを学習データとし, XのadvisedBy関係を正解データとした. 以上のデータと94個の関係規則で記述されたMLNを用い, 学習を行ってXに於けるadvisedBy関係を予測した．これを各分野で行った.
  - 例：¬student(x) v advisedBy(x,y) v tempAdvisedBy(x,y),
    ¬advisedBy(x,y) v professor(y),…
- 再現実験(PRISM2.2, Intel(R)Xeon(R) CPU E5-26900@2.90GHz)
  - 例: advisedBy(S,P):-msw(advised_by,X),(X=ta->advisedByTA(S,P);advisedByProf(S,P)).

学習評価方法

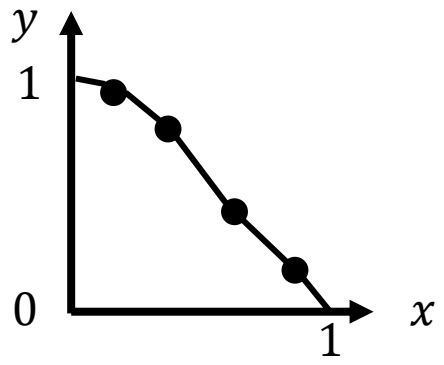
- Precision-Recall グラフ
  - 予測データにおける上位 n 個のデータを考える
    - Recall(再現率)

$$Recall = \frac{\text{上位}n\text{番目内の正解データ数}}{\text{正解データ総数}}$$

    - Precision(精度)

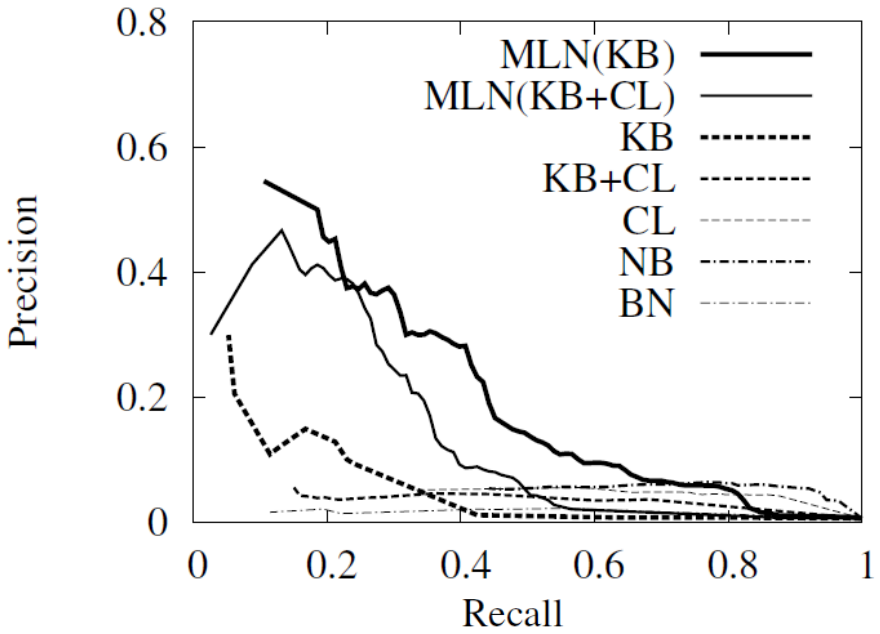
$$Precision = \frac{\text{上位}n\text{番目内の正解データ数}}{n}$$

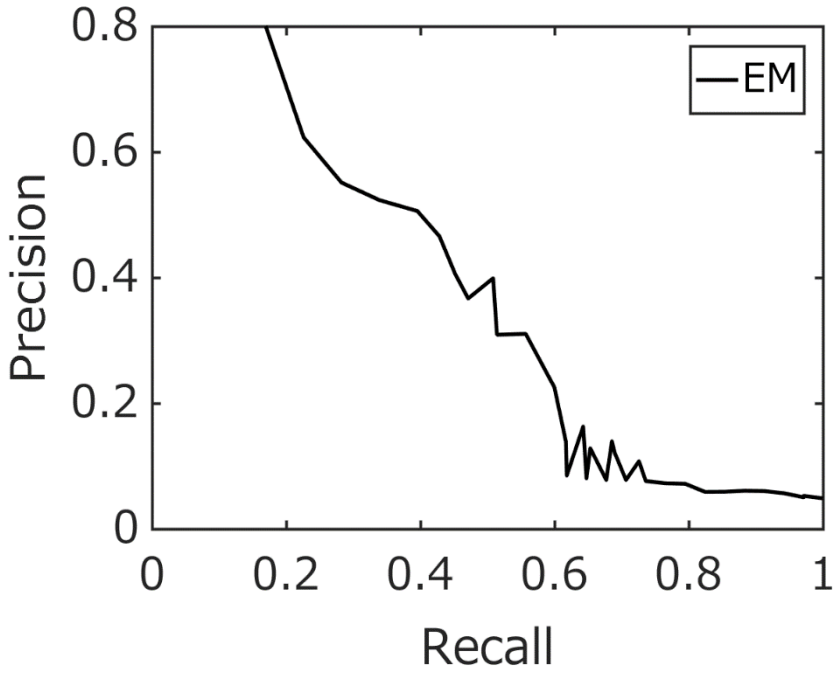
  - 次に上位 n+1 個の場合を考え, これを最後まで行う



Precision-Recall による MLN(左図) と PRISM(右図)の学習結果比較

- VFML
  - MLNなどの学習アルゴリズムが複数組み込まれたシステム





※ “Markov Logic Networks”, Matthew Richardson, Pedro Domingos, Machine Learning, 62, 2006. p.24より抜粋

