An introduction to PRISM and its applications

Yoshitaka Kameya
Tokyo Institute of Technology

Contents

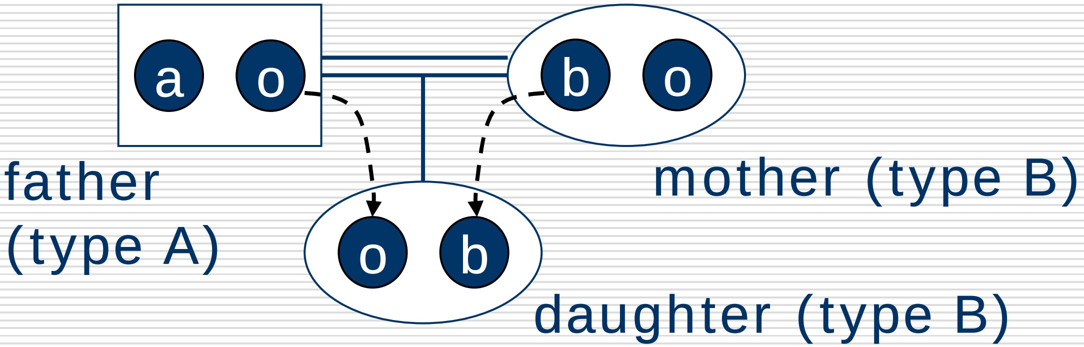
- What is PRISM?
  - Two examples:
    - from population genetics
    - from statistical natural language processing
- Details of PRISM system
- Applications

What is PRISM?

- PRISM = PRogramming In Statistical Modeling
- PRISM is a programming language/system for probabilistic modeling
  - Probabilistic extension of Prolog
  - Tool for probabilistic modeling with complex data beyond traditional data matrices
    - sequences, trees, graphs and relations

What is PRISM? - example (1)

- Inheritance of ABO blood type



- Hardy-Weinberg's law

$$
\begin{aligned}
P_A &= \theta_a^2 + 2\theta_a\theta_o \\
P_B &= \theta_b^2 + 2\theta_b\theta_o \\
P_O &= \theta_o^2 \\
P_{AB} &= 2\theta_a\theta_b
\end{aligned}
$$

P_A, P_B, P_O, P_{AB}:
Frequencies of blood types

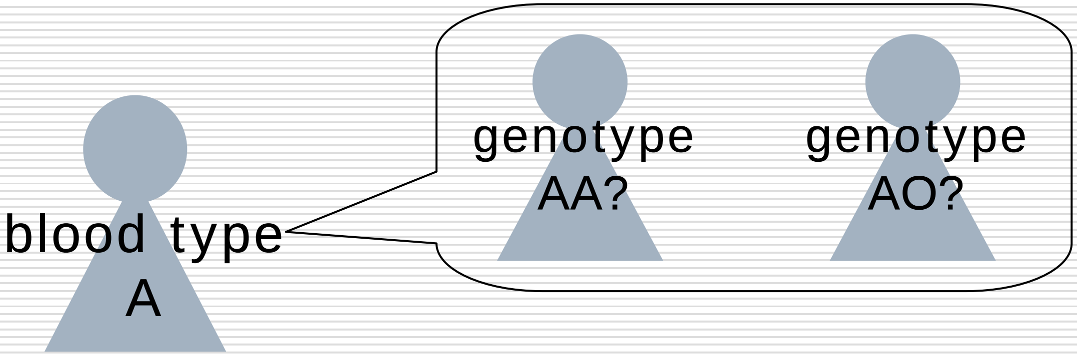
$\theta_a, \theta_b, \theta_o$:
Frequencies of genes

What is PRISM? - example (1)

- Problem:

  Estimate the frequencies $\{\theta_a, \theta_b, \theta_o\}$ of genes from the frequencies $\{P_A, P_B, P_O, P_{AB}\}$ of blood types

  - Sometimes gene frequencies are useful to characterize the population of interest
  - Note: we can only observe the blood types → *there is an ambiguity*



What is PRISM? - example (1)

- Solving this estimation problem by PRISM
  - PRISM program:

We use a *random switch* named 'gene'
whose parameters correspond to gene frequencies

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

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

Pick up two genes X and Y
from a pool of genes

```
bloodtype(P) :-
    genotype(X,Y),
    ( X=Y -> P=X
    ; X=o -> P=Y
    ; Y=o -> P=X
    ; P=ab
    ).
```

