Seminar aus maschinellem Lernen

Thema: BLOG: Probabilistic Models with Unkown objects

Vortraggeber: Di Wu

Datum: 02.12.2009

Outline

1. Introduction
2. Examples
3. Syntax and Semantics: Possible worlds
4. Syntax and Semantics: Probabilities
5. Evidence and Queries
6. Inference

1. Introduction

- The problem of unknown objects:
  - Many AI problems involve making inferences about real-world objects that underlie some data. In many cases, we do not know the number of underlying objects or the mapping between obsevations and objects.
  - The agent must infer the existence of objects that were not know initially to exist.
  - *Population estimation* , for example, involves counting a population by sampling from it randomly and measuring how often the same object is resampled.
- A probabilistic modeling language, called Bayesian logic(BLOG), allows such scenarios to be represented in a natural way.
- A well-formed BLOG model fully defines a distribution over model structures of a first-order logical language.

2. Examples

- Two typical scenarios with unknow objects – simplified versions of the population estimation and multitarget tracking problems.
- In each case, we provide a short BLOG model.
- Example 1
  - An urn contains an unknown number of balls. Balls are equally likely to be blue or green. We draw some balls from the urn, observing the color of each and replacing it. We cannot tell two identically colored balls apart; furthermore, observed colors are wrong with probability 0.2. How many balls are in the urn? was the same ball drawn twice?

BLOG model for example 1

```
1.  type Color ;  type Ball ; type Draw ;

2.  random Color TrueColor(Ball) ;
3.  random Ball BallDrawn(Draw) ;
4.  random Color ObsColor(Draw) ;

5.  guaranteed Color Blue , Green ;
6.  guaranteed Draw Draw1 , Draw2 , Draw3 , Draw4 ;

7.  #Ball ~ Poisson[6]( ) ;

8.  TrueColor(b) ~ TabularCPD[ [0.5 , 0.5] ]( ) ;
9.  BallDrawn(d) ~ Uniform( {Ball b}) ;
10. ObsColor(d)
11.       if ( BallDrawn(d) != null ) then
12.            ~ TabularCPD[ [0.8 , 0.2] , [0.2 , 0.8] ](TrueColor(BallDrawn(d))) ;
```

(2)

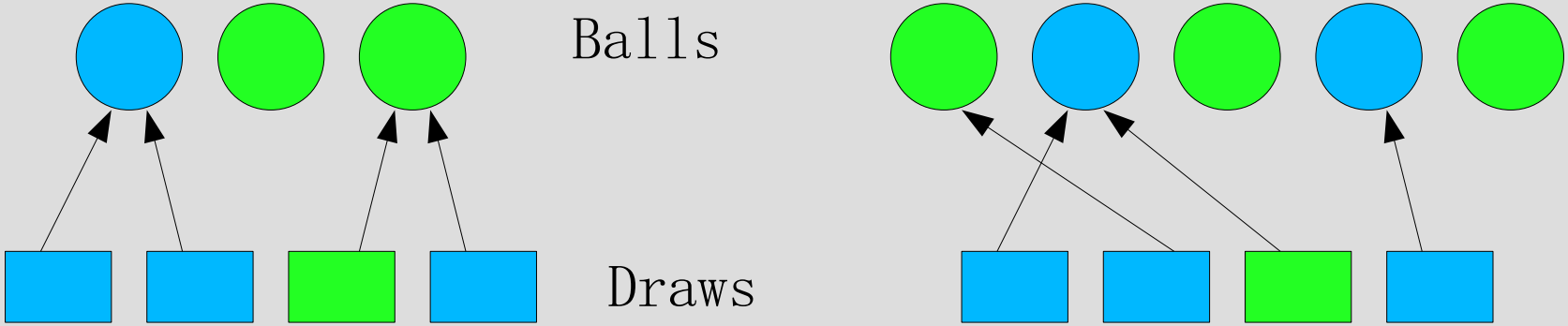
- Example 2
  - An unknown number of aircraft exist in some volume of airspace. An aircraft's state (position and velocity) at each time step depends on its state at the previous time step. We observe the area with radar : aircraft may appear as identical blips on a radar screen. Each blip gives the approximate positions of the aircraft that generated it. However, some blips may be false detections, and some aircraft may not be detected. What aircraft exist and what are their trajectories ?

