

# Individuals, relations and structures in probabilistic models

James Cussens

Department of Computer Science  
University of York  
Heslington, York, YO10 5DD, UK  
jc@cs.york.ac.uk

## Abstract

Relational data is equivalent to non-relational structured data. It is this equivalence which permits probabilistic models of relational data. Learning of probabilistic models for relational data is possible because one item of structured data is generally equivalent to many related data items. Succession and inclusion are two relations that have been well explored in the statistical literature. A description of the relevant statistical approaches is given. The representation of relational data via Bayesian nets is examined, and compared with PRMs. The paper ends with some cursory remarks on structured objects.

## 1 Learning from iid samples

Recall from [Cussens, 2000], the well-known correspondence between the mathematical abstractions used in statistics and the real world. This correspondence is given diagrammatically in Figure 1. This view sees Nature as a machine which probabilistically spits out data in response to questions (inputs) that we give it. In some cases (e.g. clustering, density estimation) the independent variables do not play an important role—the machine does not require any input to produce an output. This probabilistic machine has many names in the literature, it is Hacking’s “chance set-up” [Hacking, 1965] and Popper’s “generating conditions” [Popper, 1983].

This probabilistic machine is often taken to produce output by selecting its output from some population of possible outputs. Such a reconceptualisation is sometimes strained: “But only excessive metaphor makes outcomes of every chance set-up into samples from an hypothetical population” [Hacking, 1965, p. 25]. But it is pretty much hard-coded into the standard Kolmogorovian formalisation of probability. Kolmogorov’s axiomatisation defines a probabilistic model to be a probability space  $(\Omega, \mathcal{F}, P)$ . Here  $\Omega$  is the population, and outputs (actually subsets of  $\Omega$  in  $\mathcal{F}$ ) are chosen according to  $P$ .

In standard approaches to statistical inference (or ‘learning’; the terms will be used interchangeably in this paper) we assume that the observed data is composed of *independent and identically distributed (iid)* items sampled from  $\Omega$ . The homogeneity of such data permits estimation of  $P$ . To

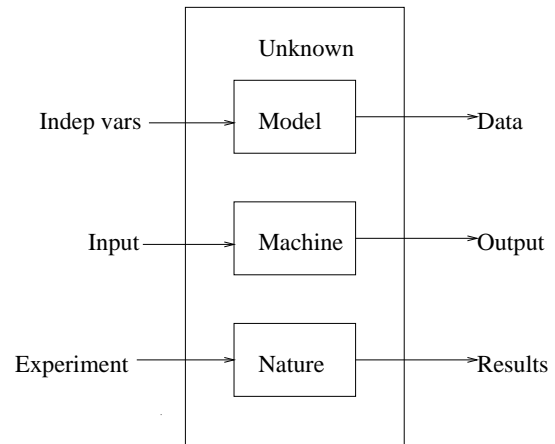


Figure 1: Statistical Inference

continue the metaphor of the machine, which will be used throughout, we assume that: each datum is generated by one run of the machine, the same machine is used for each datum, and previous outputs of the machine do not affect how it operates on future runs. The iid assumption often permits accurate estimation of parameters from data, and sometimes both parameters *and* the structure of the model/machine.

## 2 Relational learning

In many cases we are presented with data where the iid assumption is invalid. In such situations, let us say that we are faced with a *relational learning* problem, on the grounds that the items of data will be related in some way. What is perhaps new in current AI research in this area is that this issue is being approached in a *general* way: formalisms—often related to first-order logic—are being created where data items may be related in an arbitrary manner. However, there exists valuable work in the statistical literature which focuses on particular relationships between data. In Sections 2.1 and 2.2 we examine two specific relations: respectively, that of succession and the “isa” relation which forms the basis of hierarchical models.