※ MLB(KB), MLN(KB+CL), KB, KB+CL,CL, NB, BNは学習アルゴリズムを指す

LDA

- LDA (latent Dirichlet allocation) は文書の生成的トピックモデルで PLSI ($p(d,\{w_1,\cdots,w_N\}) = p(d)\prod_{n=1}^{N} p(w_n \mid z_n)p(z_n|d)$) のBayes版
- 文書はN語から成り, 語彙サイズをV, トピック数をKとすると
  - トピックzの分布をDirichlet分布からサンプリング : $\boldsymbol{\theta} = \theta_1 \cdots \theta_K \sim \mathrm{Dir}(\theta|\alpha)$
  - zの語wの分布をDirichlet分布からサンプリング : $\boldsymbol{\eta}_z = \eta_1 \cdots \eta_V \sim \mathrm{Dir}(\theta|\beta, z)$
  - 以下をN回繰り返す
    - トピックzをCategorical分布$z \sim p(z|\boldsymbol{\theta})$からサンプリング
    - 語wをCategorical分布$w \sim p(w|\boldsymbol{\eta}_z)$ からサンプリング

$$\mathrm{Dir}(\theta \mid \alpha) = \frac{\Gamma(\sum_{j=1}^{k}\alpha_j)}{\prod_{j=1}^{k}\Gamma(\alpha_j)}\theta_1^{\alpha_1}\cdots\theta_k^{\alpha_k}, \text{ where } \begin{cases} \theta = (\theta_1,\ldots,\theta_k), \\ \alpha = (\alpha_1,\ldots,\alpha_k) \end{cases}$$

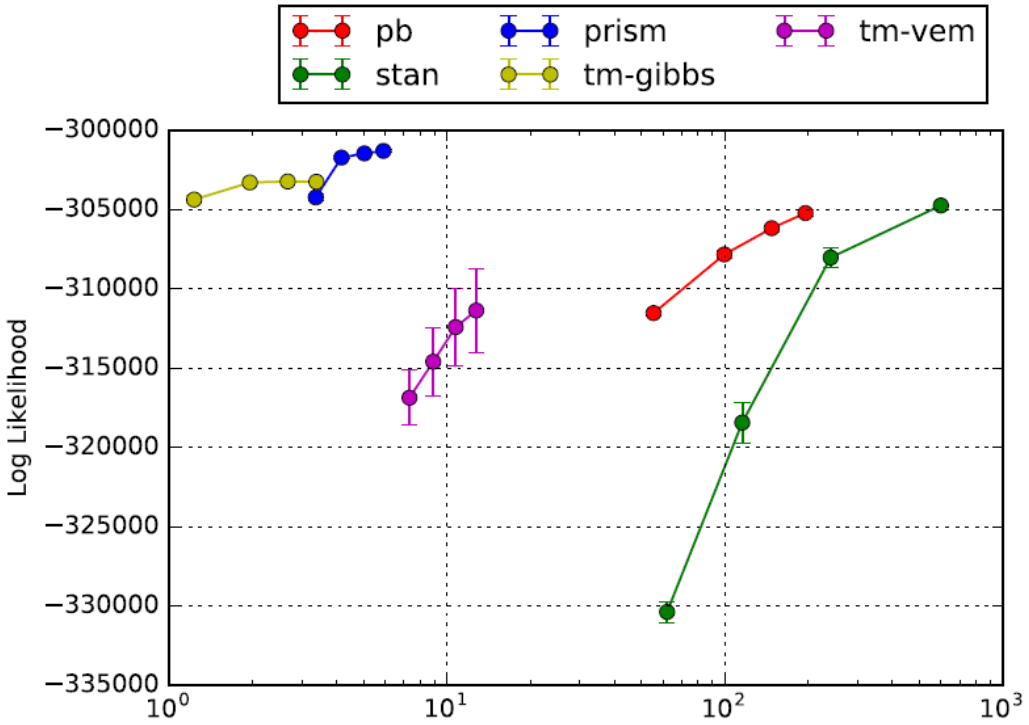
- PRISM program:

```
values(topic,[1,2,..,10]).     % K=10
values(word,[1,2,..,25]).      % V=25
doc(D):- (D=[W|R] -> msw(topic,Z),msw(word(Z),W),doc(R) ; D=[]).
```

既存言語との比較:
StanとR(topicmodels)

- Stan:大規模な手続き型確率モデリング言語(MCMCによるBayes推論が主)
- [Turliuc et al. 2016]*が幾つかのPPによるLDAの実装を比較した.
- 尤度と実行時間を測った(尤度は当てはまりの良さを示す. 実行時間は秒単位)

PRSIMは対数周辺尤度で1位
計算時間で2位



実験データはLDAを使い生成した(語彙は25語, 10トピック, 1000文書, 100語/文書)

```
values(word,[1,2,..,25]).
values(topic,[1,2,..,10]).
doc(D):- (D=[W|R] ->
    msw(topic,Z),
    msw(word(Z),W),
    doc(R) ; D=[]).
```

PRISM プログラム

Fig. 6. Likelihood against log average execution time. Higher is better.

* Probabilistic abductive logic programming using Dirichlet priors, Turliuc et al. IJAR (78) pp.223–240, 2016

既存言語との比較:（続き）

StanとR(topicmodels)

- 学習したモデルの汎化性能をperplexity（$2^{-\frac{1}{N}\sum_i log\ p(x_i)}$, x_i:data）で測る
- Perplexityとトピック数の関係を測った（トピック数10が正解）