BLOG model for example 2

```
1.  Type Aircraft ; type Blip ;

2.  random R6Vector State(Aircraft , NaturalNum) ;
3.  random R3Vector ApparentPos(Blip) ;

4.  nonrandom NaturalNum Pred(NaturalNum) = Predecessor ;

5.  origin Aircraft Source(Blip) ;
6.  origin NaturalNum Time(Blip) ;

7.  #Aircraft ~ NumAircraftPrior( ) ;
8   State(a,t)
9.        if t = 0 then ~ InitState( ) ;
10.       else ~ StateTransition(State(a , Pred(t))) ;
11. #Blip(Source = a, Time = t) ~ DetectionCPD(State(a , t)) ;
12. #Blip(Time = t) ~ NumFalseAlarmsPrior( ) ;
13. ApparentPos(b)
14.           if (Source(b) = null ) then ~ FalseAlarmDistrib( )
15.           else ~ ObsCPD(State(Source(b) , Time(b))) ;
```

3.Syntax and Semantics: Possible worlds

- A model structure ω of a typed, free, first order language consists of an extension $[\tau]^{\omega}$ for each type τ, which may be an arbitrary set, and an interpretation $[f]^{\omega}$ for each function symbol *f*. If *f* has return type τ_0 and argument types $\tau_1 \ldots \tau_k$, then $[f]^{\omega}$ is a function from $[\tau_1]^{\omega} \times \ldots [\tau_k]^{\omega}$ to $[\tau_0]^{\omega} \cup \{\text{null}\}$.
- Two model structures are shown as follows, the purpose of a BLOG model is to difine a distribution over such structures.



Outcomes with fixed object sets

- BLOG models for fixed objects sets have five kinds of statments.
  - Type declaration
  - Random function declaration
  - Nonrandom function declaration
  - Guaranteed object statement ($G_M(\tau)$)
  - Dependency statement
- The first four kinds of statements listed above define a particular typed first-order language L_M for a model M.
- The set of *possible worlds* of M, denoted Ω_M, consist of those model structures of L_M, where the extension fo each type τ is $G_M(\tau)$, and all nonrandom function symbols have their given interpretations.

(2)

- For each random function f and tuple of appropriately typed guaranteed objects $o_1, \ldots, o_k$, we can define a random variable (RV) $f[o_1, \ldots, o_k](\omega) := [f]^{\omega}(o_1, \ldots, o_k)$
- In a simplified version of example 1 where the urn contains a known set of balls {Ball1, . . . , Ball8} and we make four draws, the RVs are TrueColor[Draw1], . . . , TrueColor[Draw8],BallDrawn [Draw1], . . . , BallDrawn[Draw4], and ObsColor[Draw1], . . . , ObsColor[Draw4].
- The possible worlds are in one-to-one correspondence with full instantiations of these basic RVs. Thus, a joint distribution for the RVs defines a distribution over possible worlds.

Unknown objects

- A BLOG model defines a generative process in which objects are added iteratively to a world. To describe such processes, we first introduce *origin function declarations* .
- An origin function must take a single argument of some type τ; it is then called a τ-origin function.
- Generative steps that add objects to the world are described by *number statements* .
  - For example : #Blip(Source = a, Time = t) ∽ DetectionCPD(State(a , t)) ;
- The beginning of a number statement has the form:
  - $\#\tau(g_1 = x_1 , \ldots , g_k = x_k)$

(2)

- Consider a number statement for type τ with origin functions $g_1, \ldots, g_k$. An object $q \in [\tau]^\omega$ satisfies this number statement applied to $o_1, \ldots, o_k$ in ω if $[g_i]^\omega(q) = o_i$ for i = 1, . . . ,k, and $[g]^\omega(q)$ = null.

- The generated objects are defined as follows: when a number statement with type τ and origin functions $g_1, \ldots, g_k$ is applied to generating objects $o_1, \ldots, o_k$, the generated objects are tuples $\{ (\tau, (g_1, o_1), \ldots, (g_k, o_k), n) : n = 1, \ldots, N \}$, where N is the number of objects generated.
  - e.g. (Blip, (Source, (Aircraft, 2)), (Time,8), 1)

(3)

- The *universe* of a type τ in a BLOG model M, denoted $U_M(\tau)$, consists of the guaranteed objects of type τ and nested tuples of type τ that can be generated from guaranteed objects through finitely many recursive applications of number statement.