Mapping from
genotype to
phenotype
using OR (;)
and if-then (->)

  - Run the EM (expectation-maximization) algorithm provided by PRISM system:

```
?- learn.
```

What is PRISM? - example (1)

Demo

What is PRISM? - example (2)

- Probabilistic parsing using probabilistic context-free grammars (PCFGs)
  - Example:

s → np vp	(0.8)	verb → *swat*	(0.2)
s → vp	(0.2)	verb → *flies*	(0.4)
np → noun	(0.4)	verb → *like*	(0.4)
np → noun pp	(0.4)	noun → *swat*	(0.05)
np → noun np	(0.2)	noun → *flies*	(0.45)
vp → verb	(0.3)	noun → *ants*	(0.5)
vp → verb np	(0.3)	prep → *like*	(1.0)
vp → verb pp	(0.2)		
vp → verb np pp	(0.2)		
pp → prep np	(1.0)		

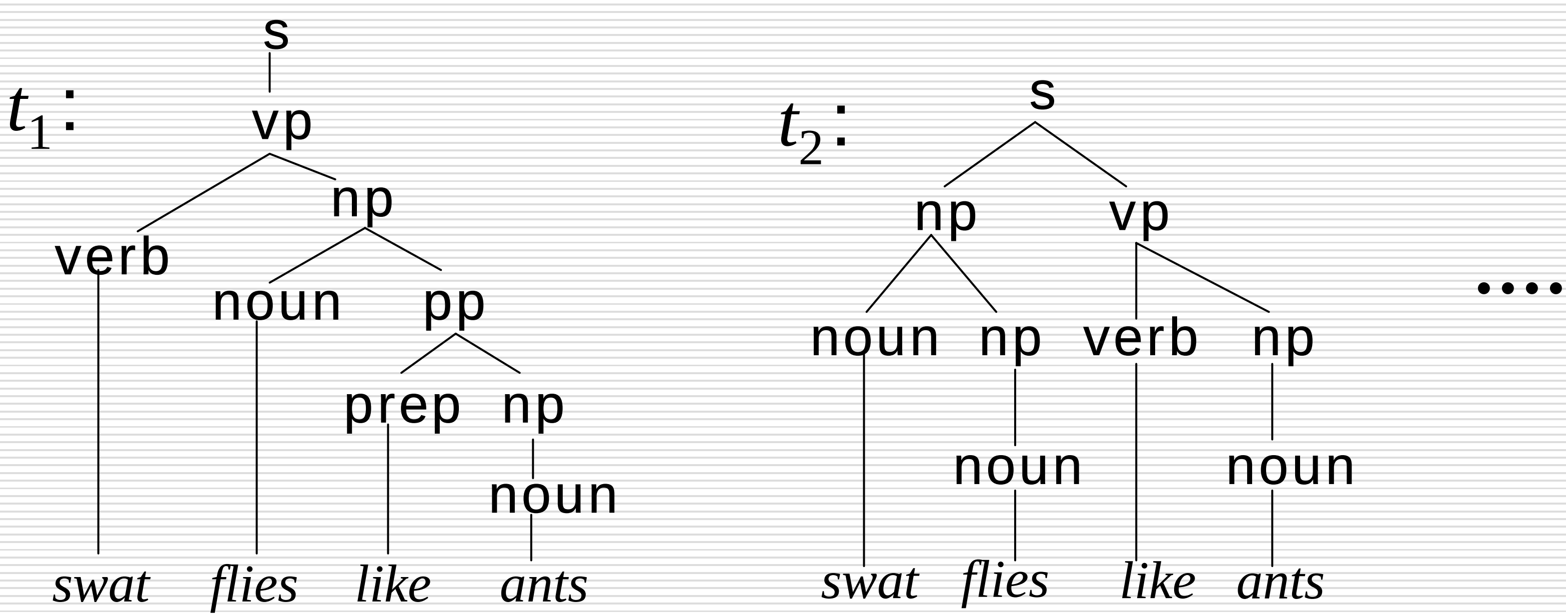
from [Charniak 93]

  - A sentence is derived by applying probabilistic rules:

s → vp → verb np → *swat* np → *swat* noun pp
→ *swat files* pp → *swat flies* prep np → *swat flies like* np
→ *swat flies like* noun → *swat flies like ants*

What is PRISM? - example (2)