PRSIMは明確に正解トピック数を当てている

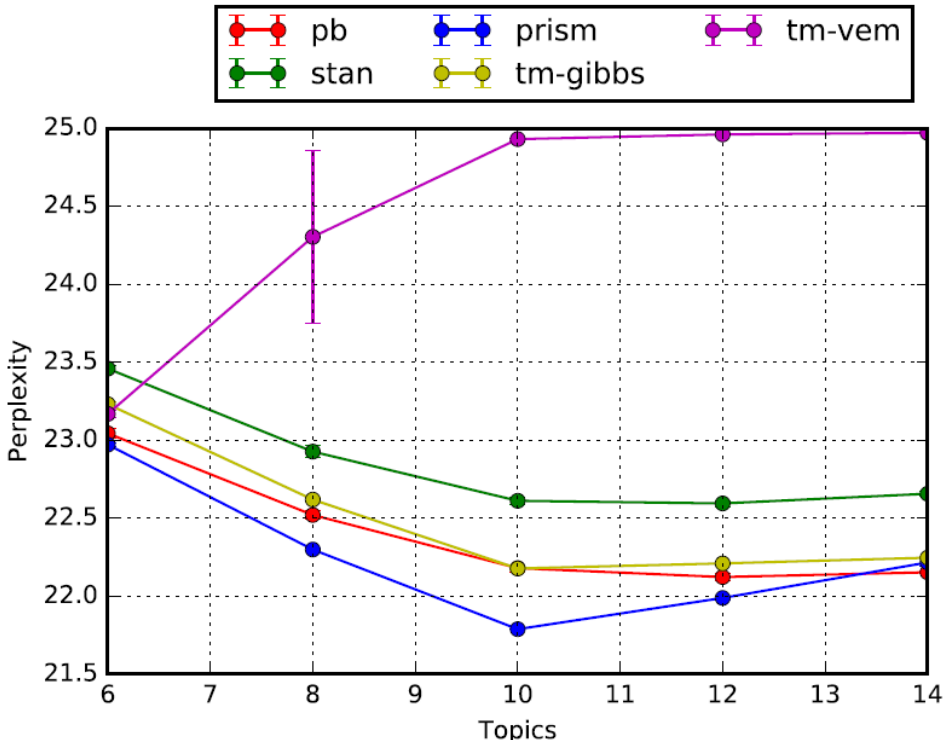
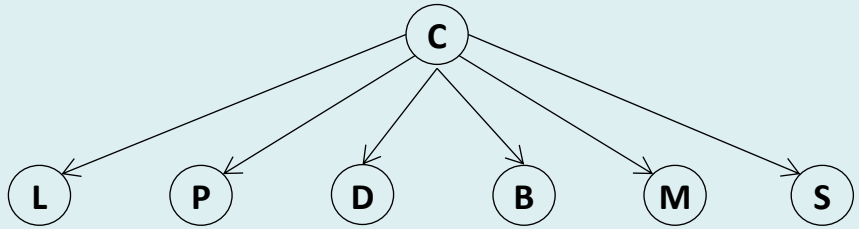


Fig. 9. Average fold perplexity in 5CV against number of topics in the evaluated model. from [Turliuc et al. 2016]

生成的な判別モデル

- Logistic regressionやCRFの一般化
  - HMMに適用するとlinear-chain CRFになる
  - PCFGに適用するとCRF-CFGになる
- 分類に応用
  - 完全データ用と不完全データ用の２種類の生成的PRISMプログラムを書く
  - 生成モデルに対し学習時間が掛かるが判別性能が良い

判別モデル:ナイーブベイズ



Naïve Bayes (NB)

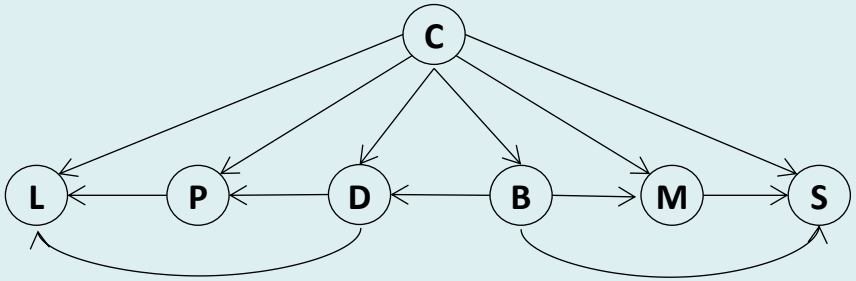
UCI-car data
- size = 1728
- #class = 4
- #attrs = 6
- no missing value

```
values(class,[unacc,acc,good,vgood]).
values(attr(buying,_),[vhigh,high,med,low]).
values(attr(maint,_),[vhigh,high,med,low]).
values(attr(safety,_),[low,med,high]).
values(attr(doors,_),[2,3,4,more5]).
values(attr(persons,_),[2,4,more]).
values(attr(lug_boot,_),[small,med,big]).
```

ナイーブベイズ プログラム

```
nb(Attrs,C):-
    Attrs = [B,M,S,D,P,L],
    msw(class,C),
    msw(attr(buying,     [C]),  B),
    msw(attr(maint,      [C]), M),
    msw(attr(safety,     [C]),  S),
    msw(attr(doors,      [C]),  D),
    msw(attr(persons,    [C]),  P),
    msw(attr(lug_boot,   [C]),  L).
```

判別モデル：BAN



BAN (BN Augmented Naïve Bayes)

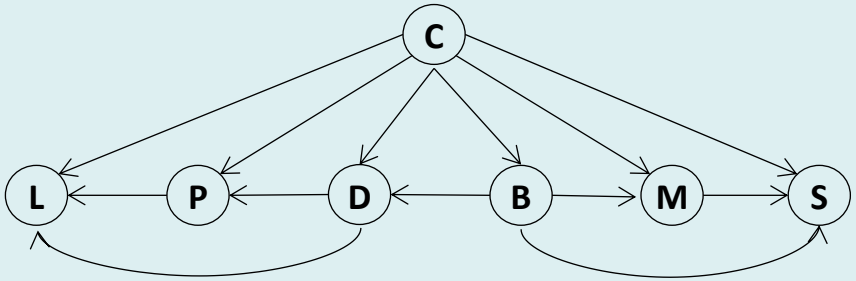
UCI-car data
- size = 1728
- class = 4
- #attrs = 6
- no missing value

```
values(class,[unacc,acc,good,vgood]).
values(attr(buying,_),[vhigh,high,med,low]).
values(attr(maint,_),[vhigh,high,med,low]).
values(attr(safety,_),[low,med,high]).
values(attr(doors,_),[2,3,4,more5]).
values(attr(persons,_),[2,4,more]).
values(attr(lug_boot,_),[small,med,big]).
```

```
bn(Attrs,C):-
    Attrs = [B,M,S,D,P,L],
    msw(class,C),
    msw(attr(buying,    [C]),      B),
    msw(attr(maint,     [C,B]),    M),
    msw(attr(safety,    [C,B,M]),  S),
    msw(attr(doors,     [C,B]),    D),
    msw(attr(persons,   [C,D]),    P),
    msw(attr(lug_boot,  [C,D,P]),  L).
```

BAN プログラム

判別モデル：CRF-BAN



BAN (BN Augmented Naïve Bayes)