- For a BLOG model M, the set of possible worlds Ω_M is the set of model structures ω of L_M such that:
  - For each type τ, $G_M(\tau) \subseteq [\tau]^\omega \subseteq U_M(\tau)$;
  - Nonrandom functions have the specified interpretations;
  - For each number statement in M with type τ, the set of objects in $[\tau]^\omega$ that satisfy this number statement is $\{ (\tau, (g_1, o_1), \ldots, (g_k, o_k), n) : n = 1, \ldots, N \}$ for some number N.

(4)

- For every type τ , each element of $[\tau]^{\omega}$ satisfies some number statement.

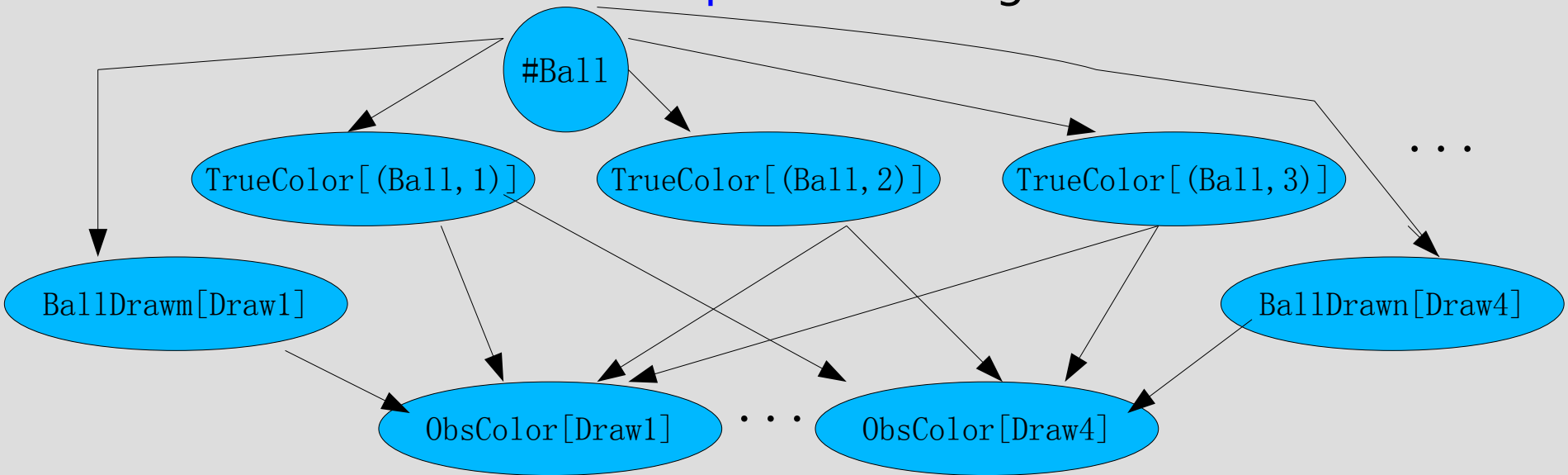
- To archieve the same effect with unknown objects, we need two kinds of basic RVs, for a BLOG model M , the set V_M of basic random variables consists of:
  - For each random function f, a function application RV $f[o_1, \ldots, o_k](\omega)$ is equal to $[f]^{\omega}(o_1, \ldots, o_k)$ if $o_1, \ldots, o_k$ all exist in ω, and null otherwise;
  - For each number statement with type τ and origin functions $g_1, \ldots, g_k$, a number RV $\#\tau[g_1 = o_1, \ldots, g_k = o_k](\omega)$ equal to the number of objects that satisfy this number statement applied to $o_1, \ldots, o_k$ in ω.
- For any BLOG model M and any complete Instantiation of V_M there is at most one correspondent model structure in Ω_M

4.Syntax and Semantics:Probabilities

- Dependency and number statements specify exactly how the steps are carried out in our generative process
  - State(a,t)
    if t = 0 then ~ InitState( ) ;
    else ~ StateTransition(State(a , Pred(t))) ;
  - Applied for every basic RV of the form State[a,t]
  - Distribution for State[a,t] is given by the *elementary CPD*
  - Syntax for the first clause : if *cond* then ~ *elem-cpd* (*arg1, . . . , argN*)
- A BLOG model defines a certain Bayesian network(BN) over the basic RVs.
- We write σ to denote an instantiation of a set of RVs vars(σ), and σ_X to denote the value that σ assigns to X. If a BN is finite, then the probability it assigns to each complete instantiation σ is $P(\sigma) = \prod_{X \in vars(\sigma)} p_X(\sigma_X \mid \sigma_{pa(X)})$, where p_X is the CPD for X.