## 2.1 Succession

Time-series analysis is a venerable form of relational learning with a large literature. It is often applied to financial data where, say, the price of pork bellies today (let's call it  $X_i$ ) is not independent of its price yesterday ( $X_{i-1}$ ). So, although  $X_i$  and  $X_{i-1}$  may be identically distributed they will not be independent, hence the data is not iid. Returning to the metaphor of the machine: we have the same machine each time, but the output of the last run forms part of the input of the next run.

Given that the convenient iid assumption is lacking how is learning possible? To answer this it is useful to make a quick detour into the mathematical formalism. A time-series is modelled as a *stochastic process* which is defined “as a family of random variables  $\{X_i, i \in I\}$  defined on some probability space  $(\Omega, \mathcal{F}, P)$ ” [Brockwell and Davis, 1991, p. 8]. The index set  $I$  may be discrete (as in the case of daily commodity prices) or continuous.

In learning, our goal is to estimate the underlying probabilistic model from data. Since here this model is a stochastic process it looks as if each data point must be a realisation of the stochastic process, i.e. each data point is to be a joint instantiation of all the  $\{X_i, i \in I\}$ . By running this sequence-generating probabilistic machine many times over we could get an iid sample each element of which is a joint instantiation. Unfortunately, in general, the machine is run just once. We will only get one such data point—since we cannot repeatedly rewind history and observe, say, the price of pork bellies on 24th Jan 1999 many times over.

But learning is still possible if we assume the joint distribution of the  $\{X_i, i \in I\}$  is *structured*. Take the simplest AR(1) model (AR(1) is also known as a Markov process):

$$\forall i : X_i \sim \alpha X_{i-1} + \epsilon_i$$

where the  $\epsilon_i$  are iid. Now the observation of each  $X_i$  becomes a data point and contributes towards the estimation of  $\alpha$ . The point is that there is a *repetitive* structure. It is the same (unknown)  $\alpha$  for all  $X_i$ . There are two further things to note here. Firstly, individuals (for example days) are not explicitly represented; only attributes of individuals, such as the price of pork bellies on that day. These attributes (the  $\{X_i\}$ ) are directly connected without any intervening individuals of which they are the attributes. Secondly, this description requires quantification over random variables, without time-series analysts requiring a new ‘first-order’ probabilistic formalism.

## 2.2 Hierarchy

Sequence data is not the only case where iid assumptions break down. Consider the following situation (where “an assumption of exchangeability” is essentially an iid assumption, and for “covariate” read “attribute”):

... in studying scholastic achievement we may have information about individual students (for example, family background), class-level information (characteristics of the teacher), and also information about the school (educational policy, type of neighborhood). ... With covariates defined at multiple levels, the assumption of exchangeability of

units or subjects at the lowest level breaks down, even after conditioning on covariate information. The simplest extension from a classical regression specification is to introduce as covariates a set of indicator variables for each of the higher-level units in the data—that is, for the classes in the educational example ... But this will in general dramatically increase the number of parameters in the model ... [Gelman *et al.*, 1995, p. 366]

Here it is not good enough to produce a regression model (probabilistically) mapping information specific to an individual to scholastic achievement for that individual. We also have (in ILP speak) background knowledge which is not specific to individuals. Each student is a member of a particular class, each class is contained within a particular school and each school is in a particular neighbourhood. Each of these levels in this hierarchical setup will have attributes which will affect the student's scholastic achievement.

Prefiguring a little the discussion of PRMs in Section 2.4, we can imagine a relational database system with tables for **Student**, **Class**, **School** and **Neighbourhood**. Each of these tables will have fields for information specific to objects of that class, so-called “descriptive attributes” [Getoor *et al.*, 2001]. Following [Neville *et al.*, 2003], we will call these *intrinsic* attributes. There will also be fields for “foreign keys” which contain the names of related objects from different classes. For example, **Student** might have fields for attendance and age as well as a foreign key field naming the class that each student is in.

The salient relationship in this case is that of inclusion or membership: each student is a member of a class, etc. Note also that students in the same class are related to each other simply by being members of the same class. Students in the same school or neighbourhood are also related, but presumably these relationships are weaker.