UCI-car data
- size = 1728
- class = 4
- #attrs = 6
- no missing value

```
values(class,[unacc,acc,good,vgood]).
values(attr(buying,_),[vhigh,high,med,low]).
values(attr(maint,_),[vhigh,high,med,low]).
values(attr(safety,_),[low,med,high]).
values(attr(doors,_),[2,3,4,more5]).
values(attr(persons,_),[2,4,more]).
values(attr(lug_boot,_),[small,med,big]).
```

```
bn(Attrs,C):-
    Attrs = [B,M,S,D,P,L],
    msw(class,C),
    msw(attr(buying,     [C]),       B),
    msw(attr(maint,      [C,B]),     M),
    msw(attr(safety,     [C,B,M]),   S),
    msw(attr(doors,      [C,B]),     D),
    msw(attr(persons,    [C,D]),     P),
    msw(attr(lug_boot,   [C,D,P]),   L).
bn(Attrs):- bn(Attrs,_).
```

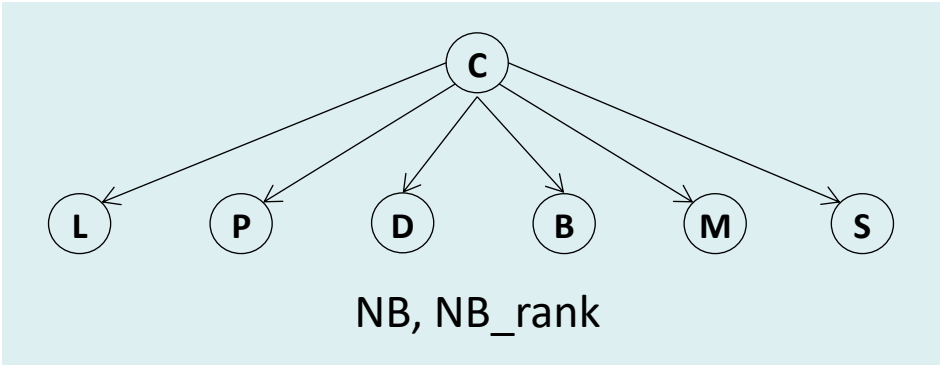
CRF-BAN プログラム

正確度(accuracy)比較

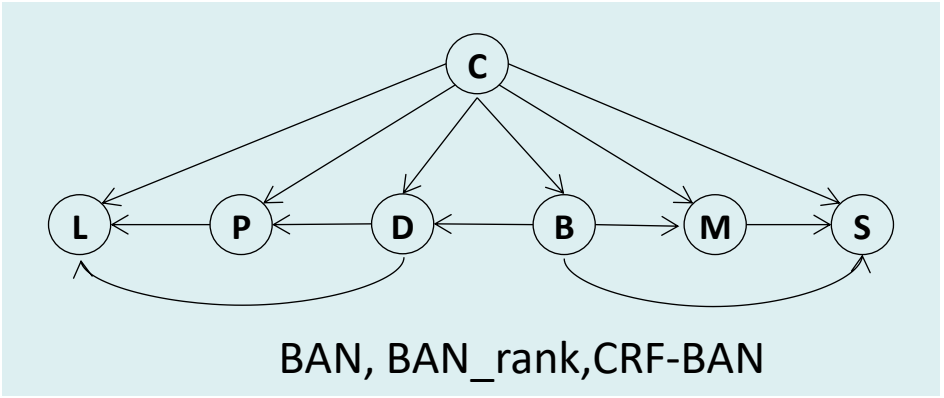
- 問題: データからパラメータ学習して車のクラスを当てる
- 順序ペア: $nb([a_1..,a_6],c_{correct}) \succ nb([a_1..,a_6],c') \ (c' \neq c_{correct})$

モデル	accuracy	学習法
NB	86.1%	MLE (learn/1)
NB_rank	88.2%	Rank learning (rank_learn/1)
BAN	91.5%	MLE (learn/1)
BAN_rank	97.7%	Rank learning (rank_learn/1)
CRF-BAN	99.8%	LBFG (crf_learn/1)