(2)

- In an infinite BN, we can write a similar expression for each finite σ that is closed unter the parent relation($X \in vars(\sigma) \rightarrow pa(X) \subseteq vars(\sigma)$)
  If the BN is acyclic and each variable has finitely many ancestors, then these probability assignments define a unique distribution.
- The difficulty is that in the BN corresponding to a BLOG model, variables often have infinite parent sets. e.g.



(3)

- An instantiation σ supports a basic RV V of the form $f[o_1,...,o_k]$ or $\#\tau[g_1 = o_1,...,g_k = o_k]$ if all possible worlds consistent with σ agree on (1) whether all the objects $o_1,...,o_k$ exist, and, if so, on(2) the applicable clause in the dependency or number statement for V and the values for the CPD arguments in that clause.
  - e.g the instantiation (BallDrawn[d] = (Ball,13), TrueColor[(Ball,13)] = Blue) determines the value of the sole CPD argument TrueColor(BallDrawn(d)). it supports the variable ObsColor[d],
  - ObsColor(d)
    if (BallDrawn(d) != null) then
    ∽ TabularCPD[[0.8 , 0.2] , [0.2 , 0.8]](TrueColor(BallDrawn(d)))

(4)

- A finite instantiation σ is self-supporting if its instantiated variables can be numbered $X_1,...,X_N$ such that for each $n \leqq N$, the restriction of σ to $\{X_1,...,X_{N-1}\}$ supports X_N.
- A distribution P over Ω_M satisfies a BLOG model M if for every finite, self-supporting instantiation σ with $vars(\sigma) \subseteq V_M$: $P(\Omega_\sigma) = \prod_{n=1 \text{ to } N} px_n(\sigma x_n \mid \sigma\{x_1,...,x_{n-1}\})$ where Ω_σ is the set of possible worlds consistent with σ and $X_1,...,X_N$ is a numbering of σ.
- A BLOG model is well-defined if there is exactly one probability distribution that satisfies it.

(5)

- Theorem:
  Let M be a BLOG model. Suppose that V_M is at most countably infinite, and for each $V \in V_M$ and $\omega \in \Omega_M$, there is a self-supporting instantiation that agrees with ω and includes V. Then M is well-defined.
- Proof:
  - Define a sequence of auxiliary random variables $\{ Y_n : 0 \leq n \leq |V_M| \}$ on Ω_M
  - Let $Y_0(\omega) = X(\omega)$ where X is the first basic RV that is supported by the empty instantiation.
  - For $n \geqq 1$, let $\sigma_n(\omega)$ be the instantiation ($Y_0 = Y_0(\omega),\ldots,Y_{n-1} = Y_{n-1}(\omega)$). then let $Y_n(\omega) = Z(\omega)$ where Z is the first basic RV that is supported by $\sigma_n(\omega)$, but has not already been used to define $Y_m(\omega)$ for any $m < n$.

5. Evidence and Queries

- We can use arbitrary sentences of L_M as evidence and queries but sometimes such sentences are not enough.
- We allow the user to extend the language when evidence arrives, adding constant symbols to refer to observed objects.
- e.g. given four radar blips at time 8,one can assert
  - {Blip r : Time(r) = 8} = {Blip1, Blip2, Blip3, Blip4};
  - This asserts that there are exactly four radar blips at time 8, and introduces new constants in one-to-one correspondence with those blips.
- The macro augments the model with dependency statements for new symbols. We have:

  Blip1 ~ Uniform({ Blip r : (Time(r) = 8) });
  Blip2 ~ Uniform({ Blip r : (Time(r) = 8) & (Blip1 != r) });

  and so on.

6.Inference

- The generative process intuition suggests a rejection sampling algorithm.
- Suppose we are given a partial instantiation e as evidence, and a query variable Q. To generate each sample, our algorithm starts with an empty instantiation σ. Then it repeats the following steps:
  - Enumerate the basic RVs in a fixed order until we reach the first RV V that is supported by σ but not already instantiated in σ
  - Sample a value v for V according to V's dependency statement; and augment σ with V = v. The process continues until all the query and evidence variables have been sampled.
  - If the sample is consistent with the evidence e, then the program increments a counter N_q , where q is the sampled value of Q, otherwise it rejects the sample. after N accepted samples, the estimate of P (Q = q | e) is N_q / N.