- Probabilistic parsing:
  - Compute the <u>most probable</u> parse for a given sentence (e.g. *"swat flies like ants"*)



$P(t_1) = 0.2*0.3*0.2*0.4*0.45*1.0*1.0*0.4*0.5 = 0.000432$

$P(t_2) = 0.8*0.2*0.05*0.4*0.45*0.3*0.4*0.4*0.5 = 0.00003456$

What is PRISM? - example (2)

- PRISM program for PCFGs:

```
values(s,[[np,vp],[vp]]).
values(np,[[noun],
           [noun,
           [noun,
values(vp,[[verb],
           [verb,np],
           [verb,pp],
           [verb,np,pp]]).
values(pp,[[prep,np]]).
values(verb,[[swat],[flies],[like]]).
values(noun,[[swat],[flies],[ants]]).
values(prep,[[like]]).
```

```
pcfg
          -L1):-
          nal(LHS) ->
          S,RHS),proj(RHS,L0-L1)
  ; L0 =
  ).

proj([],L-L).
proj([X|Xs],L0-L1):-
  pcfg(X,L0-L2),proj(Xs,L2-L1).
```

LHS is replaced with RHS by a rule 'LHS → RHS'

To generate a sentence L, we start with the symbol 's'

Symbols in RHS are processed one by one

We use random switches msw(noun,[swat]), msw(noun,[flies]), msw(noun,[ants]) to make a choice among the rules noun→swat, noun→flies, noun→ants.

What is PRISM? - example (2)

- Probabilistic parsing by the `viterbif` routine:

```
| ?- viterbif(pcfg([swat,flies,like,ants])).

pcfg([swat,flies,like,ants])
   <= pcfg(s,[swat,flies,like,ants]-[])
pcfg(s,[swat,flies,like,ants]-[])
   <= pcfg(vp,[swat,flies,like,ants]-[]) & msw(s,[vp])
      :
pcfg(noun,[ants]-[])
   <= pcfg(ants,[ants]-[]) & msw(noun,[ants])
pcfg(ants,[ants]-[])

Viterbi_P = 0.000432
```

The most probable parse is $[[\mathrm{swat}_{\mathrm{verb}}[\mathrm{flies}_{\mathrm{noun}}[\mathrm{like}_{\mathrm{prep}}\ [\mathrm{ants}_{\mathrm{noun}}]_{\mathrm{np}}]_{\mathrm{pp}}]_{\mathrm{np}}]_{\mathrm{vp}}]_{\mathrm{s}}$ and its generative probability is 0.000432

What is PRISM? - example (2)

Demo

What is PRISM?

- Merit of PRISM programming
  - High expressivity from first-order representations
    - We can write our model in a compact and readable form
  - Declarative semantics
    (distribution semantics [Sato 95])
  - A lot of built-ins for ease of probabilistic modeling
    - We need not to derive/implement model-specific probabilistic inference algorithms
    - All we need to do is write our model

Basic probabilistic inferences

- PRISM system provides built-in predicates for:

  Sampling

  For a given goal G, return answer substitution σ with the probability $P_\theta(G\sigma)$

  Probability computation

  For a given goal G, compute $P_\theta(G)$

  Viterbi computation

  For a given goal G, find the most probable explanation $E^* = \mathrm{argmax}_{E\in\psi(G)} P_\theta(E)$ where $\psi(G)=\{E_1, E_2, ..., E_K\}$ are possible explanations for G