accuracy:10-fold CV



NB, NB_rank

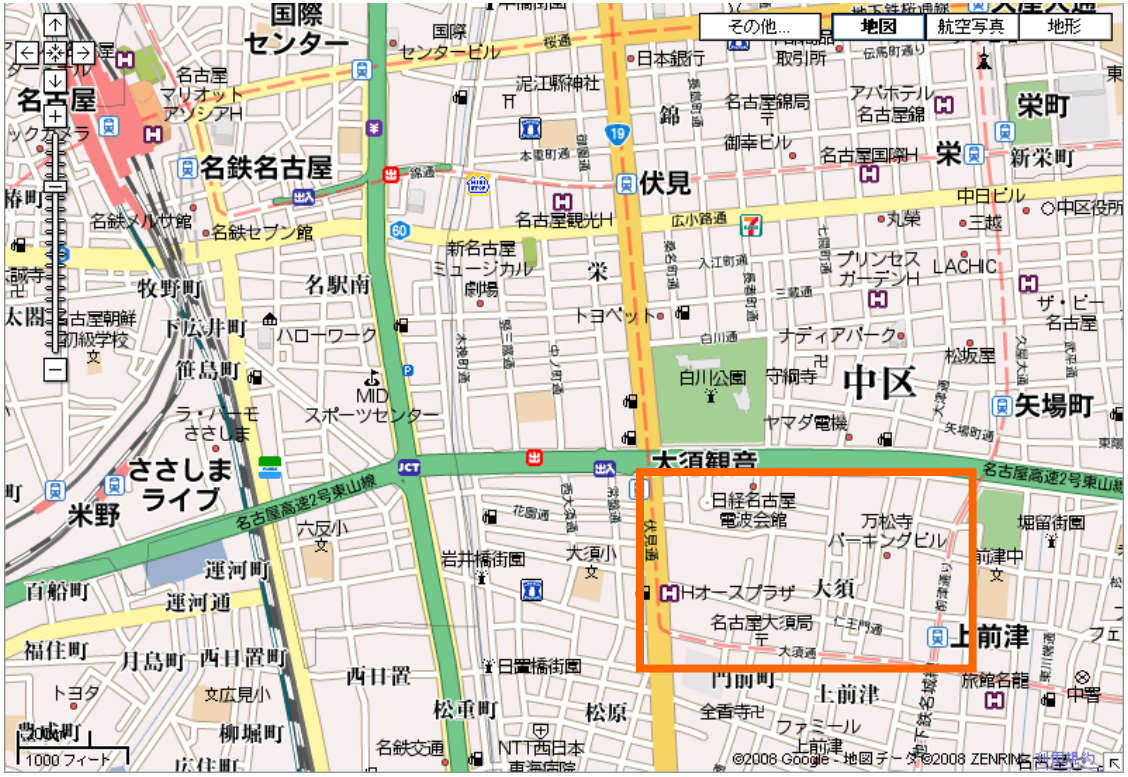


BAN, BAN_rank,CRF-BAN

- rank_learn/1 によるパラメータ学習はlearn/1（最尤推定）を凌ぐ
- CRF-BAN が正確度最高であるが crf_learn/1 による学習時間は長い

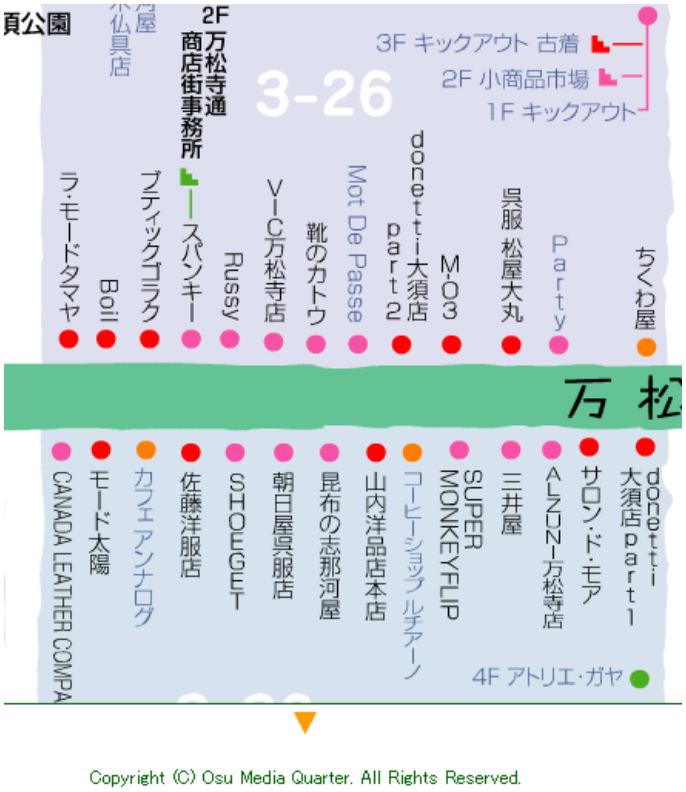
商店街回遊者のモデリング

- 名古屋市大須地区は大須観音を中心に発展した地域であり、神社や寺など多く立地する。
- また、電気街、古着屋・衣料品店、雑貨屋などが各商店街ごとに集まっている。
- そのため多種多様な人が集まる店やスポットがある。
- 商店街回遊者の行動系列を混合マルコフ連鎖によりモデル化しクラスタリングした
- 名古屋工業大学兼田研究室との共同研究



■Google マップより引用。

商店街のモデリング



Copyright (C) Osu Media Quarter. All Rights Reserved.