(2)

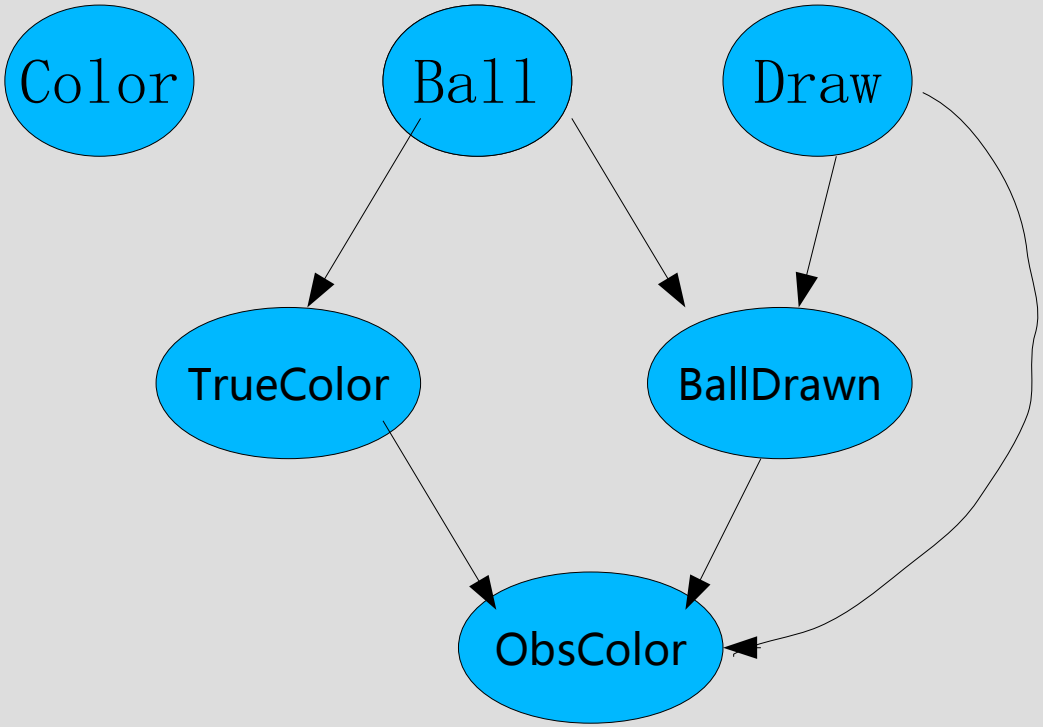
- The algorithm requires a subroutine that determines whether a partial instantiation σ supports a basic RV V.
  - For a basic RV V of the form $f[o_1,...,o_k]$ or $\#\tau[g_1=o_1,...,g_k=o_k]$, the subroutine begins by checking whether all of $o_1,...,o_k$ exist.
  - If some of these number variables are not instantiated,then σ does not support V.
  - If some of $o_1,...,o_k$ do not exist, then return the default value for V.
  - If they do exist, then the subroutine follows the semantics for dependency statement. It iterates over the clauses until it reaches a clause whose condition is either undetermined or determined to be true given σ.
  - If the condition is undetermined ,then σ does not support V.
  - If it is determined to be true, then the subroutine evaluates each of the CPD arguments in this clause.
  - If σ determines the values of all arguments then sample a value for V. otherwise σ does not support V.

Termination Criteria

- When can we be sure that the algorithm will take a finite amout of time?
  - The first way this process could fail to terminate is if it goes into an infite loop while checking whether a particular variable is supported.
  - The sample generator also fails to terminate if it never constructs an instantiation that supports a particular query or evidence variable.
- The solution is to define a symbol graph: the symbol graph for a BLOG model M is a directed graph whose nodes are the types and random function symbols of M, where the parents of a type τ or function symbol f are
  - The random function symbols that occur on the right hand side of the dependency statement for f or some number statement for τ
  - The types of variables that are quantified over in formulae or set expressions on the right-hand side of such a statement.

(2)

- The types of the arguments for f or the return types of origin functions for τ.



(3)

- Theorem : Suppose M is a BLOG model where
  - **1** uncoutable built-in types do not serve as function arguments or as the return types of origin functions;
  - **2** each quantified formula and set expression ranges over a finite set once origin function restrictions are taken into account;
  - **3** the symbol graph is acyclic.
    Then M is well-defined

Experimental results

- Asserting that 10 balls were drawn and all appeared blue, and querying the number of balls in the urn.
- Using a guided likelihood weighting algorithm described by Milch et al.