  Hindsight computation

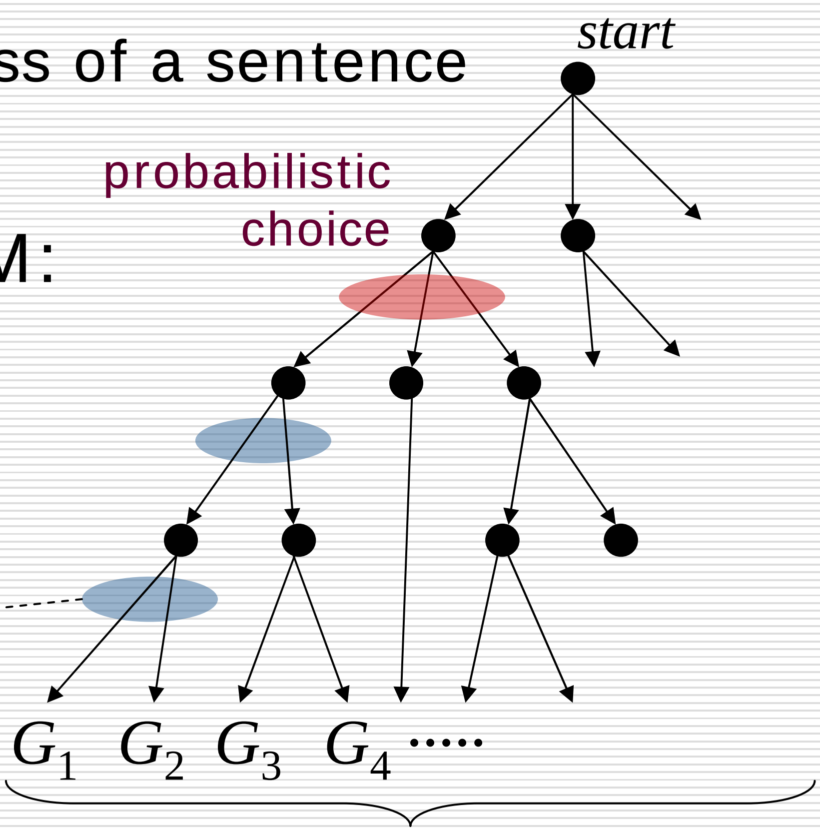
  For a given goal G, compute $P_\theta(G')$ or $P_\theta(G' \mid G)$ where G' is a subgoal of G

  EM learning

  Given a bag $\{G_1, G_2, ..., G_T\}$ of goals, estimate the parameters θ that maximizes the likelihood $\Pi_t P_\theta(G_t)$

Generative modeling

- Basically we need to model the generation process of the data at hand
  - Just like a derivation process of a sentence using PCFGs
- Key assumptions in PRISM:
  - Exclusiveness condition
  - Uniqueness condition
  - ...



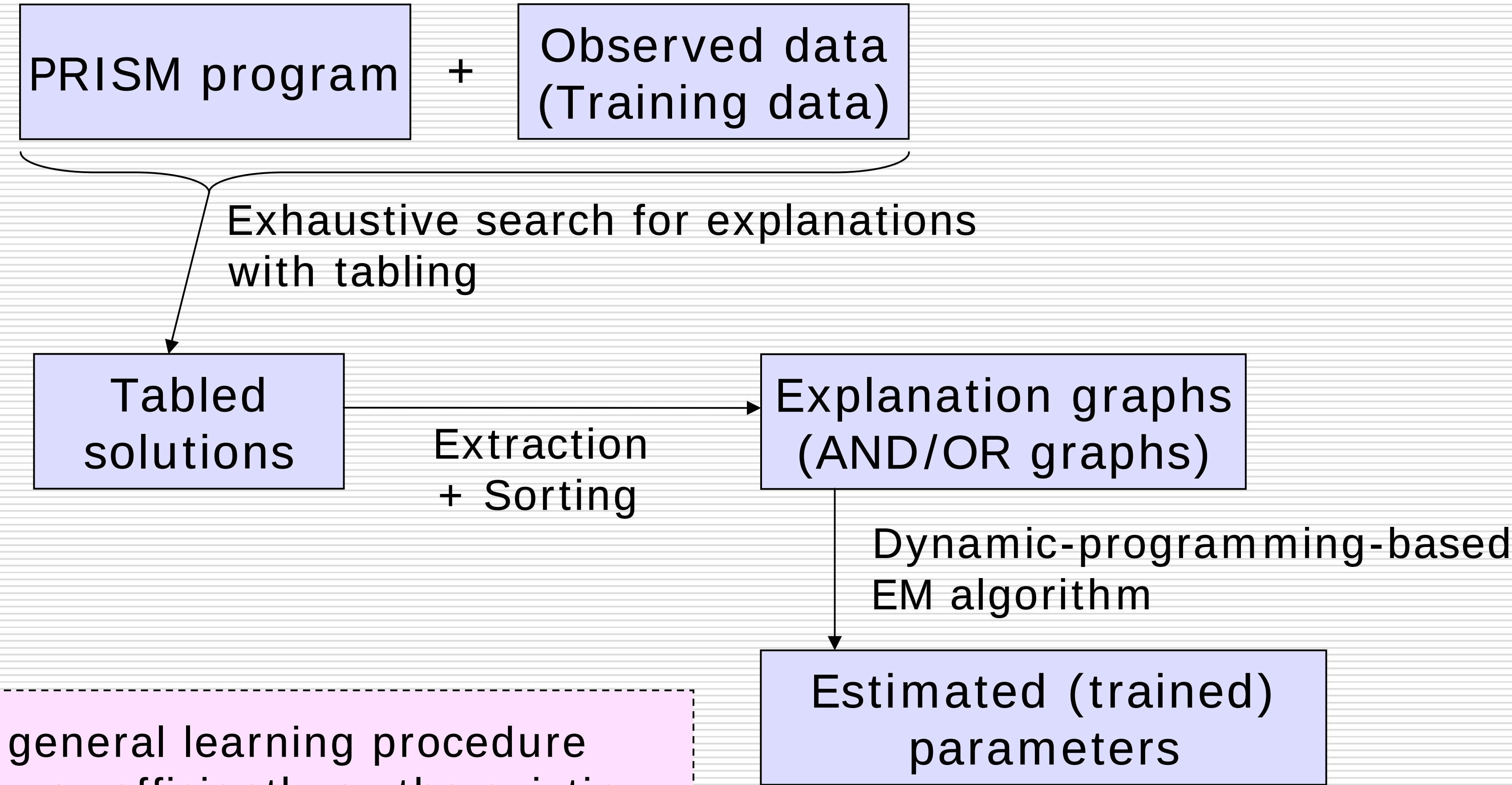
Exclusiveness condition:
Generation paths are exclusive to each other