食べる ●	着る ●	買う ●	家電 ●	その他 ●
3	11	13	0	1

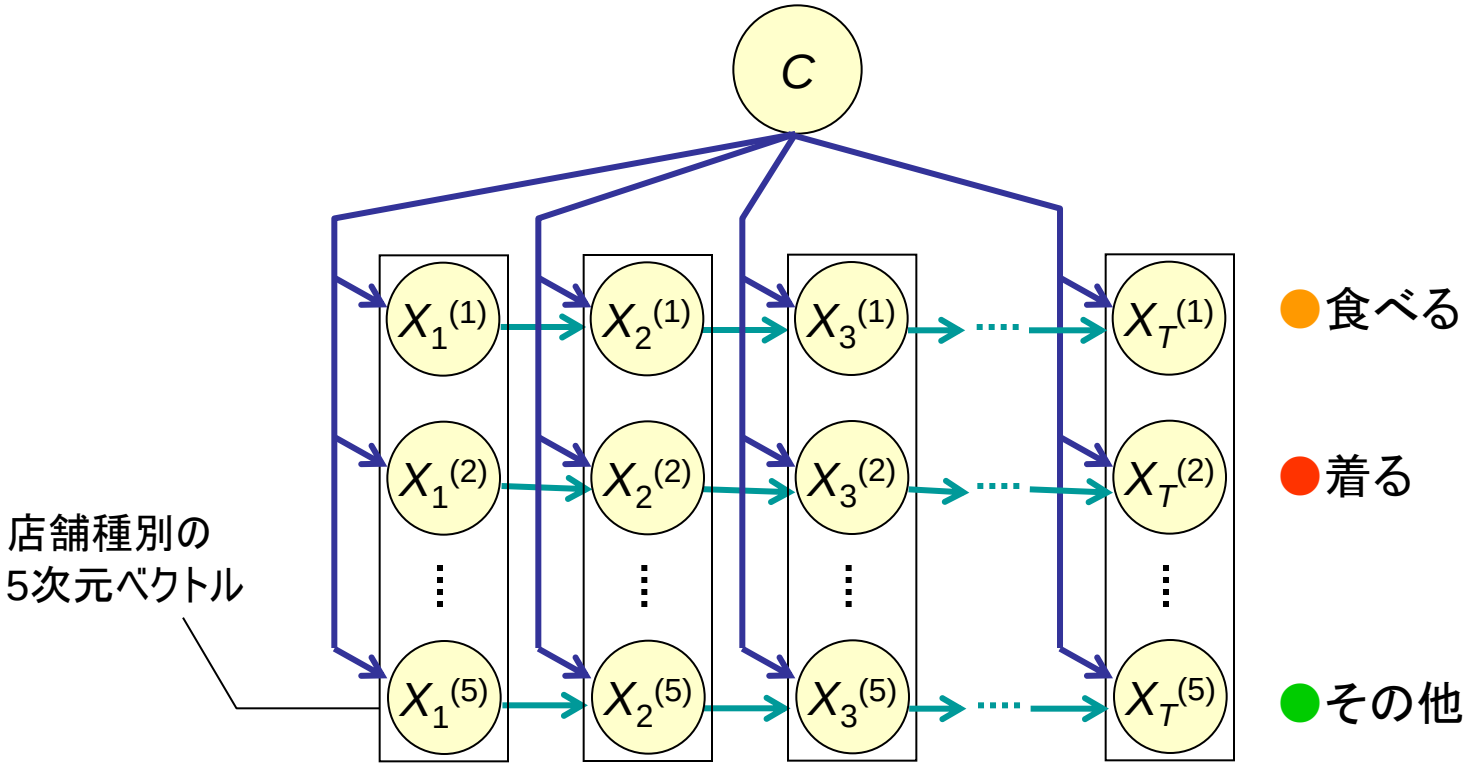
分類別の店の有無を2値で表現

食べる ●	着る ●	買う ●	家電 ●	その他 ●
1	1	1	0	1

■大須マップWEB版(http://www.osumap.jp/2005/B_2.htm)より引用

43

回遊モデル：混合マルコフ連鎖

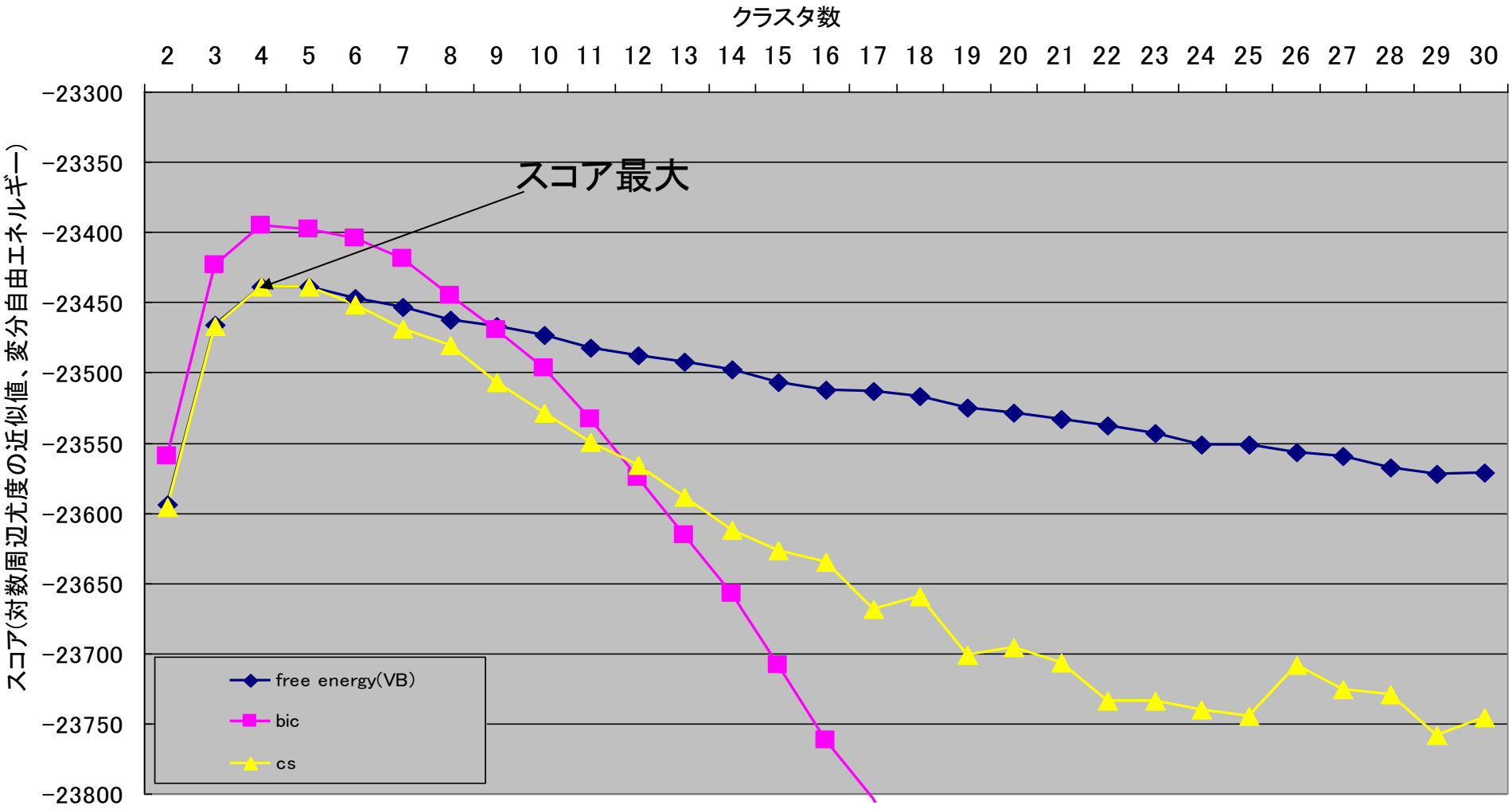


C：クラスタ

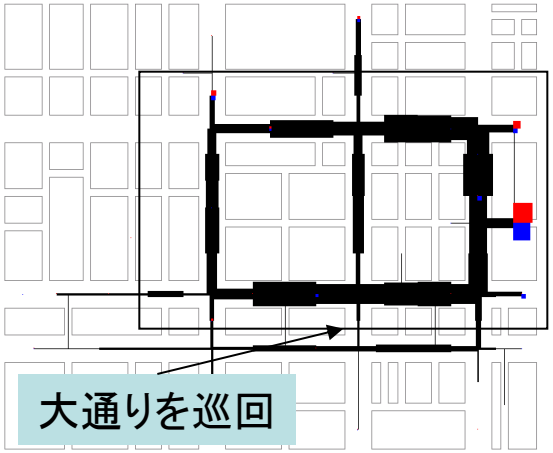
$X_i^{(j)}$：i番目の通行路に対してのj番目の店舗属性の値