The option of representing all this information in a manner identical to that for individual-specific information is rejected. This rejected option corresponds to ‘propositionalisation’ to use ILP speak again. It will increase the number of parameters because it will increase the length of the covariate vector considerably. The same “exploding attribute-space” phenomenon tends to occur when ILP learning scenarios are propositionalised. An appropriate probabilistic model is a hierarchical one to reflect the hierarchical nature of the data:

... sensible estimation of these [the parameters in the model] is only possible through further modeling, in the form of a population distribution. The latter may itself take a simple exchangeable or iid form, but it may also be reasonable to consider a further regression model at this second level to allow for the effects of covariates defined at this level. In principle there is no limit to the number of levels of variation that can be handled in this way. Bayesian methods provide ready guidance on handling the estimation of unknown parameters, [Gelman *et al.*, 1995, p. 366]

To make the connection between hierarchical modelling and Bayesian computation more concrete, we will consider

a particular example using the BUGS [Spiegelhalter *et al.*, 1996] system.

### 2.3 Bayesian nets for hierarchical models

Consider the following litters probabilistic model taken from [Spiegelhalter *et al.*, 1996]. We consider survival rates in two sets of pigs. Each set of pigs contains 16 litters. “We would like to assume that the survival rates in the litters within each set are similar, but not identical.” [Spiegelhalter *et al.*, 1996]. In other words, we assume that there are phenomena at the level of sets which affect survival rates. One could imagine, for example, that the two sets of litters come from two different farms. This is clearly a hierarchical set-up, but it also implies that within each set, the individual sows are related in some way.

Suppose we want to compute the probability that a piglet, born to some particular sow, will survive. If we have observed survival rates for the litters of other sows in the same set, this should effect the value of the probability we are trying to compute. How can this be done?

Here is the approach given in [Spiegelhalter *et al.*, 1996]. Let the  $i$ th ( $1 \leq i \leq 16$ ) sow in the  $j$ th ( $1 \leq j \leq 2$ ) set be called  $sow_{ij}$ . For each sow  $sow_{ij}$  we wish to compute  $p_{ij}$  the probability that a piglet of hers will survive. Clearly, the number of piglets born so far ( $n_{ij}$ ) and the number of those that have died ( $r_{ij}$ ) for  $sow_{ij}$  are pertinent intrinsic attributes.

“The simplest conjugate model is to assume the observed number of deaths  $r_{ij}$  in litter  $i$  of group [set]  $j$  is binomial with sample size  $n_{ij}$  and true rate  $p_{ij}$ , and then assume the true rates are drawn from a beta distribution with unknown parameters.” [Spiegelhalter *et al.*, 1996] The crucial point is that these parameters  $a_j$  and  $b_j$  are common for all sows in set  $j$ . This probabilistic model is represented as the Bayesian net given in Figure 2. Note the use of ‘plate’ notation in order to compress the representation. The square box around  $n_{ij}$  indicates that  $n_{ij}$  is always assumed instantiated so no distribution need be defined for it. The corresponding BUGS language source code is given in Figure 3.

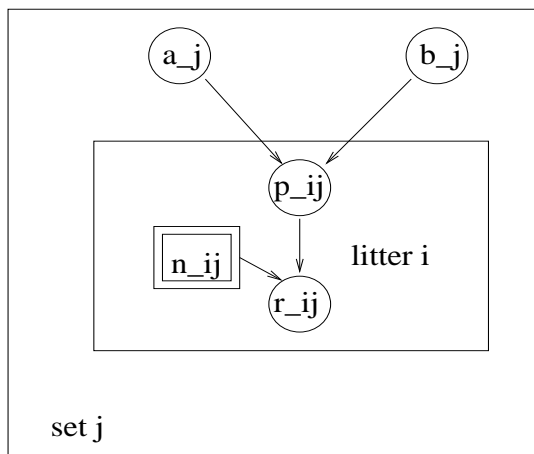


Figure 2: Bayesian net representing a hierarchical model [Spiegelhalter *et al.*, 1996]

```

for (j in 1:2) {
  for (i in 1:16) {
    r[i,j] ~ dbin(p[i,j],n[i,j]);
    p[i,j] ~ dbeta(a[j],b[j]);
  }
  a[j] ~ dgamma(1,.001);
  b[j] ~ dgamma(1,.001);
}