Uniqueness condition:

G's are exclusive and $\Sigma_G P(G) = 1$

Efficient probabilistic inferences

- Data flows in EM learning of parameters



This general learning procedure works as efficiently as the existing model-specific EM algorithms

Advanced probabilistic inferences

- Handling failures in the generation process (version 1.8)
- Model selection (version 1.10)
- Variational Bayesian learning (version 1.11)
- Data-parallel EM learning (version 1.11)
- Deterministic annealing EM algorithm (version 1.11)

Basically there is no need to modify the model part of the program

Applications (1)

- Another hypothesis on blood type inheritance (from J. F. Crow (1983), *Genetics Notes*)
  - Theory I: alleles at two loci -- A/a and B/b

Phenotype	Genotype in theory I	Genotype in theory II
A	A- bb	AA, AO
B	aa B-	BB, BO
O	aa bb	OO
AB	A- B-	AB

(known as correct now)

- Goal: compare two theories by BIC (Baysian Information Criterion), a well-known Bayesian model score

Applications (1)

- Another theory on blood type inheritance
  - PRISM program for theory I:

```
values(locus1,['A',a]).
values(locus2,['B',b]).

genotype(L,X,Y) :-
   msw(L,X),
   msw(L,Y).
```

```
bloodtype(P) :-
   genotype(locus1,X1,Y1),   % 1st locus
   genotype(locus2,X2,Y2),   % 2nd locus
   ( X1=a, Y1=a, X2=b, Y2=b -> P=o
   ; ( X1='A' ; Y1='A' ), X2=b, Y2=b -> P=a
   ; X1=a, Y1=a, ( X2='B' ; Y2='B')  -> P=b
   ; P=ab
   ).
```

  - 100 samples from typical Japanese:

Bloodtype	A	B	O	AB
#persons	38	22	31	9

Applications (1)

- Another theory on blood type inheritance
  - Run the programs for theories I and II:

```
?- prism(bloodAaBb).
   :
?- learn.
?- prism_statistics(bic,BIC).
   :
BIC = -135.64984669945866 ?
```

The value of BIC can be obtained after learning

```
?- prism(bloodABO)
   :
?- learn.
?- prism_statistics(bic,BIC).
   :
BIC = -132.66708143397111 ?
```

➔ Theory II (ABO) has higher BIC score than that of Theory I (AaBb)

➔ Theory II is more reasonable *according to the data* (χ^2-test is used in the previous approach)

Conclusion

- PRISM is a general programming language/system for probabilistic modeling
  - PRISM language allows us to handle complex data such as sequences, trees, graphs, relations, ...
  - PRISM system provides a lot of built-ins for ease of probabilistic modeling
- We believe PRISM is fairly useful but currently its main target is generative models
  - Eliminating the modeling assumptions is future work

Further materials...

- Please visit:
  http://sato-www.cs.titech.ac.jp/prism/
- Latest version: 1.11 beta 7
- You can download:
  - Software
  - Papers