$(1 \le i \le T, 1 \le j \le 5)$

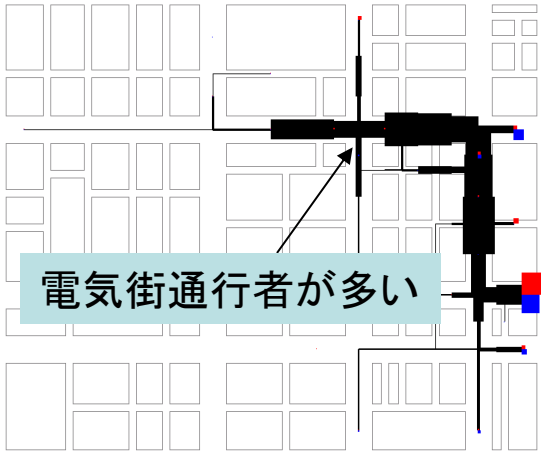
回遊者のクラスタ数とモデルスコア



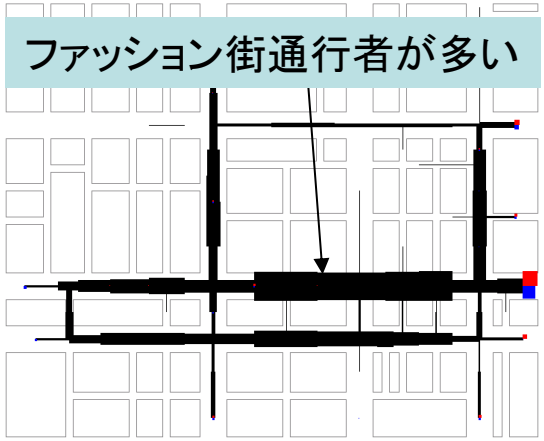
変分ベイズ法によるクラスタリング結果



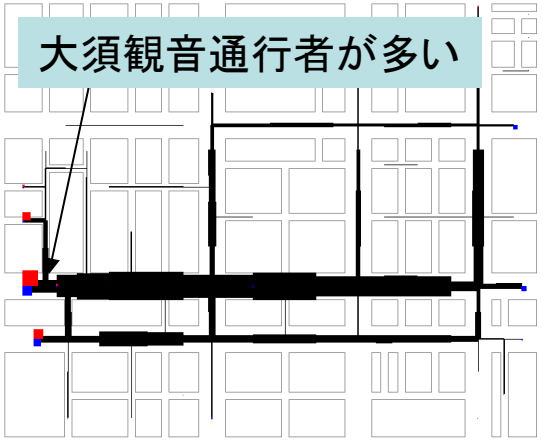
クラスタA(52人)



クラスタB(55人)

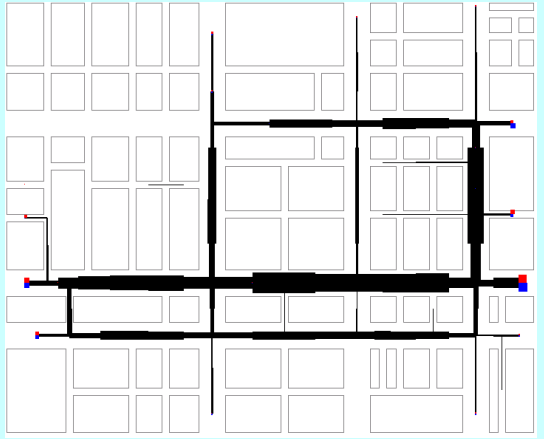


クラスタC(120人)



クラスタD(140人)

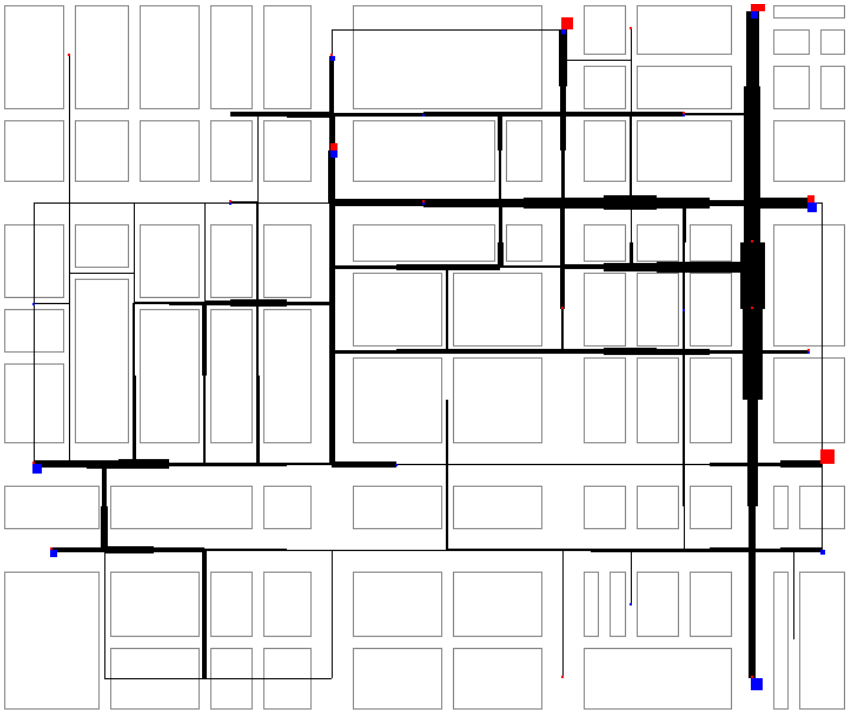
クラスタE(37人)



データ全体(404人)