```

Figure 3: BUGS language representation of the Bayesian net given in Figure 2 [Spiegelhalter *et al.*, 1996]

What is interesting here is that

1. no individuals are explicitly represented in the model, only attributes of individuals;
2. consequently, relationships between individuals (such as might be represented by a foreign key relationship) can not be explicitly represented, so the membership relationship between a sow and her set is not represented nor is the derived relationship between sows in a given set;
3. the individual sows in set  $j$  are related via a very abstract quantity—the parameter vector  $(a_j, b_j)$
4. it is essential that the mediating quantity  $(a_j, b_j)$  is uninstantiated

The BUGS documentation has many other good examples of hierarchical models; the current example is one of the simpler ones.

Returning to our machine metaphor, we can say that since each litter has its own probability ( $p_{ij}$ ) for piglets’ surviving, there is a separate machine (specified by  $p_{ij}$ ) for each litter which ‘tosses a coin’ and decides the fate of each piglet. However, within each set these machines are related. We model this by imagining that there is a machine-outputting machine for each set, specified by  $(a_i, b_i)$ , which outputs the  $p_{ij}$  machines.

When concocting this probabilistic model it seems inconceivable that the statistician did not have particular individuals (sows, piglets) and classes (litters, sets) in mind. But by the time we have the probabilistic model all individuals have been eliminated. They merely have a ghostly presence in the indices of the random variables. It would be useful if we could find a way of formalising this elimination. To explore this question we now turn to a probabilistic formalism where individuals *are* explicitly represented.

### 2.4 Probabilistic relational models

The ingredients of PRMs [Getoor *et al.*, 2001] are as follows. First consider *relational schemas*. A relational schema specifies a set of classes  $\mathcal{X} = X_1, \dots, X_n$ . With each class  $X_i$  there is associated a set of *descriptive attributes* (i.e. attributes which individuals in that class can have) and *reference slots* these are ‘attributes’ whose values are the names of individuals in other classes related to individuals in this class. An *instance* of a schema defines (i) a set of individuals partitioned between the classes  $\mathcal{X}$  and (ii) and values for all attributes

(real ones and instantiations of reference slots) of all individuals. A *relational skeleton* is a partial definition of an instance where only the individuals and the relations are given—the descriptive attributes are left uninstantiated. A PRM specifies a conditional probability distribution over the values of each descriptive attribute, so that given a relational schema, the PRM defines a distribution over *completions* of the skeleton. A completion of a skeleton is an instance of the schema.

PRMs are a relational ‘upgrade’ of Bayesian networks. Given that in Section 2.3 we have argued that at least some relational learning problems can be represented using plain old Bayesian networks, we need to examine the claimed differences between PRMs and Bayesian networks:

However, there are two primary differences between PRMs and Bayesian networks. First, a PRM defines the dependency model at the class level, allowing it to be used for any object in the class. In a sense, the class dependency model is universally quantified and instantiated for every element in the class domain. Second, the PRM explicitly uses the relational structure of the model, in that it allows the probabilistic model of an attribute to depend also on attributes of related objects. The specific set of related objects can vary with the skeleton  $\sigma$ ; the PRM specifies the dependency in a generic enough way that it can apply to an arbitrary structure. [Getoor *et al.*, 2001]

The first difference is inessential from a mathematical point of view, despite being important for practical model-building. We have seen that grouping together random variables associated with objects of the same class (graphically via the plate notation, or in the BUGS language using `for`) achieves the same effect. In reply one could argue that using the BUGS language is a move beyond ‘plain old Bayesian nets’ since it explicitly uses the quantification alluded to by [Getoor *et al.*, 2001].