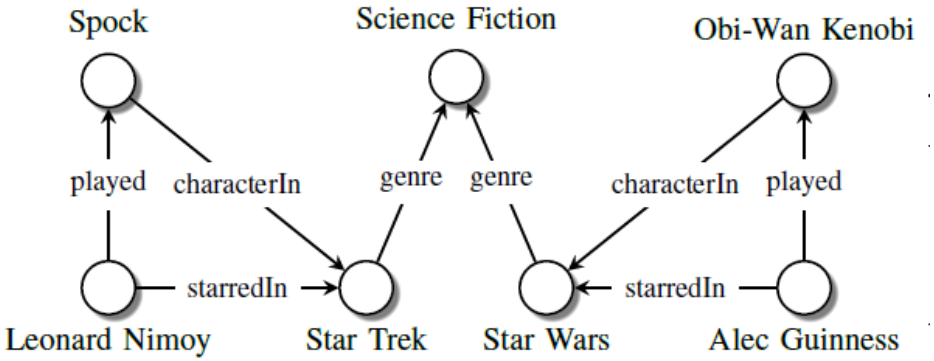
クラスタE(37人)

- 裏路地型
- 他のクラスタでは見られない大通り以外を通行するクラスタ
- 歩行距離が短く, 訪問施設数も少ない
- データ提供元の兼田研究室の分析では見つからなかった歩行者タイプ



知識グラフとPRISM

- 大規模な知識グラフ(knowledge graph, KG) が利用可能 (FreeBase,Wikidata,Knowledge Vault,..)
- 3つ組の集合(*s*,*rel*,*o*) (= 命題 *rel*(*s*,*o*))
- Q.A. on (*s*,*rel*,*o*) ：Who played Spock?



From: A Review of Relational Machine Learning for Knowledge Graphs
Maximilian Nickel, Kevin Murphy, Volker Tresp, Evgeniy Gabrilovich,
Proceedings of the IEEE 104(1): 11-33 (2016)

埋め込み法

- データを低次元空間に埋め込む
  - 実体: *s*,*o* ➔ K-次元 ベクトル **s**,**o** (K ≪ |entities|)
  - 関係: *rel* ➔ K-次元 ベクトル **rel**, K x K matrix **D**
- Translation モデル
  - TransE [Bordes+,13],..,HolE [Nickel+,16]
  - *rel*(*s*,*o*) ≈ true ⇔ ‖**s** + **rel** - **o**‖ ≈ 0
- 双線形モデル
  - RESCAL [Nickel,13],..,DistMult [Yang+,15], [Trouillon+,16]
  - *rel*(*s*,*o*) ≈ true ⇔ $\mathbf{s}^{\mathsf{T}}\mathbf{D}\mathbf{o}$ = **(s•Do)** ≈ 1 (**D** is diagonal)
- PRISM-DistMult モデル (実体ベクトル= 確率分布)

```
values(cluster(_),[1-K]):- K=50.
values(rel(_),[true,false]).

rel(S,O,V):-  msw(cluster(S),C),msw(cluster(O),C),msw(rel(C),V).
```

論理からベクトル空間へ

rel(s,o,true) ⇔
　　∃c msw(cluster(s),c) & msw(cluster(o),c) & msw(rel(c),true))

s = (P(msw(cluster(s),1)),..,P(msw(cluster(s),K)))^T
o = (P(msw(cluster(o),1)),..,P(msw(cluster(o),K)))^T
D[i,j] = P(msw(rel(i),true))) if i=j, =0 o.w.

P(rel(s,o,true))
= P(∃c msw(cluster(s),c) & msw(cluster(o),c) & msw(rel(c),true)))
= $\sum_{\{c \text{ in } [1..K]\}}$ P(msw(cluster(s),c) & msw(cluster(o),c) & msw(rel(c),true))
= $\sum_{\{c \text{ in } [1..K]\}}$ P(msw(cluster(s),c))P(msw(cluster(o),c)) P(msw(rel(c),true))
= **s**[1]**D**[1,1]**o**[1]+…+ **s**[K]**D**[K,K]**o**[K]
= **s^TDo = (s•Do)**

確率モデルのパラメータ学習

- パラメータ学習は最尤推定でも順位学習でも可能
- 最尤推定には正例があれば良い
- 順位学習では正例(s,rel,o) と負例(s',rel,o')から成る順位ペア ((s,rel,o) ≻ (s',rel,o'))についてP((s,rel,o)|θ) > P(rel(s', rel,o')|θ) となるようにパラメータθを学習➔負例が必要
- 知識グラフは正例のみ含む
- 負例を [Socher+,13] に従い生成する
  - Given (s,rel,o) in KG, randomly sample entities s', o' such that (s',rel,o), (s,rel,o') not in KG (🡐 実際はもう少し複雑)
  - ペア((s, rel,o) ≻ rel(s', rel,o)), (rel(s, rel,o) ≻ rel(s, rel,o')) as を自順位学習の訓練データおよびテストデータとして使用する

3つ組判別問題:FB15k data

- FB15k[Bordes+,13]: FBの一部，実体数14,951 関係数1,345
- Top-10 largest relations $\{rel_0,..,rel_9\}$ を実験に使った

Data set	rel_0	rel_1	rel_2	rel_3	rel_4	rel_5	rel_6	rel_7	rel_8	rel_9
training	15998	12893	12166	12175	11547	11636	9466	9494	9439	9465
valid	1706	1353	1304	1191	1195	1200	1049	976	965	969
test	2060	1591	1451	1555	1478	1384	1123	1168	1190	1160

- 問題: $\{rel_0,..,rel_9\}$ の各関係についてPRISM-DistMult modelのパラメータを学習し，正例・負例を判別し，5回繰り返して正確度を測る.

	rel_0	rel_1	rel_2	rel_3	rel_4	rel_5	rel_6	rel_7	rel_8	rel_9
Rank learning	87.4%	79.8%	74.1%	72.7%	62.0%	62.7%	54.8%	55.0%	69.4%	69.5%
MLE	85.9%	60.4%	68.7%	72.6%	61.8%	56.1%	52.0%	50.0%	66.5%	69.7%
T-Test	=	>	=	=	=	>	=	>	=	=

- 順位学習は最尤推定より悪くないパラメータを推定する

終わりに

- 人工知能の発展には高級言語による界面言語が必要である
  - 確率プログラミング言語を使い，モデリングと学習・計算を分離することにより確率モデルの開発・維持の手間を減らす
- PRISM2.3は論理ベースの確率プログラミング言語として多種多様な学習機能を提供している
  - モデリング例を幾つか紹介した
  - http://rjida.meijo-u.ac.jp/prism/ からダウンロード可能