The second difference is more fundamental in that a PRM holds off from giving enough information to construct an equivalent Bayesian net—the missing information is contained in the skeleton  $\sigma$ . The BUGS analogue is a partially specified Bayesian net, where, for example, the actual numbers of sows and piglets are yet to be determined.

Learning in PRMs assumes the data is one single structured datum:

Our training data consists of a fully defined instance of that schema. We assume that this instance is given in the form of a relational database. [Getoor *et al.*, 2001]

So, as with time-series, the data is a single instance drawn from the underlying distribution. It is only because this instance is highly structured and hence composed of many related ‘instances’ that there is enough information to do parameter estimation, or possibly even model structure learning.

## 2.5 Eliminating and introducing individuals

It would be interesting to see whether there is an algorithm which eliminates the individuals in a PRM with a given skele-

ton  $\sigma$  thus rendering a Bayesian net (with repetitive structure) containing only “descriptive attributes”. It would be even more interesting to determine the advantages and disadvantages of such a ‘compilation’.

None of this is attempted here. Instead we just give one example of moving in the opposite direction. In Figure 4, we give a RDB presentation of the `litters` example. Adding the CPTs (extractable from Figure 3) for  $P(\text{Piglet.Lives}|\text{Piglet.Mother.Health})$ ,  $P(\text{Sow.Health}|\text{Sow.Set})$ ,  $P(\text{Set.A})$  and  $P(\text{Set.B})$  gives us a PRM with a given skeleton.

Piglet			Sow		
Name	Lives	Mother	Name	Health	Set
pinky	?	mary	mary	?	1
perky	?	mary	susy	?	1
squeaky	?	susy	anny	?	2
quirky	?	susy	...	...	...
...	...	...			

Set		
Name	A	B
1	?	?
2	?	?

Figure 4: Relational database representation of the BUGS `litters` scenario

Comparing Figure 4 with Figures 2 and 3 the with-individuals RDB approach of PRMs seems to me to have two main advantages. Firstly, the world simply does contain individuals of various classes, and consequently this is how we conceptualise it. On this count Figure 4 is the more perspicuous to a human modeller. Secondly, and for related reasons, RDBs *are where the real-world data is*, so for entirely practical reasons a probabilistic model that can be bolted on to a RDB has a lot going for it. The advantage of the BUGS approach is that by eliminating the individuals we have gained some simplicity, or at least compactness. On a practical point the BUGS MCMC-based software is also quite well developed. All these observations indicate that ‘compiling’ PRMs to structured Bayesian nets may have much to recommend it.

## 3 Structured objects versus systems of objects

The slogan of this paper has been that relational data is equivalent to non-relational structured data. However, in the examples given the structured ‘data’ is a single big data-point: an entire sequence, an entire hierarchy or an entire relational database (or at least completion thereof). A less extreme kind of structured data is data composed of a number of structured objects. In place of a conclusion, in this final section we briefly consider the representation of structured objects and connections to relational data. A thorough examination of these issues and their consequences for relational learning we leave for future work.

Consider RDBs. An RDB is a system of related atomistic objects where each individual object is ‘flat’. It has its own intrinsic attributes and its foreign keys name related objects.

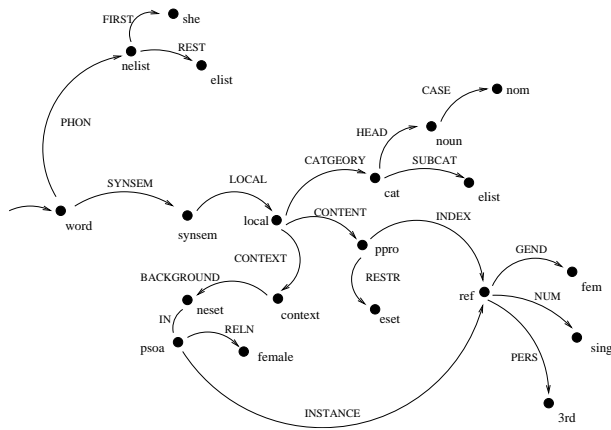


Figure 5: Lexical entry for the word “she” [Pollard and Sag, 1994, p. 17]

In logic programming terms RDBs are ground Datalog programs. However, one could argue that a putatively atomistic object (say  $a$ ) with relations to other atomistic objects (say  $b$  and  $c$ ) in fact has a structure such that  $b$  and  $c$  are in fact ‘constituents’ of  $a$ . If so, it follows that it is more perspicuous to represent  $a$  as  $f(b, c)$  where  $f$  represents how  $b$  and  $c$  constitute  $a$ . Here, the information about  $a$  has been packed into a first-order term so that a mere identifier ( $a$ ) has been replaced by something more like a description. The trade-offs between these two forms of representation have long been discussed in the ILP literature. For example Rouveirol [Rouveirol, 1994] shows how to ‘flatten’ structure representations.

An extreme example of this sort of packing occurs in lexicalised approaches to natural language grammar such as Head-Driven Phrase Structure (HPSG). In a lexicalised grammar nearly all grammatical information is represented at the word level on the grounds that words are information-rich, and should be represented as such. HPSG grammars present linguistic objects as feature-structures which are very highly structured objects. For example, Figure 5 gives a (slightly cut-down) HPSG lexical entry for the word “she” [Pollard and Sag, 1994, p. 17].

Here each node is labelled with a *sort* and the arcs correspond to features which those sorts have. Some sorts (such as *nom*) do not have features; they are called atoms. It is clear how a feature-structure could be converted to an equivalent RDB/Datalog program (in fact, this is more or less done in [Pollard and Sag, 1994]). Each arc going to a non-atom sort represents a foreign key relation; those going to atoms represent intrinsic attributes. The interesting thing here is that there is no clear distinction drawn between individuals and attributes: they are all sorts. Analogously, reference slots and descriptive attributes are all fields.

## References

[Brockwell and Davis, 1991] Peter J. Brockwell and Richard A. Davis. *Time Series: Theory and Methods*. Springer Series in Statistics. Springer, New York, second edition, 1991.

[Cussens, 2000] James Cussens. Attribute-value and relational learning: A statistical viewpoint. In Luc De Raedt and Stefan Kramer, editors, *Proceedings of the ICML-2000 Workshop on Attribute-Value and Relational Learning: Crossing the Boundaries*, pages 35–39, 2000.

[Gelman *et al.*, 1995] Andrew Gelman, John Carlin, Hal Stern, and Donald Rubin. *Bayesian Data Analysis*. Chapman & Hall, London, 1995.

[Getoor *et al.*, 2001] Lise Getoor, Nir Friedman, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In Sašo Džeroski and Nada Lavrač, editors, *Relational Data Mining*, pages 307–335. Springer-Verlag, September 2001.

[Hacking, 1965] Ian Hacking. *The Logic of Statistical Inference*. Cambridge University Press, Cambridge, 1965.

[Neville *et al.*, 2003] Jennifer Neville, Matthew Rattigan, and David Jensen. Statistical relational learning: Four claims and a survey. In Lise Getoor and David Jensen, editors, *IJCAI-03 Workshop on Learning Statistical Models from Relational Data*, Acapulco, Mexico, August 2003.

[Pollard and Sag, 1994] Carl Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago, 1994.

[Popper, 1983] Karl R. Popper. *Realism and the Aim of Science*. Hutchinson, London, 1983. Written in 1956.

[Rouveirol, 1994] Céline Rouveirol. Flattening and saturation: Two representation changes for generalization. *Machine Learning*, 14(2):219–232, 1994.

[Spiegelhalter *et al.*, 1996] D J Spiegelhalter, A Thomas, N G Best, and W R Gilks. *BUGS Examples Volume 1, Version 0.5, (version ii)*. MRC Biostatistics Unit, Cambridge, UK, 1996.